NODALIDA '93

Proceedings of
'9:e Nordiska Datalingvistikdagarna'
Stockholm 3–5 June 1993

Robert Eklund, editor

Stockholm 1994

This volume was sponsored by:
Humanistisk-Samhällsvetenskapliga Forskningsrådet (HSFR)
Skriptor International AB
Department of linguistics, Stockholm University

Typeset and layout:
Robert Eklund

Published by:
Department of Linguistics, Computational Linguistics
Stockholm University, S–106 91 Stockholm, Sweden

Printed by:
Akademitryck AB, Edsbruk 1994

ISBN 91-7153-262-5

Contents

Preface

Vi datorlingvister har egentligen alltid vetat om det men nu (äntligen) har omvärlden också så sakta börjat komma till insikt om det, nämligen att datorlingvistiken är ett nyckelområde för det som i dag benämns informationsteknologi, IT. För bara ett år sedan var det väl bara ett litet fåtal specialister som hade hört talas om IT, men nu plötsligt har intresset för området formligen exploderat. Den svenska regeringen har nyligen lagt fram ett visonärt program om storsatsning på IT i en nära framtid – "elektroniska highways" är slagordet – och tidningarna är fyllda med fantastiska visoner om hur informationsteknologin skall omdana vår värld och lägga grunden för vårt framtida välstånd. Med hjälp av de elektroniska nätverken kommer vi ha tillgång till all världens vetenskapliga, tekniska och kulturella information bokstavligen i våra fingerspetsar. Med multimediatekniken kan vi få bilder och ljud från alla jordens hörn direkt i våra vardagsrum, et cetera, et cetera. En fantastisk ny värld finns bara runt hörnet.

Men – det finns alltid ett men – vägen dithän är inte så där alldeles lätt. Rent tekniskt är det väl kanske inte så väldigt lång dit (men med tanke på att universitetsnäten fortfarande efter 20 år bara klarar sjubits-ASCII är åtminstone inte jag reservationslöst imponerad av den fart med vilket den tekniska utvecklingen skrider fram inom det här området), men för att hitta all den information som vi vet finns "där ute" så måste vi ha ofantligt mycket intelligentare användargränsnitt än vad vi i dag har – Gopher i sin nuvarande form är inte framtidens NLI. Vi måste ha söksystem som förstår att söka efter den information som användaren frågar efter och inte bara söka efter de informationskällor som råkar innehålla de ord som frågaren använt i sin sökfråga. Söksystemen måste också kunna finna den relevanta informationen oavsett vilket språk den finns representerad i och informationen skall kunna presenteras för användaren på ett språk som denne förstår. Utöver allt detta så finns den ytterligare dimensionen att systemen i tillämpliga fall skall kunna förstå och ta emot muntliga instruktioner och likaså i tillämpliga fall leverera sina svar i form av talat output.

För alla tillämpningar som nämnts ovan – frågebesvarande system, informationssökning, automatisk översättning samt taligenkänning och talgenerering – så finns det i dag ett nästan obegränsat behov av färdiga teknologier, det vet alla. Vi datorlingvister vet också att färdiga lösningar rätt och slätt inte föreligger, utan att det krävs ett långt och mödosamt arbete för att få något så när hyggliga lösningar. De nämnda tillämpningsområdena har alla en komponent av datorlingvistisk karaktär i sig, och det är inte en tillfällighet att för vart och ett av dem så finns det presentationer med klara implikationer för just det området i denna volym.

Även om de nämnda tillämpningarna är utomordentligt viktiga så får vi naturligtvis inte sälja vår själ. Vi har också ett inomvetenskapligt uppdrag. Men som jag ser det så behöver det inte finnas en motsättning mellan det samhälleliga behovet och detta uppdrag. Man kan fråga sig vad målet egentligen är för datorlingvistiken – om man nu kan tala om **ett** mål. En central frågeställning måste i alla fall vara att åstadkomma en semantisk modell som på ett något så när entydigt och fullständigt sätt möjliggör en automatisk härledning av betydelseinnehållet i första hand i en godtycklig mening ("sentence") och på sikt också också i en sekvens av sammanhängande meningar, alltså i en text. Inom parentes sagt, så är detta också en central frågeställning för den teoretiska lingvistiken.

Lyckas vi med det uppdraget, ja då har vi också början till lösningar för de nämnda utomvetenskapliga problemställningarna.

Benny Brodda

Topological frames in sign-based grammars

Lars Ahrenberg
Linköping

Abstract

The paper presents some ideas on how topological frames can be integrated in HPSG-like grammatical descriptions and be used for parsing. Phrase structure is taken to be purely hierarchical and is represented by the special feature DTRS. The topological frames account for basic word order constraints of major categories, while linear precedence rules account for word order constraints within the positions of a topological frame.

Introduction

In a context-free phrase structure grammar, whether augmented with features or not, a rule expresses simultaneous constraints on hierarchical and sequential relationships. Gazdar et al. (1985) showed how general rules of word order (LP-rules) could be formulated independently of hierarchical relations and, together with a set of unordered phrase structure rules (ID-rules), define a phrase structure grammar of a special form. The local tree in (1) is licenced either by the rewriting rule (2) or by the ID- and LP-rules of (3a,b).

(1) $_{VP}$[V NP PP]

(2) VP → V NP PP

(3) (a) VP → V, PP, NP; (b) V < NP, V < PP, NP < PP

Pollard & Sag (1987) developed these ideas by showing how general rules of (unordered) phrase structure can be stated within a formalism employing typed feature structures. Sequential relationships are still handled by LP-rules, but have a different domain; they no longer order dominated constituents directly, but apply to values of the special attribute PHON. The phonological expression associated with a mother must then be some permutation of the phonological expressions associated with the daughters that respect all LP-rules. An HPSG-like grammatical representation of (1) is shown in (4), where the value of PHON is determined by analogs of the LP-rules in (3b).

```
(4)   vp;
      PHON = <1 2 3>
      SYN:LOC:SUBCAT = <x>
      DTRS:HEAD = [verb; PHON = 1, SYN:LOC:SUBCAT =
      <x[np], y[np], z[pp]>]
      DTRS:CDTRS: = <y[PHON = 2], z[PHON = 3]>
```

There are problems, however, for grammars relying on LP-rules as the sole means for stating word order constraints. Languages with discontinuous constituents, such as the Scandinavian languages, and especially German, pose difficulties. There have accordingly been many proposals to augment LP-rules in various ways. Reape (1989) proposes a more complex combinatoric operation, sequence union, which allows access to non-immediate daughters of a constituent, while Engelkamp et al. (1992) propose to widen the domain of LP-rules to what they call head-domains, i.e. sets of constituents consisting of a lexical head with all its complements and adjuncts. In this paper I propose instead to restrict the use of LP-rules to smaller domains, called clusters, while augmenting the grammar with another device to handle word order regularities: the topological frame. The frames encode word order regularities that are valid for a class of constituents. They can basically be thought of as formalizations of the topological schemas used by Diderichsen (1962) and several other linguists working in his tradition. A cluster can similarly be seen as a sequence of constituents occuring within a specific position (or field) of a frame.

For reasons of space the full motivations and implications of this proposal cannot be dealt with here, though see Ahrenberg (1990) for some of the motivations. Instead I will develop a small, illustrative grammar fragment to make the proposal more tangible.

Elements of the grammar

The language fragment used is small and simplified in many respects. What I propose is quite compatible with the general assumptions of HPSG, however, apart from the account of word order regularities; I assume that it is necessary to restrict the domain of word order rules in languages like Swedish and German to types. This is after all quite a natural assumption to make in a theory assuming grammars to be organized as type hierarchies. In particular, topological frames apply to phrase types while LP-rules apply to clusters.

The basic elements of the grammar are signs and clusters. While both elements have overt expressions, indicated by the attribute STRING, only signs carry substantial linguistic information, indicated by the attribute FEATS. A cluster is basically a sequence of signs, indicated by the attribute ITEMS, which is connected and contracts specific sequential

relations w r t other signs and clusters. It is often, though not always, the case that items of a cluster have a common grammatical status. Some putative examples of clusters are:

- The complements of a head, e.g *put / the books on the table*;
- A sequence of adjacent modifiers, e.g. *a / big black / building*

Signs are either phrasal or lexical (i.e. words). A phrase is distinguished from a word by having a constituent structure indicated by the attribute DTRS. The value of DTRS is a feature structure where attributes such as HEAD, SUBJ (for subject), CDTRS (complements other than subjects) and ADTRS (adverbials and adjuncts) appear. A phrase also has a structure imposed on its expression, which is registered under the attribute PATTERN. The value of PATTERN is a topological frame, i.e. a finite list of elements constructed out of strings and dominated patterns. The value of the attribute STRING is a list of strings with no embedded lists (cf. PHON of Pollard & Sag, 1987). The value of FEATS is a feature structure where we find attributes representing morphosyntactic properties such as MOOD and SUBCAT (subcategorization). A partial description of the sentence *Johan lade väskan på bordet* (John put the bag on the table) can be found in (5).

It should be observed that the phrase structure shows more branching than the topological structure. Although a verb phrase (a predicate) is part of the phrase structure, there is no distinct topological frame for it. Instead, its topology is identified with that of the clause as the two paths, PATTERN and DTRS:HEAD:PATTERN, share the same frame.

(5) main-clause;[1]
STRING = <1:Johan 2:lade 3:väskan 4:på 5:bordet>
PATTERN = p[S;< 1, 2, <>, <3, 6>, <>>]
FEATS:MOOD = decl
FEATS:SUBCAT = <>
DTRS:SUBJ = x[STRING = 1]
DTRS:HEAD = y[vp; PATTERN = p, STRING = <2 3
4 5>]
y:FEATS:SUBCAT = <x>
y:DTRS:HEAD = v[verb; STRING = 2]
v:FEATS:SUBCAT = <x[np], z[np], w[pp]>
y:DTRS:CDTRS:ITEMS = <z[STRING = 3], w[STRING =
6<4 5>]>
w:PATTERN = [PP;<4, <5>>]
w:FEATS:SUBCAT = <u[np]>
w:DTRS:HEAD = [prep; STRING = 4]
w:DTRS:CDTRS = <u[STRING = 5]>

[1] x, y, z, ... are variables indicating structure sharing. Numbers 1, 2, 3, ... are also variables but always used for strings or patterns. Type names are written at the very beginning of a node. The types clause, np, vp and pp are all assumed to be subtypes of 'phrase', while v is a subtype of 'word'. The clause frame is assumed to have five positions. Its structure is further explained below.

The grammar as a whole defines the possible grammatical descriptions. In addition to the feature structures representing individual words, the grammar contains rules describing hierarchical and sequential relations and principles applying across rules. Every phrase structure rule expresses a relation between values of the attributes STRING, PATTERN, FEATS and DTRS for a local phrase, comprising a dominating item (a mother) and one or more items that it dominates (the daughters). The string of the unit can actually be computed from the pattern by a simple function. The relation between the string and the pattern of a phrase thus need not be specified for each individual rule. However, if the grammar is supposed to be used by a parser, we need to go in the opposite direction, which is not as simple. There are many patterns that yield the same string; e.g. the patterns <np v e <pp> e>, <np v <pp> e e>, <np v e e pp>, <np v e e <pp>>, where 'e' represents the empty string, all yield the string <np v pp>. Moreover, to filter out hypotheses we also need access to information about features and constituent structure.

For this reason it is probably a good idea to compile the grammar into a form which allows efficient parsing. In the end we would like an automatic compiler, of course, but here I can only illustrate how the topological frames can be taken as the basis for an augmented context-free grammar, using a PATR-style notation. Thus, I will simultaneously develop two sets of rules. The first set, the base grammar, applies to items which are daughters of the same node in phrase structure, while the second set, the string grammar, applies to units which are adjacent in the string.

A string grammar of the chosen format can be parsed in different ways. As will be evident there is a close relationship between the string grammar and ATNs with sub-networks corresponding to positions. Our current implementation, however, uses a bidirectional chart-parser, with a mixed strategy. Predictions are made bottom-up when heads are encountered. From there, parsing continues top-down and inside-out with material appearing to the left of the head being consumed before material to the right. In this way the information associated with the head can be exploited to full advantage. As the parser is still being developed, it is too early to report any results on its behaviour.

Combinatorics

Although the phrase structure rules cannot be stated with the same level of generality as in HPSG, they are far more general than an ordinary phrase structure grammar. Moreover, principles such as the Head Feature Principle and the Subcategorization Principle can still apply.

An assumption we will make is that lexical heads have fixed positions within the frames. In our example grammar the frame for the Swedish main clause will have five positions, where the second position is occupied by a (finite) verb and nothing else. Its structure, with type constraints associated with positions, is displayed in (6).

(6) The main clause schema (S):
<phrase, verb, cluster, cluster, cluster>

For ease of reference the positions will be called the foundation (F), the V2-position, the nexus field (N), the complement field (C) and the adverbial field (A), respectively.

For parsing purposes the lexical head is a good predictor for the occurrence of a projection. Given a finite verb it is a good chance that it is part of a main clause. In the string grammar we merge the positions appearing on either side of the lexical head and use (upper case) labels for sequences of clusters, as in (7).

(7) String grammar: main clauses (categorial part)
s → F v NCA_1

Here s and v represent strings of the indicated sign types, while F represents the contents of the foundation, and NCA_1 represents the joint contents of the last three positions. We can think of the upper case labels as representing a state of a top-down parser. This state is given by a current position (here indicated by the first letter of the label) and a state associated with parsing that position (indicated by the number attached to the label, if any).

As illustrated in (5), a constituent corresponding to a traditional predicate, is assumed, i.e. a VP consisting of a verb and all of its complement except the subject. This constituent is formed according to the following rule:

(8) Base grammar: Finite VPs in main clauses

```
vp;
PATTERN = [S;<1, 2, 3, 4, 5>]
FEATS:FIN = yes
DTRS:HEAD = w[verb;FEATS:SUBCAT=cons(x,t),STRING=2]
DTRS:CDTRS = u[cluster; ITEMS = t, STRING = 4]
```

The rule should be interpreted basically in the same way as an HPSG grammar rule, it states one way in which a phrase can be formed, in this case one option for the expression of finite VPs in Swedish, with the lexical head linked to the V2-position and the complements linked to the C-position. Thus, the relation between phrase structure and topology is

accounted for by a specific mapping between the daughters of the phrase and the positions of the frame.

The relation between phrase structure and subcategorization information follows the Subcategorization Principle (Pollard & Sag, 1987: 71). If a verb is subcategorized for a subject, an object and a prepositional object, as the verb *lägga* (put), we can augment (8) with

```
w:FEATS:SUBCAT = <x:np[nom], y:np[obj], z:pp>
```

Then, the Subcategorization Principle accounts for the following additional information to unify with (8):

```
FEATS:SUBCAT: = <x>
u:ITEMS = <y, z>
```

When we look at this rule from the point of view of the string grammar, we see that it involves non-adjacent positions. The part of the rule concerned with the V2-position is already covered by (7), but the role of the verb and the complement position must also be accounted for. Moreover, we need to do this in a way that ensures that the dependencies between verb and complements are maintained. To accomplish this we first extend (7) with some equations:

(7') String grammar: main clauses

$s \rightarrow F \; v \; NCA_1$

1:SOURCE = 3:SOURCE = 0

0:DTRS:HEAD:DTRS:HEAD = 2

The first pair of equations links the cluster categories to the clause via the attribute SOURCE. Through the third equation they are also linked to the head. The third equation states that the lexical head is two levels below its resulting projection. This is not necessarily always the case, but we make this simplifying assumption here.

The source will be inherited by all other concerned cluster categories. For instance we have a rule admitting an empty nexus position:

(9) String grammar: Empty nexus rule

$NCA1 \rightarrow CA_1$

0:SOURCE = 1:SOURCE

For clusters having complements as initial parts, we will have rules of the following form:

(10) String grammar; Complements in main clauses[1]

```
CAi  →  xp CAj
0:SOURCE = 2:SOURCE
0:SOURCE:DTRS:PRED:DTRS:CDTRS:ITEMS > 1

CAi  →  xp A1
0:SOURCE = 2:SOURCE
0:SOURCE:DTRS:PRED:DTRS:CDTRS:ITEMS > 1
```

These rules are actually schemas that cover a number of rules which together describe the possibilities for complementation in the language. They should be interpreted as follows: in position C of the clause schema, in state i, a category xp is possible, provided no more complements follow, or only complements allowed in state j of position C. The exact number of rules will depend on how we use the LP-rules. If the LP-rules are taken as a separate component of the string grammar, there will be a relatively small number of rules, but if we want the string grammar to respect the LP-constraints we can encode their effect in the states of the cluster categories.

When a finite VP combines with a subject a complete clause is generated. The position of the subject depends on the type of clause. In the case of unmarked declarative clauses (and the corresponding wh-clauses) it is placed in the first position, while in other clauses, including interrogatives and topicalized clauses, it is placed in the third position.

(11) Base grammar: Subjects in unmarked main clauses

```
main-clause;
PATTERN = [S; <1, 2, 3, 4, 5>]
FEATS:MOOD = unm
DTRS:HEAD = [vp; SUBCAT = <x>]
DTRS:SUBJ:STRING = 1
```

(12) Base grammar: Subjects in inverted main clauses

```
main-clause;
PATTERN =  [S; <1, 2, 3, 4, 5>]
FEATS:MOOD = inv
DTRS:HEAD = [vp; SUBCAT = <x>]
DTRS:SUBJ:STRING < 3
```

In (11) the subject string is identified with the string of the first position, as it is a unary position. In (12) on the other hand, the subject is merely included among the elements forming the third position cluster and its sequential order will be determined by LP-rules.

For the application of these rules a language-specific principle is supposed to be at work, the Frame Unification Principle, which says that

[1]The symbols '<' and '>' indicate membership of a list.

a non-maximal projection must share its topological frame (and hence basic rules for linearization) with a maximal projection.[1]

(13) The Frame Unification Principle

```
[phrase; DTRS = [headed-phrase;]] ⇒
[PATTERN = DTRS:HEAD:PATTERN]
```

Thus, the complement rule (7) and the subject rules combine to fill one and the same schema with orthographic material.

The corresponding rules of the string grammar are as in (14) and (15):

(14) String grammar; Subject in unmarked clauses.

```
F  →  np
0:SOURCE:DTRS:SUBJ = 1
0:SOURCE:FEATS:MOOD = unm
```

(15) String grammar; Subject in inverted clauses.

```
NCAi  →  np NCAj
0:SOURCE:DTRS:SUBJ = 1
0:SOURCE:FEATS:MOOD = inv
0:SOURCE = 2:SOURCE

NCAi  →  np CA1
0:SOURCE:DTRS:SUBJ = 1
0:SOURCE:FEATS:MOOD = inv
0:SOURCE = 2:SOURCE
```

When an adjunct combines with a head it will also end up in some position of the head's topological frame, although from a syntactic/semantic point of view the head often functions as a kind of argument to the adjunct. The following rule gives one account of the placement of sentence adverbs in Swedish. (Many other solutions are of course possible.)

(16) Base grammar: Sentence adverbs

```
main-clause;
PATTERN = [S;<1, 2, 3, 4, 5>]
DTRS:HEAD = h[main-clause;]
DTRS:ADTRS:ITEMS > x[sadv; STRING < 3]
```

There are similar rules placing adjuncts in the first and fifth positions of a main clause.

---

[1] In addition to unification of complete frames there is also the possibility of unifying positions of two frames with one another. There seems to be little use for this in a Swedish grammar, but for the scrambling phenomena of German, it could turn out to be useful. In these sentences, all complements of verbs in a chain of verbs dominating each other turn up in the same position, the Mittelfeld.

As for the string grammar we have the following corresponding rules, saying that a sentence adverb can be accepted in any state associated with the nexus position and be followed by anything accepted in that state, including nothing.

(17) String grammar; Sentence adverbs.

```
NCA_i  → sa NCA_i
0:SOURCE:DTRS:ADTRS:ITEMS > 1
0:SOURCE = 2:SOURCE

NCA_i  → sa CA_1
0:SOURCE:ADTRS:ITEMS > 1
0:SOURCE = 2:SOURCE
```

References

Ahrenberg, Lars. 1990. *A Grammar Combining Phrase Structure and Field Structure.* PROCEEDINGS OF COLING-90, VOL. 2: 1-6.

Diderichsen, Paul. 1962. *Elementær Dansk Grammatik.* Third Edition, Gyldendal, København.

Engelkamp, Judith, Gregor Erbach and Hans Uszkoreit. 1992. *Handling Linear Precedence Constraints by Unification.* PROCEEDINGS OF THE 30TH ANNUAL MEETING OF THE ACL.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum and Ivan Sag. 1985. *Generalized Phrase Structure Grammar.* Oxford, Basil Blackwell.

Pollard, Carl and Ivan Sag. 1987. *Information-Based Syntax and Semantics.* Volume 1: Fundamentals. CSLI LECTURE NOTES 13, Center for the Study of Language and Information.

Reape, Mike. 1989. *A Logical Treatment of Semi-Free Word Order and Bounded Discontinuous Constituency.* PROCEEDINGS OF THE 4TH MEETING OF THE EUROPEAN CHAPTER OF THE ACL, Manchester, England.

Pieces for a Global Puzzle

Jan Anward
Stockholm

My official rhetorical position in this paper, that of an ordinary linguist talking to computational linguists, is rapidly becoming obsolete. In a near future, there will be no non-obsolete ordinary linguists who are not also computational linguists, and no non-obsolete computational linguists who are not also ordinary linguists. So, in anticipation of the near future, I will talk as a linguist to other linguists about an exciting possibility that will require some cooperation between those linguists who know about language typology and historical linguistics and those linguists who know about programming and parsing.

1. Language typology and linguistic pre-history

The possibility I want to talk about concerns the use of typological databases to model linguistic (pre)-history and, ultimately, the possible initial state(s) of human language.

Typological databases are of course primarily used to study language typology: We use typological data to chart linguistic resources available to humans, to make inductive generalizations about what is a possible or typical human language, and to construct or support linguistic theories which make sense of the inductive generalizations we have arrived at.

However, through the works of Dryer (1989, 1991, 1992), Maddieson (1991) and Nichols (1992), it has become clear that there is an irreducible AREAL component in language typology. Linguistic diversity does not look the same all over the globe.

This areal component is precisely what allows us to introduce a HISTORICAL component into language typology, as well.

1.1 Nichols

In her important recent book *Linguistic diversity in space and time* (Nichols 1992), Nichols argues persuasively that present-day areal skewings of linguistic diversity can be used as a major source of insights into linguistic pre-history, allowing us to penetrate far beyond the 10 000 years visible to traditional comparative and historical linguistics.

In a survey of four broad structural features and seven grammatical categories in a carefully designed areal-genetic sample of around 170 languages, Nichols shows that there are significant differences in the distribution of these features and categories between three macroareas: the Old World (Africa, Europe, Asia), the New World (the Americas), and the Pacific (Australia, New Guinea, Oceania).

On a global scale, Nichols finds a basic contrast among chiefly head-marking languages, where grammatical relations are signalled by inflections on heads of constructions (e.g. agreement on verbs and nouns), chiefly dependent-marking languages, where grammatical relations are signalled by inflections on dependents (e.g. case on nouns), and double- or split-marking languages, where both methods of signalling grammatical relations are used. However these alternatives are not equally distributed over the globe, as can be seen from table 1: Old World languages are predominantly dependent-marking, while New World languages are predominantly head-marking, and Pacific languages are predominantly double- or split-marking.

Table 1. Head/dependent marking in macroareas. Based on Nichols (1992).
Head/dependent marking is here measured as the percentage of dependent markings (D) out of all markings of grammatical relations (dependent markings (D) + head markings (H) + detached markings (F)).

Macroarea	*Area*	*Dependent marking %*
Africa		70
Eurasia	A N East	60
	N Eurasia	64
	S + SE Asia	74
Oceania	N Guinea	50
	Australia	65
	Oceania	53
America	W North	32
	E North	32
	Meso	19
	South	37

Nichols also finds that the contrast between head- and dependent-marking is a good predictor of the distribution of her other structural features: complexity (number of inflections, essentially), alignment (how subjects and objects are marked, through case-marking and/or agreement), and word order. In both language types, moderate morphological complexity, accusative alignment (direct objects have a distinctive marking), and verb-final word order are unmarked, but head- and dependent-marking favor different marked types of complexity, alignment, and word order.

Head-marking tends to favor low complexity, stative-active alignment (agents have a distinctive marking) or hierarchical alignment (participants that are high on an animacy hierarchy have a distinctive marking), and verb-initial or free word order, while dependent-marking tends to favor high complexity, ergative alignment (transitive subjects have a distinctive marking), and verb-medial order. As a consequence, the marked types of complexity, alignment, and word order also show significant areal skewings in their global distribution.

The significant contrasts that Nichols finds between the Old World, the New World, and the Pacific (Australia, New Guinea, Oceania) indicate, in her opinion, "long-standing affinities and disparities" (Nichols 1992: 185) between these areas. Several cluster analyses reveal that inter-area divergence is greatest in the Pacific and that the greatest affinity between areas is between the Pacific and the New World. There is lesser affinity between the Old World and the Pacific and a great divergence between the Old World and the New World. These data, Nichols suggests, support a model of the peopling of the Earth, where the Old World is populated from Africa via the Near East, and then first Australia, second the New World, and finally New Guinea are populated from a center in South East Asia. Relying on archaeological evidence, Nichols dates the colonization of Australia to 50 000 years BP, and the beginning of circum-Pacific colonization to 35 000 years BP.

The mechanisms which Nichols uses to derive present-day linguistic diversity from these migrations are an assumption of initial diversity, and a model, borrowed from population genetics, where initial diversity is stabilized as populations stabilize in colonized areas. A small initial difference with respect to the presence of a feature F, say 60% +F and 40% -F, is eventually stabilized as 100% +F and 0% -F. This would mean that a small initial difference in favor of dependent-marking in the languages of the populations that remained in the Old World would eventually result in 100% dependent-marking languages in the Old World, while a small initial difference in favor of head-marking in the languages of the populations that colonized the New World would eventually result in 100% head-marking languages in the New World. None of the processes would have run their full course, though, due to, for example, insufficent time depth.

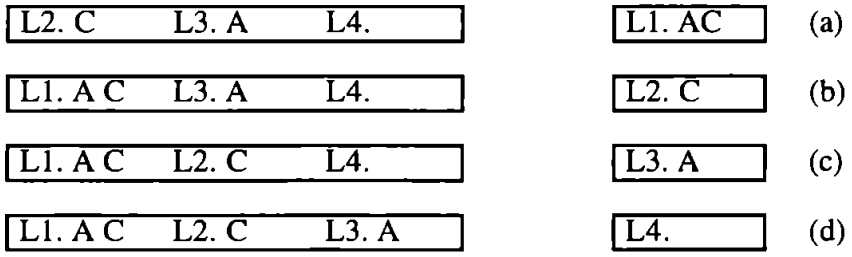
1.2. Problems with Nichols' model

Nichols' great merit is to have opened up the fascinating prospect of reading off linguistic pre-history from present-day areal skewings of various linguistic phenomena. However, Nichols' implementation of this prospect is far from satisfying.

Nichols' model of linguistic change on a global scale is essentially a spatio-temporal projection of the statistical differences she finds. As such, it abstracts away from the many local historical processes involved, subsuming them all under the single notion of levelling of initial skewings. However, as soon as we try to spell out levelling in terms of actual historical processes, it becomes clear that Nichol's model is based on a number of questionable assumptions.

Consider the following model case, where we have an area featuring four languages, two of which have case (L1, L2), and two of which have subject agreement (L1, L3). In global terms, L1 is double-marking, L2 is dependent-marking, L3 is head-marking, and L4 is zero-marking. The whole area is double-marking, having 2 instances of case (C) and 2 instances of agreement (A), or, in the measure used in table 1, 50% dependent-markings.

Suppose now the area is subject to a population split, and one of these languages 'walks away' to another, previously unpopulated area. The possible outcomes of such a split are shown below.



As we can see, population splits do not always skew linguistic diversity. When L1 or L4 walks away, as in (a) and (d), respectively, the old area retains its double-marking character, and the new area becomes double-marking, as well. In contrast, when L2 walks away, as in (b), the old area becomes head-marking, and the new area becomes dependent-marking, and when L3 walks away, as in (c), we get the opposite result: the old area becomes dependent-marking, and the new area becomes head-marking.

What might happen to the old area, after the splits in (a) – (d) have taken place? In particular, how might levelling be implemented? Nichols suggests that borrowing plays a vital role in levelling. And borrowing will indeed produce levelling, if we make the further assumption that only areally 'strong' features, i.e. features that are shared by a majority of the languages of certain area, are borrowed. If that is the case, A will spread in the old area in (b) and (d), and C will spread in the old area in

(c) and (d), reinforcing the head-marking character of the old area in (b) (from 33% dependent-marking to 25% dependent-marking), as well as the dependent-marking character of the old area in (c) (from 67% to 75%), but retaining the double-marking character of the old area in (a) and (d) (at 50%).

However, the assumption that only areally strong features are borrowed is not an uncontroversial assumption, to say the least. All empirical evidence suggests instead that any linguistic feature is capable of spread, under conditions of political or cultural dominance (Thomason & Kaufman 1988), which means that areal convergence on a certain feature need not reflect an initial skewing in favor of that feature. In our model case, then, A or C may spread in the old area in all four after-split situations, provided that they spread from a politically and/or culturally dominant language.

Another factor which may play a rôle in levelling is grammaticalization, system-internal processes whereby inflections and constructions are formed and disappear. The two standard processes in (1) produce head marking and dependent marking, respectively, in their next two last stages (see Hopper & Traugott 1993 for a review of these processes).

(1) a. Pronoun -> Agreement -> Ø
b. Noun / Verb -> Adposition -> Case -> Ø

Grammaticalization can also effect levelling, but, as with borrowing, only if it interacts in a crucial way with areal strength. If grammaticalization produces nothing but further instances of areally strong features, then it may result in A in all of the languages of the old area in (b) and (d), and in C in all of the languages of the old area in (c) and (d).

However, the assumption that grammaticalization produces just further instances of areally strong features is as untenable as the assumption that only areally strong features are borrowed. To take just the most apparent case: The first instances of agreement and case in an area can of course not be further instances of areally strong features. Thus, if the processes in (1) are indeed the only sources for agreement and case, then they must be able to introduce areally weak features.

The consequences of allowing grammaticalization to produce areally weak or even previously absent features are far-reaching. Let us spell out a possible interaction of the processes in (1), in terms of the following assumptions:

(i) Languages start out with only pronouns, nouns, and verbs, and then acquire, and lose, agreement and case through the processes in (1).

(ii) The formation of agreement is faster than the formation of case – there is one more stage involved in the formation of case.

(iii) Loss of agreement or case is much slower than their formation from pronouns and adpositions, respectively – inflections are resistant to erosion.

(iv) Restitution of agreement is about as slow as loss of agreement – a new set of independent pronouns must develop before the process in (1a) can start again.

These assumptions produce a cycle of possible language stages, shown in (2).

(2)

a.	- Agreement	- Case
b.	+ Agreement	- Case
c.	+ Agreement	+ Case
d.	- Agreement	+ Case
a.	- Agreement	- Case

Given an application of the processes in (1) that is constrained only by the assumptions of (i) – (iv), areal convergence on the feature case (stage 2c or 2d) may reflect an initial state in that area without case (stage 2a or 2b) and areal convergence on the feature agreement (stage 2b or 2c) may reflect an initial state in that area without agreement (stage 2d or 2a). Again, with more realistic assumptions about linguistic change, we find that areal convergence on a feature need not reflect an initial skewing in favor of that feature.

1.3. A proposal

I have evaluated Nichols' model of linguistic pre-history by making explicit a number of assumptions about possible linguistic changes. I would now like to suggest that this is the appropriate way forward. We should not be content with simple projections of statistical differences, but we should use what we know about linguistic change to construct precise models, based on explicit assumptions about population processes and linguistic change under various sociolinguistic conditions, which simulate how present-day diversity may arise from various postulated initial states, and thus arrive at a good guess about which initial state is the most likely one.

Such a model should of course be computational, and it should work in a computational environment, where its predictions can be tested against an actual distribution, as defined by a typological data-base, and where discrepancies between the model and the data-base lead to proposed changes in the model. Moreover, the model should have an interactive graphic interface, which permits instant illustration, on some kind of map of the globe, of actual and theoretical distributions at various times and places. Anyone who is familiar with computer games such as SimCity knows what kind of interactive graphic interface I have in mind.

The desired computational environment of the model is summarized in figure 1 below. I am grateful to Frans Gregersen for suggesting the name SimLing.

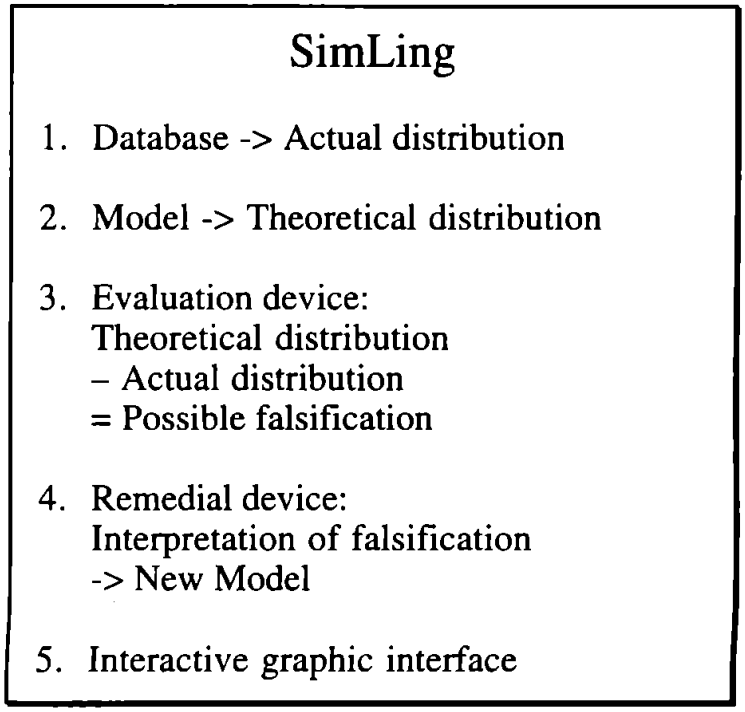


Figure 1. ***SimLing****: desired computational environment of a model of global linguistic diversity.*

2. A model of global linguistic diversity

I will now spell out some possible details of a model of global linguistic diversity, by trying to model the global distribution of two of structural features that Nichols investigates: head/dependent-marking and basic word order. I must emphasize that the specific assumptions I make are very preliminary, and should in no way be taken as established facts. My main aim is to demonstrate that a model of the kind I have in mind is a possible enterprise and to invite other linguists to think along the same lines.

The backbone of a model of global linguistic diversity is the assumption that present-day global linguistic diversity has arisen through a number of population processes which have spread successive versions of an initial state across the globe. As I have already demonstrated, this general picture must be made more precise, by means of a number of explicit assumptions about population processes and linguistic change. In addition, the initial state and its successive versions are constrained by assumptions about which expressive means are available to natural languages, and the successive versions of a particular initial state are constrained by assumptions about which discrepancies between generations can be introduced by language acquisition and language use under various social conditions, in particular the social conditions created by the assumed population processes. The general outline of a model of this kind is shown in figure 2 below.

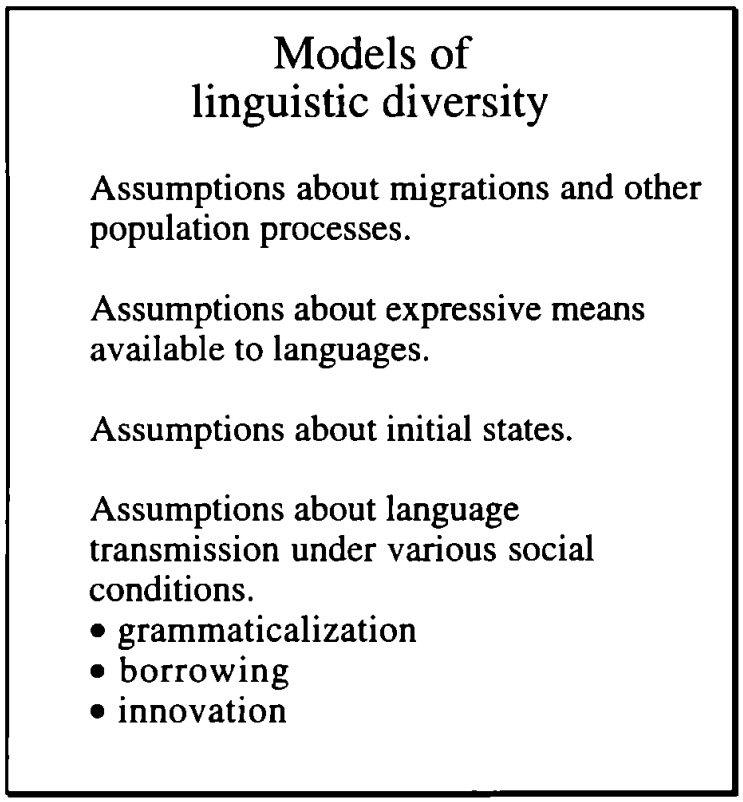


Figure 2. Components of models of linguistic diversity.

2.1. Population processes

Cavalli-Sforza (1991), summarizing a number of studies of global genetic diversity, suggests that present-day genetic diversity results from two fundamental population splits, which can be surmised to have occurred 60 – 100 thousand years ago. The first split differentiated those who stayed in Africa from those who went on to West Asia, and the second split

differentiated those who went to the North, to Europe, Central Asia, and America, from those who went to the South, to South Asia, Southeast Asia, Australia, New Guinea, and Oceania. There are many ways of incorporating these basic splits into a model of linguistic diversity. I would like to propose that the two basic splits first and foremost define a spatial network for global migrations, which is built around two centers. The first of these centers is West Asia, where the two basic splits postulated by Cavalli-Sforza took place: the split between Africa and the rest of the world and the split between the Northward migrants and the Southward migrants. The second center is East Asia, where those who stayed on in Asia were differentiated from those who went on to Australia, New Guinea, and Oceania, on the one hand, and the Americas, on the other hand. East Asia is also the meeting place of the Northward migrants and the Southward migrants. Japanese, for example, which has proved impossible to relate in a simple way to any language family, might be a very concrete instance of this meeting of North and South. According to Shibatani (1990), the most probable origin of Japanese is an Altaic (Northern) language superimposed on an Austronesian (Southern) language, possibly with a Dravidian (Southern) language sandwiched in between. The meeting of the Southern and the Northern routes may also have resulted in some Southward migrants going on to America, something which would make sense of the strong evidence for a Circumpacific linguistic area that Nichols finds in her data.

This spatial network for global migrations is shown in rough outline in figure 3 below.

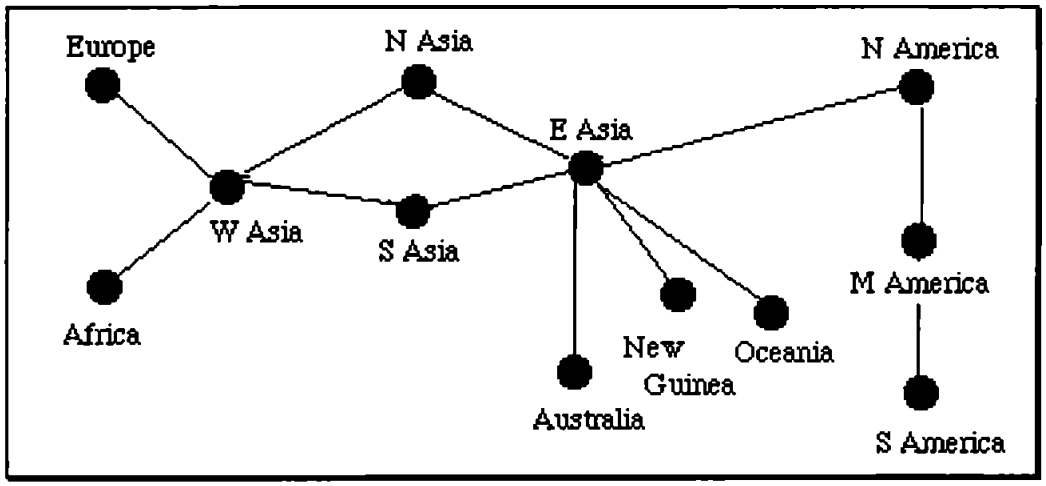


Figure 3. Spatial network for global migrations.

Renfrew (1992, 1994) has recently proposed a model of the population of the Earth, based on archaeological, genetic, and linguistic evidence. According to Renfrew, populations have spread across the globe mainly through five major waves of migration: 1. *initial colonizations*, before 40 000 BP: colonization by hunter-gatherers of previously unpopulated

areas; 2. *circumpolar dispersals*, after 10 000 BP: colonization by hunter-gatherers of areas previously covered by ice; 3. *agricultural dispersals*, after 10 000 BP: expansions, mostly into previously populated areas, in connection with the introduction and spread of agriculture; 4. *élite dominance expansions*, after 10 000 BP: invasions of previously populated areas by a military dominant élite; 5. *late colonial expansions*, after 1500: invasions of previously populated areas by a military dominant élite.

The network in figure 3 can be used to describe the first four of these waves of migration. However, after 1500, communications are reshaped in fundamental ways. First sea routes and then air routes are opened between all contintents, and printing and electronic media enable languages to travel without an accompanying population. The modern linguistic situation can hardly be put on a map. Therefore, I will treat language history up to 1500 only, and will, in this context, ignore both the spread of Indo-European after 1500, and the resulting genocides and glottocides.

Following Ruhlen (1987), we recognize 19, more or less tentative, linguistic macrogroups: In *Africa*: Khoisan, Niger-Kordofanian, Nilo-Saharan, and Afroasiatic; in *Eurasia*: Afroasiatic, Indo-European, Uralic-Yukaghir, North Caucasian, Kartvelian, Altaic, Elamo-Dravidian, Sino-Tibetan, Chukchi-Kamchatkan, and Austroasiatic; in *Australasia*: Austronesian, Papuan, and Australian; and in *America*: Eskimo-Aleut, Na-Dene, and Amerind.

In Renfrew's model, modified by the assumption of a circumpacific dispersal, these macrogroups have arrived in their present-day places in the following ways (which I will call macrogroup histories):

Initial colonization of Africa:	Khoisan
Initial colonization of SE Asia, from W Asia:	Austric
Initial colonization of America, from W Asia:	Amerind
Initial colonization of C Asia, from W Asia:	N Caucasian
Initial colonization of Australasia, from E Asia:	Australian
	Papuan
Circumpolar disperal to N Eurasia and N America, from W Asia:	Uralic
	Chukchi-Kamchatkan
	Na-Dene
	Eskimo-Aleut
Agricultural dispersal in Africa:	Nilo-Saharan, Niger-Kordofanian
Agricultural dispersal to S Asia, from W Asia:	Dravidian
Agricultural dispersal to Europe, from W Asia:	Indo-European
Agricultural dispersal in W Asia and to Africa, from W Asia:	Afroasiatic

Agricultural dispersal to C Asia, from W Asia:	Kartvelian Sino-Tibetan
Agricultural dispersal/circumpacific dispersal to Australasia and W America, from E Asia:	Austronesian Amerind
Élite dominance expansion to S Asia and C Asia, from W Asia and C Asia:	Indo-European Sino-Tibetan Altaic

Each area in the network of figure 3 can now be assigned a history, which, to simplify matters, is the union of the histories of the macrogroups that presently occupy the area. The history of Africa, for example, is the sum of the histories of Khoisan, Nilo-Saharan, Niger-Kordofanian, and Afroasiatic. Possible components of such areal histories in the model are:

IC(Africa):	Initial colonization in or from Africa before 40 000 BP
IC(W Asia):	Initial colonization from W Asia before 40 000 BP
IC(E Asia):	Initial colonization from E Asia before 40 000 BP
CD(W Asia):	Circumpolar dispersal from W Asia, after 10 000 BP
AD(Africa):	Agricultural dispersal in Africa, after 10 000 BP
AD(W Asia):	Agricultural dispersal in or from W Asia, after 10 000 BP
AD(E Asia):	Agricultural dispersal in or from E Asia, after 10 000 BP
EE(W Asia):	Élite dominance expansion from W Asia, after 10 000 BP

The actual areal histories incorporated in the model are shown in figure 4.

Areal histories

Africa:	IC(Africa), AD(Africa), AD(W Asia)
West Asia:	IC(Africa), AD(W Asia)
Europe:	IC(W Asia), AD(W Asia)
North Asia:	CD(W Asia)
South Asia:	AD(W Asia), EE(W Asia)
East Asia:	IC(W Asia), AD(E Asia), EE(W Asia)
Australia:	IC(E Asia)
New Guinea:	IC(E Asia)
Oceania:	AD(E Asia)
North America:	IC(W Asia), CD(W Asia), AD(E Asia)
Mesoamerica:	IC(W Asia), AD(E Asia)
South America:	IC(W Asia), AD(E Asia)

Figure 4. Areal histories in the spatial network. IC = Initial Colonization; AD = Agricultural Dispersal; CD = Circumpolar Dispersal; EE = Élite Expansion.

2.2. Expressive means

I want the model being developed to say something interesting about the global distribution of two of Nichols' structural features: head/dependent-marking and basic word order. To begin with, we must decide which expressive means make up these structural features.

Another merit of Nichols (1992) is that she provides a more complete picture of sentence structure options than is normally provided in studies of Universal Grammar. The extension of alignment patterns to include also stative – active alignment and hierarchical alignment and the recognition of both agreement and case-marking as exponents of alignment are necessary steps to achieve a more realistic model of sentence structure options available to natural languages.

Here I will take Nichols' argument one step further. Consider the following story, from Labov (1972):

(3) This boy punched me
and I punched him
and the teacher came in
and stopped the fight

Punch and *stop* express two-place predicates, and *come in* expresses a one-place predicate. The arguments of these predicates are characterizable in terms of thematic rôles, as in (4), and these thematic rôles form a thematic hierarchy (Jackendoff 1990, kap. 11).

(4) *come in* (Theme)
punch (Agent, Goal)
stop (Agent, Theme)

The arguments are also characterizable along two other dimensions: an animacy dimension, where referents are ranked according to closeness to speech act participants (Silverstein 1976, 1987), and a discourse flow dimension, where referents are ranked according to their topicality in the ongoing discourse. Simple thematic, animacy, and discourse flow hierarchies are shown in (5a), (5b), and (5c), respectively.

(5) a. Agent > Goal > Theme
b. Ego,Tu > Humans > Animals > Plants > Objects > Abstracts
c. Topic > Definite > Indefinite

In (3), animacy ranks the referents as: I/me > this boy, the teacher > the fight. Discourse flow ranking of the referents in (3) is not obvious, but would probably essentially agree with their animacy ranking.

The various kinds of alignment that Nichols recognizes, as well as a few more, can now be explicated in terms of how agreement, case, and word order mark positions on one or more of these hierarchies.

Consider first agreement. If there is one agreement-trigger, then we have as options at least: accusative alignment, where the trigger is the highest argument on the thematic hierarchy (θ-high); ergative or stative-active alignment, where the trigger is the lowest argument on the thematic hierarchy (θ-low); and hierarchical alignment, where the trigger is the highest argument on the animacy hierarchy (A-high), as well as the highest argument on the thematic hierarchy, unless an inverse marking on the verb shows that the A-high argument is θ-low. If there are two argument-triggers, then the second agreement-marker marks the polar opposite of the first agreement-marker. In accusative agreement, the second agreement-marker often signalizes that the trigger is high on the discourse-flow hierarchy (D-high), as well.

In (3), *this boy*, *I*, *the teacher*, and ø (the null subject of the last sentence) would be primary agreement triggers in an accusative alignment, while *me*, *him*, *the teacher*, and *the fight* would be primary agreement triggers in an ergative alignment, and *me*, *I*, *the teacher*, and ø would be primary agreement triggers in a hierarchical alignment.

A similar story can be told of case. In accusative alignment, θ-low has an overt marking, when it is distinct from θ-high; in ergative and stative-active alignment, θ-high has an overt marking, when it is distinct from θ-low (ergative) or always (stative-active). There is often a component of A-high and/or D-high in accusative case, and a component of A-low in ergative case.

In (3), *this boy* and *I* would have overt case in an ergative alignment, while *me*, *him*, and *the fight* would have overt case in an accusative alignment.

This can be summarized in a simple model, where agreement markers and case markers are taken to signal combinations of θ-high / θ-low, A-high / A-low, and D-high / D-low. And this model can be extended to word order, as well. Position before another argument, and position before or after the head can also be taken to signal such combinations. In a strict SOV-language, for example, where S must precede O, position before the head does not say anything about θ-, A- or D-value, but position before another argument signals θ-high. As is well-known, word order often signals D-value. Word order may also signal A-value.

As Nichols points out, agreement and case (and of course word order) occur in S, NP, and/or PP, with occurrence in PP implying occurrence in NP and occurrence in NP implying occurrence in S.

A rather complete parametric model of the expressive means underlying head / dependent marking and basic word order will thus include the following components:

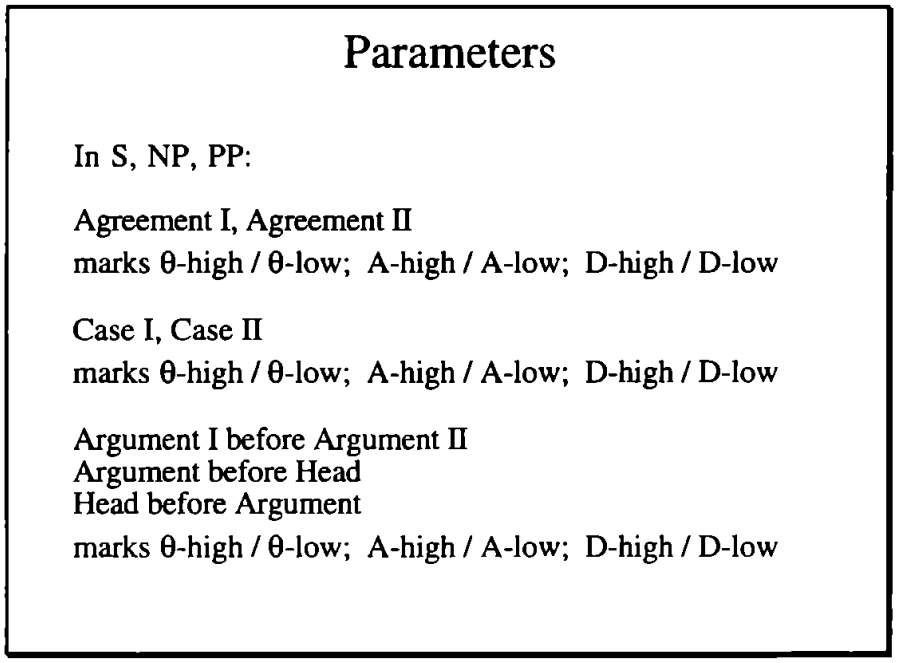


Figure 5. Parameters underlying head/dependent-marking and basic word order.

Here, though, I will use an extremely simple parametric model, with only four parameters: Presence (+) or absence (-) of agreement in S, presence (+) or absence (-) of case in S, verb before object (+VO) or object before verb (-VO), and verb before subject (+VS) or subject before verb (-VS). I assume, contrary to fact, that subject always precedes object. [+VO; +VS] then sets basic word order to VSO, [+VO; -VS] sets basic word order to SVO, [-VO; +VS] is excluded, and [-VO; -VS] sets basic word order to SOV. This simplified parametric model is summarized in figure 6.

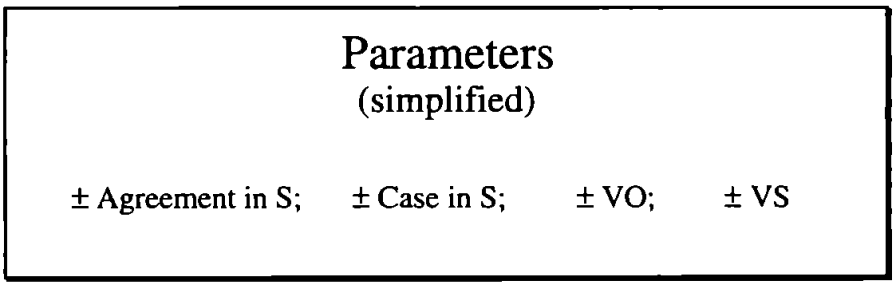


Figure 6. Parameters underlying head/dependent-marking and basic word order (simplified).

2.3. Global distribution of expressive means

The areal distributions in Nichols' sample of the simple parameter values of figure 6 are shown below in figures 7 and 8. Figure 7 shows the amount of head-marking (agreement) and dependent-marking (case) in S.

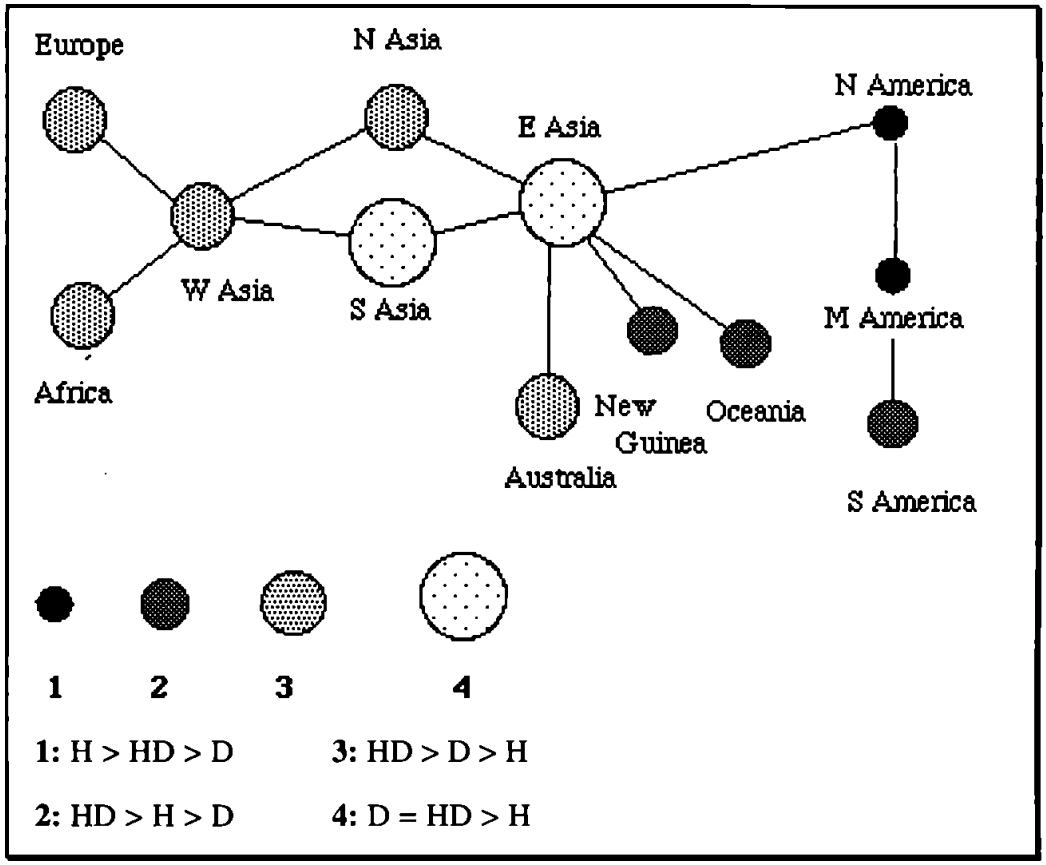


Figure 7. Areal distribution of head-marking and dependent-marking. H = Nr of languages with only head-marking in S; HD = Nr of languages with both head-marking and dependent-marking in S; D = Nr of languages with only dependent-marking in S. Based on Nichols (1992).

For each area in the appendix of Nichols (1992), I counted the number of languages with only head-marking in S (H), the number of languages with both head-marking and dependent-marking in S (HD) and the number of languages with only dependent-marking in S (D). As we can see, the result agrees with Nichols' general result: most dependent-marking in the Old World, less dependent-marking in the Pacific, and least dependent-marking in the Americas. Figure 8 shows the distribution of basic VO and VS orders.

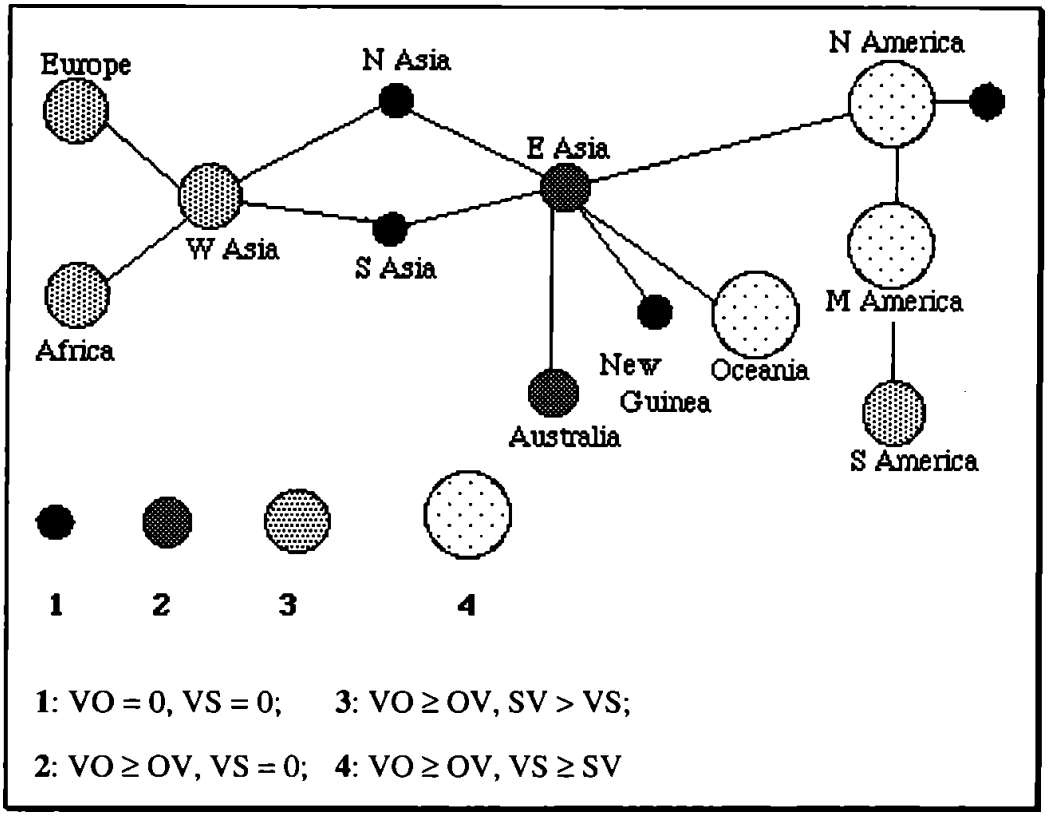


Figure 8. Areal distribution of basic word orders. OV, VO, SV, VS = Nr of languages with OV, VO, SV, and VS, respectively, as basic order. Note that the figures for West and East North America are very different. Based on Nichols (1992).

2.4. Initial states

The four combinations of head- and dependent-marking in figure 7 (which I designate as D1, D2, D3, and D4) relate to the cyclic stages of head/dependent-marking in (2) in the following way (since stage a does not appearin figure 7, it is designated as D0):

(6)	a.	- Agreement	- Case:		D0
	b.	+ Agreement	- Case:		D1: H > HD > D
	c.	+ Agreement	+ Case:	D2:	HD > H > D
				D3:	HD > D > H
	d.	- Agreement	+ Case:	D4:	D = HD > H

Any of these stages can of course be taken as the initial state of global linguistic development, but as far as I know only stage a and stage b have been seriously proposed. Most theories of grammaticalization at least implicate an initial state with only uninflected nouns and verbs, i.e. D0. A minority position is held by Jespersen (1922) and Swadesh (1971), whose suggested initial states are best described as radically head-marking languages, i.e. D1.

As for word order, Givón (1979) has suggested SOV as an initial state, and this suggestion can be taken to motivate a linear model with three stages: [-VO; -VS] - [+VO; -VS] - [+VO; +VS]. The model is linear because there seems to be no way leading from stage c back to stage a. The three stages correspond to the four distributions of VO and VS in figure 8 (which I designate as VO0, VO1, VO2, and VO3) in the following way:

(7)	a.	-VO	-VS:	VO0:	VO = 0, VS = 0
	b.	+VO	-VS:	VO1:	VO ≥ OV, VS = 0
	c.	+VO	+VS:	VO2:	VO ≥ OV, SV > VS
				VO3:	VO ≥ OV, VS ≥ SV

Since the model is linear, only VO0 can be an initial state.

This model is hardly the last word on word order change, though. The parameters are too simple, to begin with: neither OV and VO nor SV and VS are necessarily mutually exclusive. And there is no consensus on what is a possible word order change. Thus, the model in (7) should only be taken as an illustrative first approximation.

2.5. Transmission and change

In Indo-European, the changes from D2 to D4/D0 and from VO0 to VO2 seem to have taken around 10 000 years. If we generalize that pace, then the stages in (6) and (7), D0 – D4 and VO0 – VO3, respectively, would each last 5000 years, and the cycle in (6) would take 25 000 years.

With these figures, it is easy to derive predictions about the linguistic history of an area. Take Oceania, for example. Today, Oceania is in D2 and VO3. This means that Oceania would have been in D0 $(2-0) \cdot 5000 =$ 10 000 years ago and in VO0 $(3-0) \cdot 5000 = 15\,000$ years ago. However, since the process in (7) is cyclic, Oceania would also have been in D0 10 000 + 25 000 = 35 000 years ago, 10 000 + 25 000 + 25 000 = 60 000 years ago, and so on.

The general formula for deriving such predictions is given in (9). When a process is linear, $\text{Duration}_{\text{cycle}} = 0$, by stipulation.

(9) $\text{Stage j} = \text{Stage i} + (j - i) \cdot \text{Duration}_{\text{stage}} + n \cdot \text{Duration}_{\text{cycle}}$

The predictions computed for each area are given in table 2 below, together with its history.

Table 2. Temporal distance from initial states.

Area	**History**	**Temporal distance from D0 (thousand years)**	**Temporal distance from VO0 (thousand years)**
Africa:	IC(Africa) AD(Africa) AD(W Asia)	D3 = D0 + 15/40/65/90	VO2 = VO0 + 10
West Asia:	IC(Africa) AD(W Asia)	D3 = D0 + 15/40/65	VO2 = VO0 + 10
Europe:	IC(W Asia) AD(W Asia)	D3 = D0 + 15/40/65	VO2 = VO0 + 10
North Asia:	CD(W Asia)	D3 = D0 + 15/40/65	VO0 = VO0 + 0
South Asia:	AD(W Asia) EE(W Asia)	D4 = D0 + 20/45/70	VO0 = VO0 + 0
East Asia:	IC(W Asia) AD(E Asia) EE(W Asia)	D4 = D0 + 20/45/70	VO1 = VO0 + 5
Australia:	IC(E Asia)	D3 = D0 + 15/40/65	VO1 = VO0 + 10
New Guinea:	IC(E Asia)	D2 = D0 + 10/35/60	VO0 = VO0 + 0
Oceania:	AD(E Asia)	D2 = D0 + 10/35/60	VO3 = VO0 + 15
North America	IC(W Asia) CD(W Asia) AD(E Asia)	D1 = D0 + 5/30/55	VO0 = VO0 + 0 & VO3 = VO0 + 15
Mesoamerica:	IC(W Asia) AD(E Asia)	D1 = D0 + 5/30/55	VO3 = VO0 + 15
South America	IC(W Asia) AD(E Asia)	D2 = D0 + 10/35/60	VO2 = VO0 + 10

How are we to make sense of these figures? Let me just explore one of several possibilities. Suppose that a population split brings about a discontinuity in the transmission of a linguistic tradition, through which certain aspects of the tradition are lost to a language which 'walks away'. In the case of head/dependent-marking, what would be lost is inflectional morphology – a generalization of a well-known feature of the discontinuity in transmission associated with pidginization and creolization. If we try the hypothesis that this kind of discontinuity is primarily a consequence of initial colonization (including circumpolar or agricultural dispersal into a previously unpopulated area), a hypothesis which is consistent with Nichols' demonstration that head/dependent-marking shows a high degree of genetic stability, then we might, for example, use the data in table 3 to construct a possible scenario.

Table 3. Temporal distance from initial D0 in a scenario with initial colonization as trigger.

Area	**History**	**Temporal distance from D0 (thousand years)**
Africa:	IC(Africa)	90
West Asia:	IC(Africa)	65
Europe:	IC(W Asia)	40
North Asia:	CD(W Asia)	15
South Asia:	AD(W Asia)	20/45
East Asia:	IC(W Asia)	45
Australia:	IC(E Asia)	40
New Guinea:	IC(E Asia)	35
Oceania:	AD(E Asia)	10
North America	IC(W Asia)	30
Mesoamerica:	IC(W Asia)	30
South America	IC(W Asia)	35

The scenario that follows from table 3 is fairly realistic, if we match it against Renfrew's datings. The split between Africa and the rest of the world would have taken place in Africa 65 000 years ago, the split between North and South would have taken place in West Asia 45 000 years ago, and the splits leading to colonization of Australia, New Guinea, and the Americas would have taken place 40 000, 35 000, and 30 000 – 35 000 years ago, respectively. The date for circumpolar dispersal to North Asia, 15 000 years ago, is a little too early, but the discrepancy is not serious, given the extremely rough calculations on which the model rests. The only serious discrepancy in table 3 concerns South Asia. An agricultural dispersal 20 000 years ago is clearly an entirely unrealistic assumption. However, this discrepancy is easily corrected, if we assume that South Asian languages are the product of a continuous linguistic tradition that goes back to the split between North and South 45 000 years ago, that is, if we introduce IC(W Asia) into the history of South Asia.

What about word order, then? What would be lost here, I suggest, are constraints on word order. Thus, a discontinuity would make it possible to use a non-traditional order for various expressive and communicative purposes. However, this can only happen, I conjecture, when social control is weak, as it would be in agricultural dispersals, when expansion no longer takes place through intact bands of hunter-gatherers, but through a number of step-by-step migrations by smaller family units. In other words, it would take an agricultural, or comparable, dispersal to trigger off the development in (8). This conjecture is consistent with Nichols' demonstration that word order shows a low degree of genetic stability, but a high degree of areal stability.

Consider, against this background, a scenario that be constructed from the data in table 4.

Table 4. Temporal distance from initial VO0 in a scenario with agricultural dispersal as trigger.

Area	**History**	**Temporal distance from VO0 (thousand years)**
Africa:	AD(Africa) AD(W Asia)	10
West Asia:	AD(W Asia)	10
Europe:	AD(W Asia)	10
North Asia:		0
South Asia:	AD(W Asia) EE(W Asia)	0
East Asia:	AD(E Asia) EE(W Asia)	5
Australia:		10
New Guinea:		0
Oceania:	AD(E Asia)	15
North America	AD(E Asia)	15 + 0
Mesoamerica:	AD(E Asia)	15
South America	AD(E Asia)	10

Fairly compatible with the data in table 4 is a scenario where VO and VS orders result from two independent agricultural, or comparable, dispersals: one from East Asia, starting 15 000 years ago, and spreading to Oceania and the Americas; and one from West Asia, starting 10 000 years ago, and spreading to Africa, Europe, and South Asia. These postulated dispersals may be a little too early, but this can be corrected by adjusting $\text{Duration}_{\text{stage}}$.

There are three areas that do not fit this scenario at first blush. Word order changes in South and East Asia are too small to match the time depth of the dispersals postulated to affect these areas, and Australia shows word order change without a corresponding dispersal. However, both South Asia and East Asia have been subject to élite dominance expansions, and it is not very far-fetched to assume that these expansions brought along enough social control to arrest word order change in these areas. In Australia, there is evidence of a wide dispersal of one of the branches of Australian, Pama-Nyungan, and we may take this dispersal to be responsible for word order change in Australia. AD(Australia) should then be added to the history of Australia.

2.6. Summary

The areal histories in figure 4, complemented by IC(W Asia) in the history of South Asia and AD(Australia) in the history of Australia, the simplified parameters in figure 6, the historical processes in (6) and (7), the assumed initial states of these processes and the stipulated values of $Duration_{stage}$ and $Duration_{cycle}$, and the assumptions that transmission discontinuities with respect to head/dependent-marking are the results of initial colonizations, while transmission discontinuities with respect to word order are the results of agricultural (or comparable) dispersals, together produce the following scenario to account for the global distributions of head/dependent marking and word order in figures 7 and 8:

A split between Africa and the rest of the world took place in Africa 65 000 years ago, and a further split between North and South took place in West Asia 45 000 years ago. These splits were followed by splits leading to the colonization of Australia, New Guinea, and the Americas, which took place 40 000, 35 000, and 30 000 – 35 000 years ago, respectively. After that, circumpolar dispersal to North Asia, 15 000 years ago, was followed by two independent wide-ranging agricultural, or comparable, dispersals: one from East Asia, starting 15 000 years ago, and spreading to Oceania and the Americas; and one from West Asia, starting 10 000 years ago, and spreading to Africa, Europe, and South Asia. These dispersals were followed by élite dominance expansions into South Asia and East Asia, and were roughly contemporary with a wide-ranging dispersal of Pama-Nyungan in Australia.

This scenario might not be the 'right' one (it is, in fact, unlikely to be the right one, considering the number of corners I have cut), but it allows for a convenient summary of the main points of this paper: 1) It is possible to construct such scenarios from what we know about typology and change; and 2) To do this in an effective way, we should have access to a SimLing environment (figure 1) which produces such scenarios, in a graphically pleasing form, from revisable sets of model assumptions (figure 2), to account for global distributions based on large typological databases.

References

Cavalli-Sforza, L. L. 1991. *Genes, Peoples and Languages*, SCIENTIFIC AMERICAN, November 1991, pp. 72–78.

Dryer, M. S. 1989. *Large Linguistic Areas and Language Sampling*, STUDIES IN LANGUAGE 13, pp. 257–292.

Dryer, M. S. 1991. *SVO languages and the OV:VO typology*, JOURNAL OF LINGUISTICS 27, pp. 443–482.

Dryer, M. S. 1992. *The Greenbergian Word Order Correlations*, LANGUAGE 68,81–138.

Givón, T. 1979. *On Understanding Grammar*, Academic Press.

Hopper, P. J. and E. Closs Traugott. 1993. *Grammaticalization*, Cambridge University Press.

Jackendoff, Ray. 1990. *Semantic Structures*, MIT Press.

Jespersen, O. 1922. *Language. Its nature, development and origin*, Allen & Unwin.

Labov, W. 1972. *The Transformation of Experience in Narrative Syntax*, in *Language in the Inner City*, University of Pennsylvania Press, pp. 354–396.

Maddieson, I. 1991. *Testing the Universality of Phonological Generalizations with a Phonetically Specified Segment Database: Results and Limitations*, PHONETICA, pp. 193–206.

Nichols, J. 1992. *Linguistic Diversity in Space and Time*, University of Chicago Press.

Renfrew, C. 1992. *Archaeology, genetics and linguistic diversity*, MAN 27, 445–478.

Renfrew, C. 1994: *World Linguistic Diversity*, SCIENTIFIC AMERICAN, January 1994, pp. 104–110.

Ruhlen, M. 1987. *A Guide to the World's Languages*, Stanford: Stanford University Press.

Shibatani. M. 1990. *The Languages of Japan*, Cambridge University Press.

Silverstein, M. 1976. *Hierarchy of Features and Ergativity*, in R. W. Dixon (ed): *Grammatical Categories in Australian Languages*, Australian Institute of Aboriginal Studies, pp. 112–171.

Silverstein, M. 1987. *Cognitive Implications of a Referential Hierarchy*, in M. Hickmann (ed): *Social and Functional Approaches to Language and Thought*, Academic Press, 125–164.

Swadesh, M. 1971. *The Origin and Diversification of Language*, Aldine . Atherton.

Thomason, S. G. and T. Kaufman. 1988. *Language Contact, Creolization and Genetic Linguistics*, University of California Press.

System Architecture and Control in the Multra System

Björn Beskow
Uppsala

Abstract

This paper discuses the system architecture and control in the Multra system. The Multra system is briefly described, and its modular architecture is discussed. The control in the system is divided into global and module-internal control. In the inter-modular control, a *blackboard architecture* is introduced to control the interaction and synchronization of the modules. The blackboard architecture is also shown to enable parallel solutions. In the intra-modular control, the *specificity principle* is introduced. Its relation to subsumption is discussed, and the principle is shown to provide a declarative way to control interaction between linguistic rules. Finally, the preference formalism is presented, used to express preferences between analysis results.

Introduction

The MULTRA system

The MULTRA system is a prototype of a multilingual computer support for translation and writing, and has been developed within the project *Multilingual Support for Translation and Writing* at Uppsala University (see Sågvall Hein (1993)). One of the functionalities of the Multra system is machine translation. The user, working in an interactive document processing environment can mark a region of the document and have it translated on the fly. The region can, in principle, range from a single word to the whole document.

The Multra machine translation component is transfer-based. Translation is performed on a sentence level, but exploiting the type information provided by the document representation format. The translation is performed by four independent modules (see figure 1), responsible for analysis of the source language, preference ordering of the analysis results, transfer, and synthesis of the target language. An attribute-value logic is the common representation formalism for all the modules.

The different modules are implemented as separate Unix processes, communicating through TCP/IP sockets. The processes can thus execute on different machines or on different processors on the same machine.

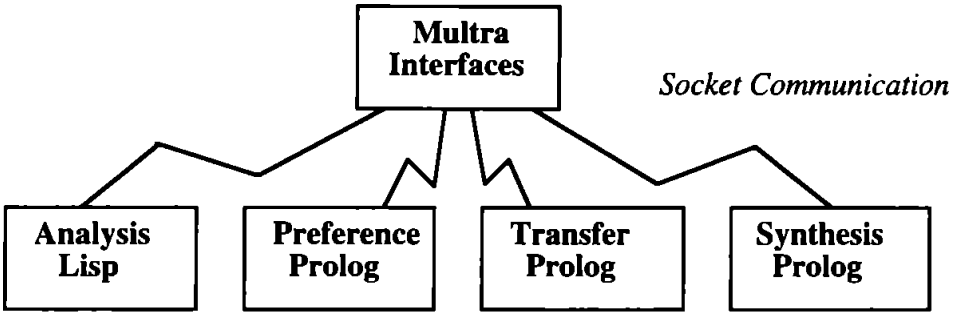


FIG. 1 : Multra System Architecture

Logic and Control

The four main modules implement the global logic of the system. A set of rules implements the module internal logic for each module.[1] The task of *control* is to specify the interaction between the parts, both on a global and a module-internal level. The logic of the Multra system has been described elsewhere.[2] In the following sections, we will discuss the intra-modular and inter-modular control in the Multra system.

Inter-Modular Control

As we have seen, the modules in the Multra system are fully autonomous. However, the result of a module may form input to another module, resulting in a sequential information flow through the system. Because of the modules being fully autonomous, they may however very well execute in parallel. For instance, the parser starts by parsing the first segment. When ready, the parser output constitute input to the preference machine. The parser may however start parsing the next segment without having to wait for the other modules to process the first segment. The same holds for all modules.

The inter-modular control must therefore enable the sequential flow of information through the system by providing a communication channel between the modules, and by synchronizing the work of the modules. This control is achieved by using a *Blackboard*.[3] The blackboard is a common data area accessible by all modules.

---

[1]The set of rules within a module is conceptually divided into general and domain specific rules.

[2]See e.g. Beskow (1992; 1993a; 1993b) and Sågvall Hein (1987; 1993a; 1993b).

[3]Blackboard systems have mainly been used to provide data-driven processing, to integrate information from many different sources and to have several competing threads working on the same problem. A blackboard architecture is however also very suitable for controlling interactions between modules, to synchronize modules and to exploit parallellism. See Linda (1988) for a discussion of Blackboard systems.

In the Multra system, each module can read or write first order terms on the blackboard. A module is triggered by its own special term providing its input. When the module has completed its processing, it writes its own special result term back on the blackboard and then waits for the next input term. This situation is illustrated in figure 2 below.

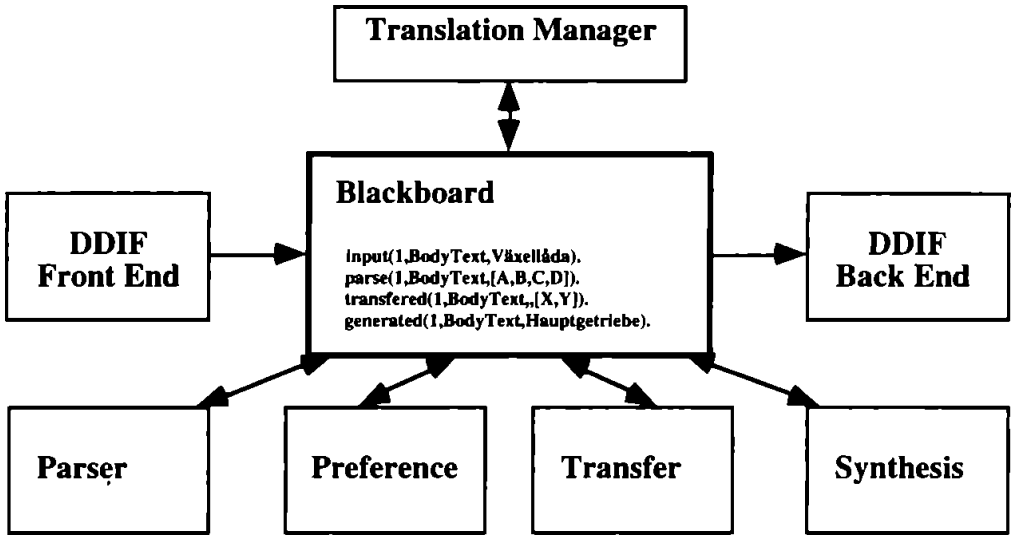


FIG. 2 : Blackboard

The parser reads terms of the form *input(N,Type,String)*, and writes terms of the form *parsed(N,Type,ParseSet)*. The preference machine reads terms of the form *parsed(N,Type,ParseSet)* and writes terms of the form *preferred(N,Type,ParseList)*. Each term has as its first argument the segment number of the processed segment. This enables a sequential flow of information through the system, in spite of the parallel nature or the processing. The synchronization of the modules is automatically achieved through the Blackboard system.

Since the modules are fully autonomous, only communicating via the blackboard, multiple instances of a module may very well exist and execute in parallel. Hence it is possible to use several instances of a module to perform a computationally heavy task. This situation is illustrated in figure 3.

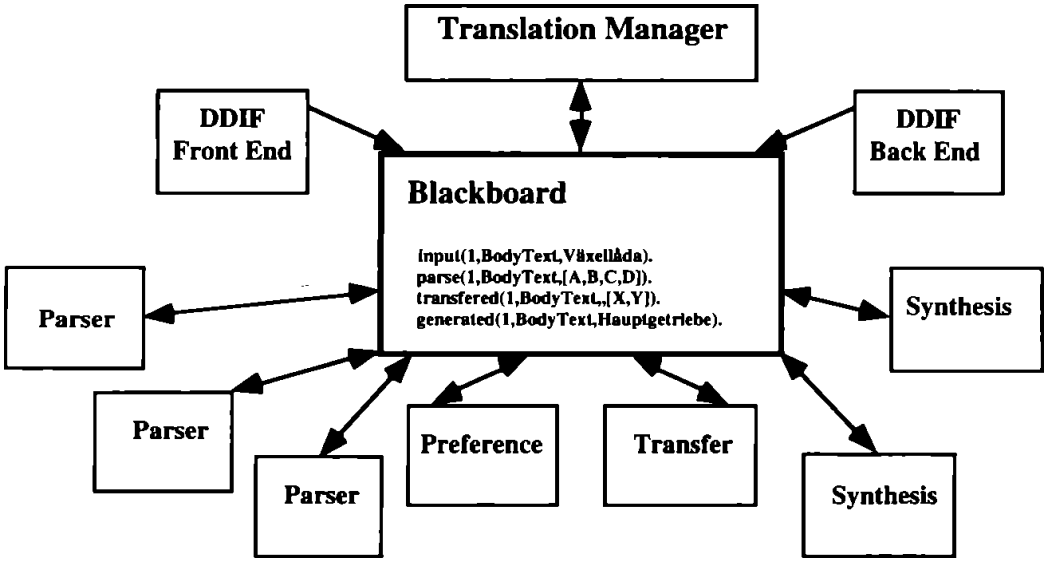


FIG. 3 : Multiple instances of modules

In fact, it is even possible to dynamically assign resources to tasks, within a *Processor Farm* model. The global controller can act as a 'farmer', having a number of processors or 'workers' under his command. The relative number of pending input terms on the blackboard for a certain module constitutes a work load measurement, measuring how heavy a certain task is. The farmer may dynamically assign more workers to a heavy task to maximize the efficiency. Labour division orders are just special control terms, written on the blackboard by the farmer.

Intra-Modular Control

Now we shall turn our attention to the module-internal control in the Multra system. The key concept here is the notion of *specificity*. The general idea is that more specific solutions should block or precede other more general solutions. A more specific translation should be preferred before a more general translation.

In terms of attribute-value logic, the subsumption relation forms a partial information ordering on attribute-value structures. Since the rule formalisms in all the modules are based on attribute-value logic, subsumption can be used to define specificity orders on rule sets. In a logical framework, the specificity principle may then be defined in terms of specificity between rules: Prefer a (constructive) translation proof based on more specific rules before a translation proof based on more general rules.

Let us look at an example. A transfer rule in the Multra formalism consists of two feature structures describing the source and target structures, and a (possibly empty) set of recursive transfer equations on

subparts of the source and target structures.[1] Consider the two transfer rules in figures 4 and 5 below, describing transfer relations between Swedish and German:

```
Label NOUN.OBJ
Source
 <* NOUN.OBJ> = ?NOUN.OBJ1
Target
 <* NOUN.OBJ> = ?NOUN.OBJ2
Transfer
 ?NOUN.OBJ1 <=> ?NOUN.OBJ2
```

FIG. 4 : Transfer rule 1

```
Label NOUN.OBJ_PP-NP
Source
 <* NOUN.OBJ PHR.CAT> = PP
 <* NOUN.OBJ PREP LEX> = AV1.PP.4
 <* NOUN.OBJ RECT> = ?RECT1
Target
 <* NOUN.OBJ> = ?NOUN.OBJ2
 <* NOUN.OBJ CASE> = GENITIVE
Transfer
 ?RECT1 <=> ?NOUN.OBJ2
```

FIG. 5 : Transfer rule 2

Rule 1 is a general rule, saying that in general, a noun object should be translated compositionally. Rule 2 is more specific, saying that a noun object that is a preposition phrase with the Swedish preposition 'av' should be translated into a genitive construction in German. The source attribute-value structure of rule 1 subsumes the source attribute-value structure of rule 2, hence rule 2 is considered more specific than rule 1. A translation based on rule 2 should be preferred before a translation based on rule 1.

[1] See Beskow (1993a) for a description of the Multra transfer formalism. Examples of complex transfer relations described in the formalism can be found in Sågvall Hein (1993b) and in Wikholm (1992).

The Preference Machine

The preference machine in the Multra system takes as input a set of attribute-value structures. This set represents the different analyses for a sentence produced by the parser.[1] If the set contains more than one element, the sentence is ambiguous. The task of the preference machinery is to compute a preference ordering on this set, returning a list with the attribute-value structures partially sorted.

The Multra preferences are defined by a set of preference rules. A preference rule defines a binary preference relation between two attribute-value structures. The set of preference rules thus defines a weak order on the set of attribute-value structures.[2] A preference rule consists of two attribute-value structures *Minor* and *Major*, representing the preferred and the dispreferred analysis result.

Figure 6 below is a simple example of a preference rule. It defines the preference relation between two attribute-value structures having different values for the attribute-value 'NUMB'. It says that the structure with value 'SING' is preferred before the structure with value 'NUMB'.

```
Preference SING-PLUR
     <* NUMB> = SING
precedes
     <* NUMB> = PLUR
```

FIG. 6 : Preference rule 1

Figure 7 below is an example of a preference rule that defines the preference relation between two attribute-value structures both having the value NP for the attribute-value 'PHR.CAT', but only the first one has the attribute-value 'POST.ATTR' defined.

```
Preference POST.ATTR
     <* PHR.CAT> = NP
     <* POST.ATTR> = ANY
precedes
     <* PHR.CAT> = NP
```

FIG. 7 : Preference rule 2

---

[1]For a description of the Multra parser, see Sågvall Hein (1987).

[2]The preference order is indeed a partial order of equivalence classes of feature structures, which correspond to a weak order (see e.g. Berge (1962) or Ore (1963)).

As all rules in the Multra system, the preference rules are themselves ordered by the specificity principle. A preference rule r is more specific than another rule r', written r $\leq_{prefrule}$ r', if and only if Minor(r') subsumes Minor(r).

The preference relation has the following semantics:

Let ϕ and ψ be two attribute-value structures, P a set of preference rules, and '$\leq_{pref}$' the preference ordering symbol. ϕ is preferred before ψ, written $\phi \leq_{pref} \psi$, if and only if

* there exists a preference rule r such that
  - Minor(r) subsumes ϕ and
  - Major(r) subsumes ψ and
  - for all r':
    if Major(r') subsumes ϕ and Minor(r') subsumes ϕ then
    r $\leq_{prefrule}$ r',

  or
* there exists a path p in both ϕ and ψ such that
  - $\delta(p,\phi) = \phi'$ and
  - $\delta(p,\psi) = \psi'$ and
  - $\phi' \leq_{pref} \psi'$

Consider the set of attribute-value structures in figure 8 below:

$$\left\{ a = \left[\begin{array}{l} CAT:NP \\ NUM:SING \\ DEF:INDEF \\ HEAD:\left[\begin{array}{l} LEX:VÄXELLÅDSHUS \\ WORD:CAT:NOUN \end{array} \right] \end{array} \right], \; b = \left[\begin{array}{l} CAT:NP \\ NUM:PLUR \\ DEF:INDEF \\ HEAD:\left[\begin{array}{l} LEX:VÄXELLÅDSHUS \\ WORD:CAT:NOUN \end{array} \right] \end{array} \right] \right\}$$

FIG. 8 : Singular/plural ambiguity

The two attribute-value structures represent the two possible readings for the Swedish noun 'växellådshus': one singular and one plural reading. If we take as our set of preference rules to be the rules in figure 6 and 7, we can see how they define a weak order on the attribute-value structures in figure 8. We can see that a $\leq_{pref}$ b holds, because of rule 1 whose Minor structure subsumes a, and whose Major structure subsumes b.

Digression: Non-existence of attribute-values

The preference formalism presented above has an interesting property: it allows for implicit non-existence conditions of attribute-values. Identity equation constraints used for describing attribute-value structures can only express positive constraints on the attribute-value structure being described. It is not possible in an identity equation to say that a certain attribute-value must not be defined, or must not have a certain value. It has been much discussed whether negative values are necessary to gain enough expressive power.[1]

The Multra preference formalism allows for an implicit way of stating negative attribute-value conditions. We have already seen a rule (in figure 7 above) which exploits this property. Consider the general reformulation of such a rule below:

```
Preference Non-existence
    <* F> = ANY
precedes
    <*> = ANY
```

FIG. 9 : Non-existence condition rule example

Consider further the two attribute-value structure pairs below:

$$
\begin{array}{lcl}
a & = & [F{:}A] \\
b & = & [\,] \\
a' & = & [F{:}A] \\
b' & = & [F{:}B]
\end{array}
$$

FIG. 10 : Non-existence condition structures example

We can see that a $\leq_{pref}$ b must hold because of the preference rule in figure 9. We can also see that a' $\leq_{pref}$ b' holds, by virtue of the same rule. However, we also find that b' $\leq_{pref}$ a' using the same rule. Since a partial order is asymmetric, it follows that a' $\equiv$ b', that is, they belong to the same equivalence class according to the preference ordering.

The example above shows how a preference rule implicitly can express a negative attribute-value condition. The rule in figure 9 above says that a attribute-value structure with the attribute F defined precedes a structure that does not have the attribute F defined.

[1] See for example Eisele & Dörre (1988).

Summary and conclusions

In this paper, the system architecture and control in the Multra system have been discussed. We have seen how a strictly modular system design enables the exploitation of parallelism. A blackboard architecture may be used to control both the global interaction between modules and the synchronization of parallel threads. The module-internal control has also been discussed. The notion of specificity has been introduced, and its relation to subsumption shown. The preference machine has also been presented. We have seen that the intra-modular control in multra is based on declarative notions and formalisms within the unification-based paradigm. The control defines and computes partial orders on attribute-value structures and on rule sets. I hope to have shown that the control mechanism of the Multra system provides a both elegant and efficient way of handling interaction on different levels.

References

Berge, C. 1962. *The Theory of Graphs and its Applications*. Methuen & Co Ltd.

Beskow, B. 1992. *Unifieringsbaserad Transfer*. Masters Thesis, Gothenburg University.

Beskow, B. 1993a. *Unification-Based Transfer. Multilingual Support for Translation and writing*. Forthcoming. Uppsala University.

Beskow, B. 1993b. *Generation in the Multra System*. Uppsala University.

Eisele, A. and J. Dörre. 1988. *Unification of Disjunctive Attribute-value Descriptions*. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*.

Ore, O. 1963. *Graphs and their uses*. Random House.

Shieber, S. 1986. *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes.

Sågvall Hein, A. 1987. *Parsing by Means of Uppsala Chart Processor*. In Bolc, L. (ed.), *Natural Language Parsing Systems*. Berlin.

Sågvall Hein, A. 1993a. *Multilingual Support for translation and Writing. MULTRA*. Project report, NUTEK/HSFR Language Technology Program.

Sågvall Hein, A. 1993b. *On The Translation of Nominal Expressions in a Multilingual Unification Based Setting*. In Hajicova, E. (ed.) *Proceedings of the Functional Description of Language*. Prague 1993. pp. 209-224.

Wikholm, E. 1992. *Schwedish-deutsche lexikalische transferregeln. Nicht-flektierbare funktionale Phrasen*. Project report. Uppsala University.

Automatic Tagging Of Turns in the London-Lund Corpus with Respect to Type of Turn

Benny Brodda
Stockholm

0. Abstract.

In this paper a fully automatic tagging system for the dialogue texts in the London-Lund corpus, LLC, will be presented. The units that receive tags are "turns"; a collection of (not necessarily connected) tone units – the basic record in the corpus – that one speaker produces while being either the "floor holder" or the "listener"; the quoted concepts are defined below. The tags constitute a classification of each turn according to "type of turn". A little sample of tagged text appears in Appendix 1, and is commented on in the text. The texts to be tagged will in the end comprise all the texts in the three subcorpora of LLC appearing in Svartvik & Quirk, "A Corpus of English Conversation", (=CEC); so far, about half of these texts have been tagged, now with the programs working properly, the rest will hopefully be tagged before the end of this year.

1. Introduction

An outline of the classification scheme underlying the present tagging system was presented in Brodda, 1988, and is essentially the same classification system used in this report. In the present project, however, the classification is made explicit through the tags, simplifying the verification problem considerably.

The tagged texts will provide a basis for a statistical investigation of the corpus; one important question that will be adressed is whether or not speakers tend to differ in the factors these tags reflect when the speakers sex, social rank, or other properties that CEC provides about the participants of each dialogue text are taken into account. Britt Erman, Stockholm, will present a linguistic investigation of such factors in the same corpus. By the end of this year, we hope to have the statstical evaluation completed.

The underlying factors the tags reflect are probably to some degree semantic, sociolinguistic and context dependent, but primarily they show considerable individual variability related to the participants speaking habits, to their mental or physical mood at the recording occasion, the topic that happens to be discussed and so on. This means that one will have to take a considerable number of texts into account in order to filter out such individual variations, while hopefully retaining some significant

residual. A sample of the of basic frequencies that will go into the statistical machinery is presented in appendix 3.

The corpus itself, as well as the programs involved in the project are adapted to an ordinary (IBM compatible) PC-environment. Once the corpus is modified as described in section 2 below, the final tagging of each text will take about one minute on a 286-PC (16 Mz) and about the same time for frequency counts.

2. Corpus preparation.

In order to get the tagging and statistical programs working properly, a substantial simplification and standardisation of the corpus itself has been carried out. Elsewhere I will present a critical and thorough analysis (Brodda, 1994) of the LLC corpus and its technical design as it has usually been distributed to research groups around the world. That report will also contain a full account of the general purpose modifications made for the present project. It is quite clear that one can simplify the texts considerably, without loss of any information whatsoever, and at the same time better suit them for automatic analysis by computer. The fact that the revised corpus requires less than half the disk space of the original text (still in pure ASCII) is probably good news as well, especially when working on a PC where diskspace is not always an unlimited resource.

2.1 The basic modification of the corpus.

The basic record of the corpus is still the T(one) U(nit), but it has now a more BROWN-corpus style structure:

```
Text-id TU-id Speaker-id t....e....x....t#
```

where the line headers here are of length 14 ("line header" = a fixed length, initial portion of each line not containing the text itself). The tone unit delimiters, "#", are moved to the actual ends of complete TUs, meaning they become formal end markers of complete basic units. A "~" is inserted as a corresponding end marker of each incomplete TU.

The texts are then sorted in ascending order with respect to the TU-id(entifier)s in such a way that the lines constituting one TU appear in the sorted text in the same relative order as in the original text; a "+" is prefixed the Speaker-id to indicate that the TU continues on the next line. In this sorting it is chunks of simultaneous speech that are shuffled around, but simultaneous speech represents nonlinear events, anyhow, so the sorted text is absolutely equivalent to the original text; cf. p. 6 in the

foreword of CEC. The sorting makes the text considerably less fragmented.

For the specific investigation presented here we did not need the prosodic markers, so we simply rinsed the text of these (which saves another 8% disk space). If this rinsing is done carefully, every word form can be rendered a "stable" spelling, which simplifies any type of parsing of the corpus (a simple parse is employed in the present project). Later we will try to see if the classification can be refined, when prosodic markers are taken into account, or if the tags correlate with these prosodic markers in one way or other.

The mentioned modifications of the corpus are all completely done by computer. We have also made a few (semi)manual modifications in order to standardise the texts further still; this standardisation is "general purpose" (not tied to this specific project), and should simplify any type of automatic analysis of the corpus; cf. Brodda, 1994.

3. A brief description of the turn classifying algorithm.

3.1 Turns and Formal Turns.

Let us start with a little exposé of things familiar to everyone and included in order to pinpoint a few phenomena that my programs identify.

Usually one "turn" in a dialogue is conceived as a stretch of speech that one participant utters in a connected sequence of words, phrases, tone units, or whatever elements speech is assumed to be made up of. In well disciplined dialogues each participant is allowed to deliver his/her turns uninterrupted; when a participant finishes his/her turn, another "takes the floor", delivers his/her turn, and the dialogue procedes in an orderly fashion. These kinds of turns I call "regular turns", and the switching between them I call "regular turn taking".

In more informal dialogues people are not that well behaved. Participants laugh, start talking when someone else already has the floor, and so on. Sometimes these are simply side comments to what the floor holder is saying, at other times the new speaker brutally takes over the floor (we have a "takeover" situation), perhaps accompanied by an increase in voice volume. Sometimes the takeover fails (perhaps the floor holder raises his/her voice still more and manages to maintain the floor); such a situation I call an "attempt".

Even when participants are disciplined and await their turns in an orderly fashion, they are not always silent, at least not in less formal situations. They deliver typical feedback signals: "yeah", "of course", "yes", "certainly", and they laugh etc., sometimes while the floor holder is actually speaking (in which case I call such signals "back channels"), or when the floor holder briefly pauses (in which case I call them "feedbacks"). Such feedback signals do not break the floor holders turn, and indeed are not meant to. In most everyday dialogue situations, such interaction is, in fact, quite necessary – in telephone dialogues it is mandatory – and has a purely supportive function. In more formal situations, such as seminars and the like, head knoddings, smiles and so on, have this same function.

The auxiliary "formal turn" concept below, is a first approximation of a more final turn concept; I will return to this later. The formal turns will be the object for the tagging algorithm.

The **formal turn** concept:
A formal turn (**FT**) is the collection of all TUs in a maximal, unbroken sequence of TUs assigned to one and the same speaker; "maximal" in the sense that the FT cannot be further extended and still be an FT (i.e. the FT in question is surrounded on both sides by either some other speaker's FTs or text end-markers). The term "speaker" here means the string constituting the Speaker-id field; thus speakers "A" and "AB" ("A" or "B") are distinct. Cf. section 3.3 about speaker ",".

In the text samples below, formal turns are identified by the FT end marker, "|".

3.2 The tagging scheme

Table 1 below, summarises the tagging scheme. Each formal turn receives a turn type tag, viz. any of the characters in the set TC of tagging characters:

```
TC = {r, c, t, a, l, m, f, b, u, ","};
```

The classification algorithm runs in two passes. In the first pass the FTs are classified "context free". In this pass all types of turns except the "c"-turns are provisionally recognised. In the second pass, which employs a kind of context sensitive rules, some of the tags from the first pass are changed in one way or another; primarily the "c"-tags that are now introduced.

Table 1. The Turn Type Classification Scheme

```
S represents the speaker,"|" is the formal turn endmarker.

regular turn:       S     r     bla bla *bla* bla|
continuation:       S     c     bla bla bla
takeover:           S     t     *bla bla* bla bla|
attempt:            S     a     *bla bla*|
feedback signal:    S     f     yes|
backchannel:        S     b     *yes*|
laugh:              S     l     (laughs)|
back ch. laugh:     S     m     *(laughs)*|
human noise:        S     u     (cough)| or *(cough)*|
external noise:     ,     ,     (bang)| or *(bang)*|

"attempt" is short for "floor stealing attempt";
"take over" is short for "brute force floor take over";
"*" denotes a "break character"; cf. below.
```

3.3 The Context Free pass.

The first pass recognises explicitly the l, m, f, b, u and "," turns according to what the FTs contain as indicated in Table 1. Thus an FT receives the tag "f" (= feedback) if it contains a mere "yes" or any other more or less synonymous word according to a little lexicon containing some 15 odd elements: ("yeah, mm, quite,.."), and if it is not enclosed in break characters (cf. immediately below); even FTs containing a combination or repetition of these elements receive this tag:

```
"A f oh yes yes yes|" (A is the speaker)
```

The same FT will receive the tag "b" if it is enclosed in a pair of "break characters", any of the characters or character combinations "*", "+", "**" or "++". Thus the following is a typical b-turn:

```
"A b *oh yes yes yes*|"
```

Break characters come in quadruples. A pair, like the one above, indicates that A's utterance is produced while someone else is talking. Immediately above or below this b-turn there should occur another FT containing a stretch of speech enclosed in the same pair of break characters, indicating that the matching stretches of speech occur simultaneously. (Cf., e.g., TUs 38 and 39 in the text sample in Appendix 1.)

The l, m, u and "," turns are likewise recognised through lexical lookups; thus, turns receive the "l" or "m" tags if the turns solely contain strings like "(laughs)", "(giggles)" or a few variants of these. The ","-turns are those FTs that appear in the original corpus without a speaker-id(entifier), typically indicating an external noise of some kind, such as "(phone rings)", "(car noise)", etc. (In the modified corpus the comma is

also used as speaker-id for such TUs, meaning that each TU in the corpus formally has an owner.) The FTs may also contain various combinations of the elements mentioned above, and then they receive a tag according to a kind of heuristic rules. Thus, an FT of the type "A (laugh) yeahl" gets the "l"-tag, the FT "A yeah (laugh)l" the "f"-tag.

Every FT not explicitly recognised in this first pass is considered to contain "real" – more substantial – speech ("bla bla" in Table. 1). Thus, real speech is negatively defined. FTs containing real speech receive any of the "a", "t" or "r" tags depending on whether the FT contains simultaneous speech in a dominant way or not. If the FT is completely enclosed in break characters, it receives the "a" tag, if it only has an initial part enclosed in such characters, it receives the "t"-tag, otherwise it receives the "r"-tag.

3.4 The context sensitive pass

In the second pass the following explicit assumption is built into the program:

The **floor holder** concept:
At any moment in time (at any place in the text actually, from the program's point of view) there is always one dialogue participant that is established as the **floor holder**, **FH**. There are exactly two ways the FH may shift, viz. through what I call **significant** turn taking events.

The program assumes an "unspecified" speaker – distinct from all actual participants – as holding the floor when a text begins.

3.4.1 Significant turn taking events.

One way the FH may shift is through **regular turn taking**: A speaker, other than the established FH, enters and delivers an FT that has been classified as an "r"-turn in the first pass. The owner of this new "r"-turn then becomes the new FH and the FT retains its "r"-tag.

Another way the floor holder may shift is through a **takeover**. This situation – a typical example of which appears in the text sample in Appendix 1 at TU 30 – occurs when the established FH's latest FT ends in a stretch of simultaneous speech that overlaps with simultaneous speech in the beginning of a new speakers FT, which has been given the "t"-tag in the first pass; these stretches of simultaneous speech must also contain "real speech". The actual floor holder shift takes place – the program assumes – precisely at the point where the inital stretch of simultaneous

speech ends in the new speakers FT. In a takeover situation, this FT retains the "t"-tag from the first pass.

Both "t"-tags and "a"-tags may sometimes be changed into "r"-tags. This happens when i. the owner of the corresponding FT differs from the established FH, and ii. the prominent stretch of simultaneous speech in this FT matches a stretch of speech that is of a "weaker" category than the present in a turn type strength hierarchy, TSH, implicitly reflected in Table 1 but more formally defined as:

```
TSH: r > t > a > f > b > l > m > u > ",";
```

where the symbol ">" (here) stands for the two-place predicate "is stronger than". For present purposes, only the order between "t", "a" and the weaker ones is of interest. Thus, if the prominent stretch of simultaneous speech in a "t"-tagged FT matches that of an "a"-tagged or weaker, then the "t"-tag is turned into an "r"-tag; an "a"-tag is similarily turned into an "r"-tag, if it matches an "f"-turn or weaker.

The full hierarchy is needed for describing certain details of the statistic calculations.

3.4.2 Continuations

A typical episode in a dialogue starts with a sequence of "r"-turns, i.e. the floor holder shifts regularly from one speaker to the other. If any of the turns of the a, f, l, b, m, u or "," types are encountered, the floor holder normally does not shift, and let us assume now that he does not.

After such an interlude, two things may happen. Either the floor holder reappears in the FT immediatly following such an interlude, or a third participant appears (remember, a shift in FT always implies a shift in speakers). In the first case this new FT receives the "c"-tag regardless of what tag it received in the first pass, and it is assumed to be a continuation of the same speaker's former turn. If another speaker appears immediately after the interlude, then this new FT is treated as any other new FT as described above, i.e. the new speaker may become the new floor holder or the corresponding FT is just another interlude.

3.5 Turns

The second pass is considerably more complex than I have indicated here. Among other things, certain FTs are broken up into sub-FTs as indicated through a "\" in the text samples below. Many more details could be

commented upon, but I think we are ready to define a final "turn" concept.

Major turns:
A **major turn** is a collection of FTs assigned to one and the same speaker, beginning with a significant turn taking event and interrupted only by such FTs that do not imply a shift in floor holder. Thus, a major turn always begins with either an "r" or a "t"-tag, i.e. when the speaker enters the floor, and zero or more "c"-tagged FTs that are continuations of the same turn. The whole turn is called a "regular" turn or a "takeover" depending on the tag on the initial FT.

Minor turns:
A **minor turn** is a formal turn that has any of the tags in the subset {a, f, b, l, m, u, ","} of TC after the second pass.

At any moment in time, the established FH is the speaker (at that moment) and the other participants are the (temporary) listeners.

4. Illustrations

What is described in section 3 above is of course a computer model intended to capture certain aspects of turn taking in the LLC-texts (or in any informal dialogue), and as any such model it captures reality more or less good. The evaluation so far, indicates very good corrolation between how the computer classifies turns in the LLC corpus and how students at the English department at Stockholm University do it. There is not enough space here to present larger samples of tagged text, but the samples given in Appendixes 1 and 2 would at least give an indication of what the tagging looks like.

The mentioned text sample illustrates a typical episode in a longer dialogue. After B's initial "r"-turn, speaker A starts an "r"-turn at TU 26 but encounters a prototypic takeover by B (the shift from TU 29 to 30). B manages then to keep the floor all the way down to TU 48. Thus, B's "t"-turn consists of the FTs (identifying each FT through its initial TU-id) 30, 34, 38, 40, 43 and 47.

Note, this takeover is also a semantic takeover. When the episode begins they are involved in a discussion about A's years as a young student, a topic that A continues to evolve in FT 26. B, however, breaks in and starts talking about her own years as a young student. (Both speakers are female).

In Appendix 2 a few special cases are given. Ill. 2.1 illustrates an interesting error of principle. FT/TU 1042, which consists of a single "yes#", is – precisely according to the algorithm – given the "f"-tag, i.e. classified as a feedback signal. If one scrutinises the context more closely, it appears, however, that this "yes" is an affirmative answer to a straightforward yes/no-question. According to any linguistic criteria it must, of course, be considered as a substantial turn; it adds semantic material to the dialogue and should be given the "r"-tag.

The text L1-5 contains about 115 FTs consisting of a single "yes", "yeah" or "yea" (enclosed in break characters or not). As far as one can deduce from the text, every one of them except the mentioned TU 1042 are feed back signals and not substantial turns (and consequently correctly tagged by the program). Text L1-5 is quite representative for the informal dialogues in the LLC-corpus, and the investigation so far seems to indicate that only about 1% of all single "yes"es produced in such dialogues represent substantial turns.

What about "no"? FTs consisting of a single "no" are, of course, considerably fewer than those consisting of a single "yes". Contrary to what one may think, though, also "no"-turns tend to be feedback signals more often than substantial turns, and they are regularly so when produced in some negative context, in which case they indicate that the no-sayer agrees with what is just said; strictly speaking "no" then means "yes" (cf. FT 1187 – Ill. 2.2 – where speaker A says both "no" and "yes"). Such a "no"-turn I call an "affirmative no". The tagging program assigns a "b"- or "f"-tag to a "no"-turn, if the preceding FT simply contains the word "not" or the word end "-n't" regardless of context, and so far this simple surface criterion has never gone wrong.

Appendix 3 contains the type of frequency tables that will underlie the statistical evaluation. The tables describe speaker A's "event history" during the whole dialogue l1-5. The first two tables show what speaker A does, how many turns of different types she produces, and the total no. of words produced during each turn type.

The last two tables describe the kinds of "attacks" speaker A encounters while holding the floor, i.e. the types of simultaneous speech other participants produce while A is the FH, and the number of words A produces during these attacks. (Some of these "attacks" are certainly not real attacks, since the feedback types are "supportive" rather than "hostile").

In the same way we obtain corresponding figures for every single speaker in any of the texts in the corpus, which figures then are inserted into a database, together with information about the speaker's sex, the

number of participants in the corresponding dialogue and the other participants' sex. This database will provide the basis for a statistical investigation of the corpus.

References

Brodda, B. 1988. *Tracing Turns in the London-Lund Corpus with BetaText.* JLLC, Vol. 3, No.2.

Brodda, B. 1994. *Simplifications and Modifications of the LLC-corpus in Preparation for Automatic Analysis Internal report*, Department of Linguistics, Stocholm University, S-10691, Stockholm, Sweden.

Erman, B. 1994. *Computer Analysis of Female and Male Conversational Strategies in same-sex and mixed-sex interaction in LLC*, Paper sent to the ALLC-ACH94 conference.

Svartvik, J and R. Quirk. 1980. *A Corpus of English Conversation*, London Studies in English, Lund.

Appendices: Illustrations

Appendix I: Sample tagged text (from L1-5T.TXT):

```
1 5 25+B    r   I don't suppose you need Old English and
1 5 25 B        Anglo-Saxon#|
1 5 26 A    r   well no .#
1 5 27 A        but [@m] you know#
1 5 28 A        *I don't#
1 5 29 A        have any language*#|
1 5 30+B    t   *[@m] well I <hadn't>* done any English at
1 5 30 B        **all**#
1 5 31 B        you know#
1 5 32 B        since O-level .#|
1 5 33 A    f   **<1 syll>** yea .#|
1 5 34 B    c   and I went to some second year {seminars}#
1 5 35+B        where there are only about half a dozen
1 5 35 B        people#
1 5 36+B        *and* they discussed what <a>
1 5 36 B        word was#|
1 5 37 A    b   *[m]*#|
1 5 38 B    c   **and -** what's a sentence#|
1 5 39 A    b   **[m]**#|
1 5 40 B    c   that's *even* more difficult .#|
1 5 41 A    b   *yeah*#\
1 5 42 A    f   yeah -#|
1 5 43 B    c   and so on .#
1 5 44+B        and then I also went to some postgraduate
1 5 44 B        ones#
1 5 45 B        which were more interesting -#|
1 5 46 A    f   yea#|
1 5 47 B    c   which he had for [dhi] - diploma -#
1 5 48 B        the main people#|
1 5 49 A    r   on -#|
1 5 50+B    r   and I suppose they're doing the same
                ones this
1 5 50 B        year#
1 5 51+B        and then you'd have a whole evening
                {battling
1 5 51 B        on} - - -#|
1 5 52-?    r   <4 to 5 sylls> - -~|
1 5 53 B    r   no#
1 5 54 B        sessions .#
1 5 55 B        several sessions#
1 5 56 B        on *nominal* groups or something#
1 5 57+B        <then> you can
1 5 57 B        pick up all the jargon#|
1 5 58 A    b   *[m]*#|
1 5 59-B    c   and~|
1 5 60 A    f   yea - -#|
1 5 61+B    c   and then sort of get the hang of
                what they're
1 5 61 B        talking about -#
1 5 62+B        I should ask him {if there are any
                seminars you
1 5 62 B        ought to go to}#|
1 5 63 A    f   yea -#|
```

Appendix 2:

==========

ILL. 2.1: An "f" that is a real turn:

```
--------------
1 5 1037 C    c   I mean I've worked in universities#
1 5 1038 C        for nearly ten years now#|
1 5 1039 A    f   yeah .#|
1 5 1040 C    c   *and*#|
1 5 1041 A    t   *are* you going to America#|
1 5 1042 C    f   yes#|                          <------
1 5 1043 A    c   [m]#
1 5 1044 A        I [z] . tried to go to America#
1 5 1045 A        earlier this year#
1 5 1046 A        *and* then decided <syll syll>#|
1 5 1047 C    b   *[mhm]*#|
```

ILL. 2.2: Examples of "affirmative no"

```
--------------
1 5 779+C     c   [@] - - but [@] they're just sort
                  of pursuing
1 5 779 C         their own research#|
1 5 780 A     f   yea#|
1 5 781 C     c   they're probably teaching elsewhere#|
1 5 782 A     f   yea#|
1 5 783 C     c   . and [@] they don't seem to
                  bother anybody#|
1 5 784 A     f   no#|                           <------
1 5 785 C     c   they seem to know their way around#|
1 5 786 A     r   so it does seem#
1 5 787 A         a fairly self-contained *unit on
                  its own*#|
-----------
1 5 1175 A    t   *I'm* also#
1 5 1176 A    [t] reasonably anxious#
1 5 1177 A        to bump into people#
1 5 1178+A        but perhaps one just . sort of - holds on
1 5 1178 A        that -#|
1 5 1179 D    r   well yes#
1 5 1180+D        that's - - - it's not so easy as .
                  you think
1 5 1180 D        *really*#|
1 5 1181 A    b   *no*#|                           <------
1 5 1182 D    c   because - being over here#
1 5 1183 D        we tend to be a bit isolated#|
1 5 1184 A    f   yeah#
1 5 1185 A        [m] - -#|
1 5 1186+D    c   [m] specially as we don't go to . to
                  coffee
1 5 1186 D        over in [dhi] . *the main building
                  you see*#|
1 5 1187 A    b   *no .#                           <------
1 5 1188 A        yes*#\
1 5 1189-A    r   that's what~|
```

Appendix 3: Sample frequency counts: Text: L1-5T.TXT

==========

```
Speaker A
Number of turns:
AFT:   309     total no. of formal turns prod. by A
ArT:    68     no. of turns produced by A
AtT:    16     no. of t-turns produced by A
AaT:     8     no. of a-turns produced by A
AfT:    74     no. of f-turns produced by A
AlT:     3     no. of l-turns produced by A
AbT:    92     no. of b-turns produced by A
AmT:     4     no. of m-turns produced by A
AuT:     0     no. of u-turns produced by A
&&

Number of words produced by A
ATW:  1786     total no. wrds produced by A
ArW:  1384     no. of words prod. by A as floor holder
AtW:   264     no. of words prod. by A during t-turns
AaW:    19     no. of words prod. by A during a-turns
AfW:    53     no. of words prod. by A during f-turns
AlW:     0     no. of words prod. by A during l-turns
AbW:    65     no. of words prod. by A during b-turns
AuW:     0     no. of words prod. by A during u-turns
&&


Number of "attacks"
ATt:    55     total no. attacks on A when being FH
ArT:     3     no. of attacks of type r
Att:    14     no. of attacks of type t
Aat:    14     no. of attacks of type a
Aft:     2     no. of attacks of type f
Alt:     0     no. of attacks of type l
Abt:    17     no. of attacks of type b
Amt:     4     no. of attacks of type m
Aut:     0     no. of attacks of type u


Number of words prod. by A during attacks
ATw:   114     total no. wrds produced by A during attacks
Arw:     2     no. of words prod. by A during r-turn attacks
Atw:    30     no. of words prod. by A during t-turn attacks
Aaw:    58     no. of words prod. by A during a-turn attacks
Afw:     2     no. of words prod. by A during f-turn attacks
Alw:     0     no. of words prod. by A during l-turn attacks
AbW.    18     no. of words prod. by A during b-turn attacks
Amw:     3     no. of words prod. by A during m-turn attacks
Auw:     0     no. of words prod. by A during u-turn attacks
```

Porting a Stochastic Part-of-Speech Tagger to Swedish

Douglass R. Cutting
Cupertino

Abstract

The Xerox Part-of-Speech Tagger (XPOST) claims to be *practical*. One aspect of practicality as defined here is *reusability*. Thus it is meant to be easy to port XPOST to a new language. To test this, XPOST was ported to Swedish. This port is described and evaluated.

Practical Part-of-Speech Tagging

In previous work on part-of-speech tagging, a *practical* part-of-speech tagger was defined as one with the following set of properties (Cutting *et al* 1992):[1]

- **accurate**

```
A     tagger should assign the  correct part  of
det/2 n      modal  v      det  adj/2   n/2   prep

speech to     every word  in     the  text.
n      prep/2 det   n     prep/4 det  n
```

  While 100% accuracy is desirable, it may not in fact be achievable. When text is manually tagged by several linguists, the tags assigned differ by a few percent, suggesting an effective upper-bound for tagging accuracy (Church 1989).

- **fast**
  Ideally, the addition of part-of-speech tagging to a system will not significantly alter the speed with which text is processed. This may be difficult to evaluate, as systems which incorporate tagging may not operate at all without tagging. As a surrogate, one may compare the cost of assigning tags with that of simply extracting words from text—*tokenization*. If tagging is not significantly slower than tokenizing then its performance impact on complex text processing systems should certainly be minimal.

[1] The Xerox Part-of-Speech tagger is available for anonymous FTP from parftp.xerox.com.

- **robust**
  A tagger should correctly tag text previously unseen by the system. It must accommodate previously unseen words, as unseen texts frequently contain unseen words. New grammatical constructions may also be encountered. Ideally a tagger will accommodate these too. However, before addressing these, one should ask: do previously unseen items occur at such a rate that handling them incorrectly affects overall accuracy? Taggers typically answer this affirmatively for new lexemes and negatively for new constructions.

- **reusable**
  It should be possible to easily configure a tagger to handle a broad range of texts and tasks. Texts vary in things as mundane as typographic conventions and as fundamental as natural languages. Different tasks may require different tagsets, e.g., some may need to distinguish subject and object pronouns, while others may not.

The author previously helped construct the Xerox Part-of-Speech Tagger (XPOST) in an attempt to meet these criteria. The present paper first reviews XPOST in the light of recent Scandinavian work on part-of-speech tagging. It then describes the author's experiences porting XPOST to Swedish while visiting the Swedish Institute for Computer Science (SICS).

Stochastic Part-of-Speech Tagging

Stochastic part-of-speech taggers operate by constructing a probabilistic model of text; then estimating the probabilities of the model, or *training*, and finally, using the trained model to assign parts-of-speech to previously unseen text. The models employed typically contain two sorts of probabilities: *transition probabilities* and *symbol probabilities*. Transition probabilities are recorded for sequences of tags, usually pairs, and indicate the probability of that sequence occurring in text, e.g. the probability that a determiner is followed by a noun. Symbol probabilities record the likelihood that a given input item, typically a word, assumes a given part of speech, e.g. the probability that "bank" is a verb. Given an input sequence (e.g. a sentence) and these two sets of probabilities, one may compute the probability of each possible tag assignment by multiplying all the applicable symbol and transition probabilities. The tag assignment with the highest such product is selected as most likely. This simple methodology has been shown to work quite well (Church 1988).[1]

[1]Comparable results have been achieved with non-stochastic methods (Eineborg *et al* 1993, Voutilainen *et al* 1992).

A weakness with this approach is that symbol probabilities are difficult to estimate for words. A substantial portion of text is composed of low-frequency words. For these words, there are not enough observations make accurate estimates of symbol probabilities. And words which are unknown when training have no observations at all. Compounding this problem, Samuelsson has shown that symbol probabilities are more significant in improving accuracy than transition probabilities (Samuelsson 1993). Together these suggest that, if one is not satisfied with the accuracy of a stochastic part-of-speech tagger, one should attempt to improve symbol probability estimation.

A common approach to such sparse-data problems is to develop an alternate representation which pools data into coarser categories, increasing the number of observations of each of a smaller set of phenomena. In XPOST, each word is represented by its *ambiguity class*—the set of tags it may assume. All words in an ambiguity class are considered identical, and their observations may thus be pooled to provide better estimates.

XPOST guesses ambiguity classes for unknown words based on their suffixes. Frequencies of suffixes of a known words in a text are analyzed to generate a table which, given a suffix, names the ambiguity class which accounts for the vast majority of the words with that suffix. This is similar to the method for handling unknown words proposed by Eklund (1993; 1994).

Another weakness of many stochastic taggers is their reliance upon hand-tagged corpora for training. While hand-tagged corpora do provide accurate estimates, they are very expensive to produce. XPOST avoids reliance on hand-tagged corpora by using a hidden Markov model (HMM).[1] The Baum-Welch (or *forward-backward*) algorithm enables one to estimate symbol and transition probabilities of an HMM without hand-tagged training data (Baum 1972).

The Baum-Welch algorithm operates by incrementally adjusting probabilities to make the training data more likely. One can steer it out of local-maxima by initializing some of the probabilities manually. For example, one might initialize the transition probability between determiner and noun to be higher than the transition probability between determiner and verb. In effect, this permits one to specify simple *a priori* grammatical constraints. Here we see that stochastic taggers are not purely data-driven and self-organizing, as is sometimes claimed by those

[1]HMMs have been used in other taggers, but not in combination with ambiguity classes (Jelinek 1985).

promoting grammar-based taggers, but rather permit integration of linguistic knowledge.

Performance of XPOST on English

On the Brown text collection (Francis *et al* 1982) XPOST achieves the following results:

- **accuracy**: the correct tag is assigned to 96% of the words (88% of the ambiguous words). This accuracy is comparable to that achieved by other stochastic part-of-speech taggers trained on tagged data.

- **robustness**: In experiments on texts with unknown words, 77% of unknown words are tagged correctly.

```
Corandic is    an   emurient grof  with  many  fribs ; it
n        v3sg  det  adj      n     prep  adj   pl      pro

granks  from  corite, an   olg  which  cargs  like   lange .
v3sg    prep  n       det  n    pro    v3sg   prep   n
```

- **reusability**: has been ported to French, German[1] and Swedish (described subsequently).

- **speed**: in Common Lisp on a Sun SPARCStation2 the tagger requires approximately one millisecond per word tagged with the Brown tagset. With 38 tags in 174 ambiguity classes, this tagset is reasonably large. Tagset size is a factor in speed, so one can expect better performance with a smaller tagset. Note that, even with this tagset, tagging (including lexicon lookup) operates at approximately the same speed as tokenization.

Average μseconds per word			
tokenizer	lexicon	tagging	total
604	388	233	1235

XPOST thus appears to meet most of the criteria for practical part-of-speech tagging.

Porting XPOST to Swedish

Teleman's corpus of tagged Swedish was used to evaluate XPOST on Swedish (Teleman 1974). The methodology was similar to that used for

[1]For more information contact Helmut Schmid <schmid@ims.uni-stuttgart.de>.

the Brown corpus. First a lexicon was induced from the entire collection containing, for each word, the list of the tags which it may be assigned. The corpus was then divided into two sections, one containing the even numbered sentences and one containing the odd numbered sentences. The former were used, without tags, to train XPOST. The latter sentences were then automatically tagged by XPOST. The tags thus assigned were then compared with the tags assigned by Teleman.

The Teleman tagged corpus contains around 85 thousand words tagged with 259 unique tags. Many of these tags occur very infrequently in the corpus, making parameter estimation difficult. The tagset was thus initially recoded to the 13 tags specified by Samuelsson (Samuelsson 1993). With this tagset, XPOST tagged 91% of the words correctly.

Examination of the errors suggested that XPOST might do better with a somewhat more refined tagset. This is not usually a good idea, as it creates more parameters to be trained, and hence, less evidence per parameter. The addition of some distinctions may not change the number of ambiguous forms, but may provide more precise grammatical contexts for disambiguating neighboring ambiguous forms. By this logic, genitive names and nouns were broken out as separate tags, and pronouns were broken into four categories: relative, personal, genitive and object. After these changes accuracy rose to 95%.

Issues

Some issues which remain to be examined in stochastic taggers include:

- Should common words be modeled individually? Some authors have proposed (e.g. Kupiec 1992) have proposed that high-frequency words should have their own ambiguity class, even if the set of tags in the class is not distinct from that in other classes. The Swedish word "om" might benefit from this treatment. It is a high-frequency word which may be used as an adverb, a conjunction or a preposition. Other words with the same ambiguity, e.g. "efter" and "sedan", are infrequent enough to benefit from having their statistics pooled, while "om" is frequent enough that it may fare better on its own.

- Voutilainen *et al* (1992) have developed a tagging method which achieves high accuracy, but which moreover, can accurately predict its errors. In other words, rather than generating the wrong tags, it is able to pass the ambiguity along so that it may be resolved by higher-level processing. This is clearly a superior property. It remains to be seen if a stochastic tagger can implement this.

Acknowledgments

Thanks to Jussi Karlgren for suggesting that I visit SICS, to Christer Samuelsson for arranging my stay, and to the whole SICS NLP group for making that stay fun.

Bibliography

Baum, L.E. 1972. *An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of a Markov Process.* In INEQUALITIES. 3: pp. 1–8.

Church, K. 1989. *A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text.* In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing.*

Church, K.W. 1988. *A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text.* In *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, Texas.

Cutting, D., J. Kupiec, J. Pedersen and P. Sibun. 1992. *A Practical Part-of-Speech Tagger.* In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy.

Eineborg, Martin and Björn Gambäck. 1993. *Tagging Experiments Using Neural Networks.* In Eklund (ed.) *Nodalida '93 – Proceedings of '9:e Nordiska Datalingvistik-dagarna', Stockholm 3-5 June 1993.* Stockholm..

Eklund, Robert. 1994. (ed) *Nodalida '93 – Proceedings of '9:e Nordiska Datalingvistik-dagarna', Stockholm 3-5 June 1993.* Stockholm.

Eklund, Robert. 1994. *A Probabilistic Word Class Tagging Module Based On Surface Pattern Matching.* In Eklund (ed), *Nodalida '93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3-5 June 1993.* Stockholm.

Eklund, Robert. 1993, *A Probabilistic Word Class Tagging Module Based On Surface Pattern Matching.* Stockholm University, Department of Linguistics, Computational Linguistics.

Francis, W.N. and F. Kucera. 1982. *Frequency Analysis of English Usage.* Houghton Mifflin.

Jelinek, F. 1985. *Markov Source Modeling of Text Generation.* In *Impact of Processing Techniques on Communication*, J.K. Skwirzinski, Editor. Nijhoff: Dordrecht.

Kupiec, J.M. 1992. *Robust Part-of-Speech Tagging Using a Hidden Markov Model.* Xerox Palo Alto Research Center.

Samuelsson, Christer. 1993. *A Morphological Tagger Based Entirely on Bayesian Inference.* In Eklund (ed): *Nodalida '93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3-5 June 1993*, Stockholm.

Teleman, U. 1974. *Manual för Grammatisk Beskrivning av Talad och Skriven Svenska.* University of Lund.

Voutilainen, Atro, Juja Heikkilä and Arto Anttila. 1992. Constraint Grammar of English. Department of Linguistics, University of Helsinki.

Tagging Experiments Using Neural Networks

Martin Eineborg and Björn Gambäck[1]
Stockholm

Abstract

The paper outlines a method for automatic part-of-speech tagging using artificial neural networks. Several experiments have been carried out where the performance of different network architectures have been compared to each other on two tasks: classification by overall part-of-speech (noun, adjective or verb) and by a set of 13 possible output categories. The best classification rates were 93.6% for the simple and 96.4% for the complex task. These results are rather promising and the paper compares them to the performance reported by other methods; a comparison that shows the neural network completely compatible with pure statistical approaches.

1. Introduction

Rayner *et al* (1988) experimented with using a formal grammar along with example-sentences to deduce a lexicon. Unfortunately, the exponential explosion that followed from ambiguities in the grammar caused the system to be very slow. The project described in this paper builds on the assumption that one way to attack this problem would be to let a neural network suggest restrictions on the possible word-classes of the unknown word derived from its word-ending and context.

Another motivation for our project is that within the language area connectionist models have so far proved discouragingly unsuccessful compared to other methods. Even though they have been tried out for several applications, such as semantic clustering, preposition choice, etc., the only language area where artificial neural networks have been successfully applied on a larger scale has been speech; however, all currently leading speech recognition systems (the ones in the US DARPA race) have discarded neural nets for Hidden Markov Models, a statistical method. The current state of affairs should however hardly be taken to be the permanent truth. The need for different machine learning methods within the language area should be evident; in this paper we will single out the topic of part-of-speech tagging for special attention, but the last

[1]The work reported here was funded by the Swedish Institute of Computer Science (Asea Brown Boveri, Telefon AB LM Ericsson, Försvarets Materielverk, IBM Svenska AB, NUTEK, and Telia AB). We would like to thank Lars Asker (Stockholm University), Ivan Bretan (IBM), Douglass Cutting (Xerox Parc), Jussi Karlgren (SICS), Pat Langley (Siemens), and Christer Samuelsson (SICS) for helpful discussions and suggestions.

words for other areas such as disambiguation, document matching, information retrieval, grammar and transfer-rule induction, etc., have certainly not been said.

The experiments we have carried out have used different back-propagation network architectures in order to assign part-of-speech tags to unknown words. A brief background to artificial neural networks and the back-propagation algorithm is given in the rest of this section. Section 2 then goes on to describe the different network architectures used in our experiments. The networks were trained on both morphological and (local) context information extracted from a tagged text corpus and then evaluated on previously unseen data from the same corpus. The results of the different experiments are given in Section 3. Section 4 compares these results to other possible methods of solving the problem, i.e., pure statistical and rule-based approaches; finally Section 5 sums up the previous discussions and points to possible future extensions.

Artificial Neural Networks

Several researchers around 1940 suggested that a more brain-like machine should be created. A first step in this direction was taken when McCulloch and Pitts (1943) proposed a model of a neuron, which, just like the biological neuron, takes several inputs and produces one output. The changes in synapses are simulated by weight variables. Modification of the weights is handled by a learning rule. A weight has two features: the sign of the weight determines if the incoming impulse is excitatory or inhibitory and the absolute value of the weight determines to what degree notice should be taken to the incoming impulse. When the incoming values are above a certain level (the threshold) the neuron fires according to a firing rule. In the McCulloch & Pitts model the firing rule can be expressed by the following simple mathematical formula:

the neuron fires iff $\Sigma_k \, x_k \, w_k > \theta$

where x_k is the value received from neuron k
w_k is the weight associated with input from neuron k
θ is the threshold

This model uses only a two-valued output indicating firing, or not. It is still the basis of many neural networks, but has been improved upon several times, in particular when Widrow and Hoff came up with a learning rule called the Widrow-Hoff rule or the delta rule (Widrow 1962).

It can be expressed as:

$$w_k(t+1) = w_k(t) + \alpha\, \delta_k(t) x_k(t)$$

where α is a constant (gain term) typically $0.01 \leq \alpha \leq 10$
$w_k(t)$ is the value of weight k at time t
$\delta_k(t)$ is the error of neuron k at time t
$x_k(t)$ is the incoming value from neuron k at time t.

It was shown by Rosenblatt (1962) that the delta rule causes the weights to converge. He also developed the perceptron, a neuron able to classify binary or continuous valued input into one of two classes; however, a serious blow against neural science came when Minsky and Papert (1969) showed that a perceptron neural network consisting of only one layer is unable to handle nonlinear functions; to do so a hidden layer has to be included in the net. A hidden neuron receives input from other neurons and transmits output to other neurons. A hidden layer consists only of hidden neurons. In 1986 Rumelhart, Hinton, and Williams came up with a network that could handle hidden layers. The method is called backpropagation and will be further described below.

Another model was created by Kohonen (1984/88). It differs from the previous in that it organizes the input data by itself without the correct output pattern being presented, i.e., it uses unsupervised learning. A Kohonen net consists of a number of neurons organized in a two-dimensional plane called a map. The input pattern is given to all neurons at the same time. The neuron for which the Euclidean distance between the input-vector and the weight-vector is a minimum is selected as being the response of the given pattern.

The Backpropagation Algorithm

Backpropagation uses a two-phase learning cycle. During the first phase, the input pattern is propagated through the network. Some sort of distance, usually the Euclidean distance, is calculated between the actual output and the desired output of the net. This distance is the error of the net. The second phase starts with the error being propagated backwards through the net, adjusting the weights along its way. Then the next pattern can be processed. This cycle, called an epoch, continues until the net satisfactory has learnt all patterns, the weights are then frozen and need not be altered. The neurons used differ from those of McCulloch and Pitts in that real values are used as weights, thresholds, and outputs. The output of the neuron is given by:

$o_m = 1 / (1 + \exp\{a_i\})$

where $a_i = \Sigma_j (w_{ij} * x_{ij}) + \theta_i$ is the activation of the i:th neuron.

and w_{ij} is the j:th weight of neuron i

x_{ij} is the j:th input to neuron i

θ_i is the threshold of neuron i.

There are two weight adjustment rules:

for output neurons the error: $\delta_{pj} = (\theta_{pj} - o_{pj})\, o_{pj}\, (1 - o_{pj})$

for hidden neurons the error: $\delta_{pj} = (\Sigma_k\, \delta_{pj} w_{kj})\, o_{pj}\, (1 - o_{pj})$

2. Test Set-ups

A large number of backpropagation network architectures were tested. This section will describe how the net-input was encoded and the actual architectures of the different networks used in the experiments.

Encoding of Network Input

In the text below we will need to use several character sets, e.g., Alphabet_1 and Alphabet_2 respectively defining the Swedish and ASCII alphabets, sets for Swedish vowels and consonants, and some morphologically and phonologically motivated subsets of these. When defining the mappings of the network inputs, we will also need to discuss a particular type of vectors, namely binary vectors of different length with only one 1. These will be referred to as Bin_n where n is the number of digits in the vector. Strings of characters, lexemes, will be subindexed according to what alphabet the included characters belong to.

To represent the encoding of letters, we will introduce five functions which informally can be said to map the character sets above onto the binary vectors Bin_n and perform the following tasks: f_1 simply divides Alphabet_1 into vowels and consonants; f_2 further subdivides the consonants by phonetic category, that is into plosives, fricatives, laterals, trills, and nasals; f_3 is like f_2, but the vowels A and E are singled out from the others, since they behave rather in a special way when inflection is performed; while f_4 and f_5 encode the entire Swedish and ASCII alphabets, respectively.

For the encoding of grammatical categories we will introduce five other functions mapping from the lexemes to the binary vectors, thus: h_1 splits Lexeme_1 into nine categories: nouns, adjectives, verbs, pronouns,

determiners, adverbs, prepositions, conjunctions, and infinitival markers; h_2 adds two more categories, one for auxiliaries and one for sentence delimiters; h_3 is like h_1, but with special categories for auxiliaries, idiomatic expressions, and present and past participles. It also splits the conjunctions into subordinating and coordinating ones; h_4 further subdivides the adjectives by comparative form (i.e., positive, comparative, and superlative) and the adverbs by type (normal, comparative, superlative, and comparison); finally, h_5 does for $Lexeme_2$ what h_1 does for $Lexeme_1$, but with extra categories for names, numbers, characters, and sentence delimiters.

Network Architectures

All backpropagation networks were three layer architectures consisting of an input layer, a hidden layer, and an output layer. Information was given in localized form. In order to examine the feasibility of the approach, the sizes of the networks were initially kept at moderate levels to increase only gradually. Two information sources were used: the internal structure of the lexeme and N-grams. An N-gram refers to the grammatical categories of N-1 neighbouring words, so we will use 1-gram to refer to the word itself, a 2-gram (here) denotes the word itself and the word to the left, a 3-gram denotes a 2-gram and the word to the right, and so on. When combining the two information sources the vectors were simply appended. The resulting vector was then fed to the network.

All networks in this paper were trained and tested using the Teleman corpus (Teleman 1974). This text consists of almost 80000 tagged Swedish words gathered from a wide range of different genres. The training could be very time consuming, but fortunately for the most part the networks converged rapidly. Typically, only a few epochs were needed until a satisfactory performance was reached. The small number of epochs needed is very likely a result of the text used for training. Since it contains many duplicates, most input patterns were seen and trained several times during one epoch. The training continued as long as seemed reasonable or as long as the performance did not decrease when evaluated on previously unseen material.

TABLE 1: Summary of the network setups for the experiments

Net	Gram (N)	Category-function	Letter-functions 6	5	4	3	2	1	Training epochs	examples
<18,5,3>	3	h_1	-	-	-	-	-	-	2000	5000
<26,5,3>	3	h_1	-	-	f_1	f_1	f_1	f_1	2000	5000
<42,20,3>	3	h_1	-	-	f_2	f_2	f_2	f_2	2000	5000
<44,20,3>	3	h_1	-	f_1	f_2	f_2	f_2	f_2	2000	5000
<52,20,3>	3	h_1	-	f_1	f_3	f_3	f_3	f_3	2000	5000
<73,20,3>	3	h_1	-	f_1	f_3	f_3	f_3	f_4	2000	5000
<136,20,3>	3	h_1	-	f_1	f_4	f_4	f_4	f_4	2000	5000
<165,20,3>	3	h_1	f_1	f_4	f_4	f_4	f_4	f_4	2000	5000
<165,20,3>	3	h_1	f_1	f_4	f_4	f_4	f_4	f_4	2000	7500
<165,20,3>	3	h_1	f_1	f_4	f_4	f_4	f_4	f_4	1000	10000
<165,40,3>	3	h_1	f_1	f_4	f_4	f_4	f_4	f_4	100	7500
<169,20,3>	3	h_2	f_1	f_4	f_4	f_4	f_4	f_4	100	10000
<204,40,3>	3	h_3	f_4	f_4	f_4	f_4	f_4	f_4	50	20000
<212,40,3>	3	h_4	f_4	f_4	f_4	f_4	f_4	f_4	100	20000
<282,80,13>	3	h_5	-	-	f_5	f_5	f_5	f_5	50	30000
<295,80,13>	4	h_5	-	-	f_5	f_5	f_5	f_5	50	30000
<423,80,13>	4	h_5	f_5	f_5	f_5	f_5	f_5	f_5	150	30000

Table 1 describes each network in some detail. The number of neurons of a specific network is indicated by a triple <I,H,O> where I is the number of neurons in the input layer, H the same for the hidden layer, and O for the output layer. The other columns of the table define mapping functions, indicate the number of training epochs, etc. Thus the first net, for example, is called <18,5,3>, since it had 26 neurons in total. It used 3-grams only, so its single source of information was that of the context. The grammatical category mapping used, h_1, was very simple distinguishing only between nouns, adjectives, verbs, pronouns, determiners, adverbs, prepositions, conjunctions, and the infinitival marker. Note that no information at all was extracted from the unknown word. As shown in the table, it was trained for 2000 epochs on a text consisting of 5000 examples.

The other nets combined the two information sources available by also inspecting the letters of the unknown word. In order not to make the networks unnecessarily large the mapping between the actual letter and its representation was kept as simple as possible. At first letters mapped onto one of only three classes: vowels, consonants, or ∅, the latter indicating the lack of any input character in a specific position. This letter-classification was refined first by subdividing the consonants (plosives, fricatives, laterals, trills, and nasals) and later on by separating the letters A and E from the other vowels. Some nets (like <165,40,3>) were included in order to examine if the result would improve with a larger hidden layer, while other nets (as <169,20,3>) mapped the 3-grams differently, for example with the h_2 function which separates the auxiliary verbs from the domain ones and also recognizes sentence delimiters, enabling the tagger to categorize the first and last words of a sentence.

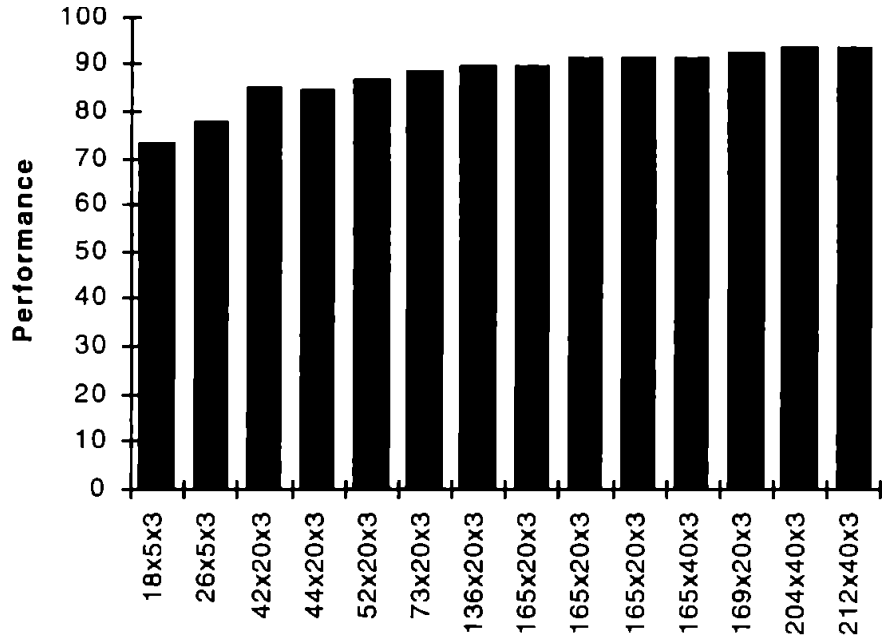


FIGURE 1: Peak performance of the nets on the simple task

3. Results

The networks were tested using an unseen part of the Teleman corpus. The corpus consists of several different types of text. Thus the results should be as general as possible. Figure 1 shows the performance of the nets on the first classification task, part-of-speech categorization. The network with the worst result was not surprisingly the <18,5,3> one, which only used 3-grams. It reached a classification rate of about 73% which is not so bad considering that it extracts no information at all from the word that is to be categorized. When information was added about the internal structure of the unknown word the networks performed better. The more detailed this information was the better did the network perform. The amount of examples used for training was also a parameter that varied. Generally, the more examples that were available to the network the better it performed. The networks with the best results were nets <204,40,3> and <212,40,3>. They both reached a classification rate of 93.6%.

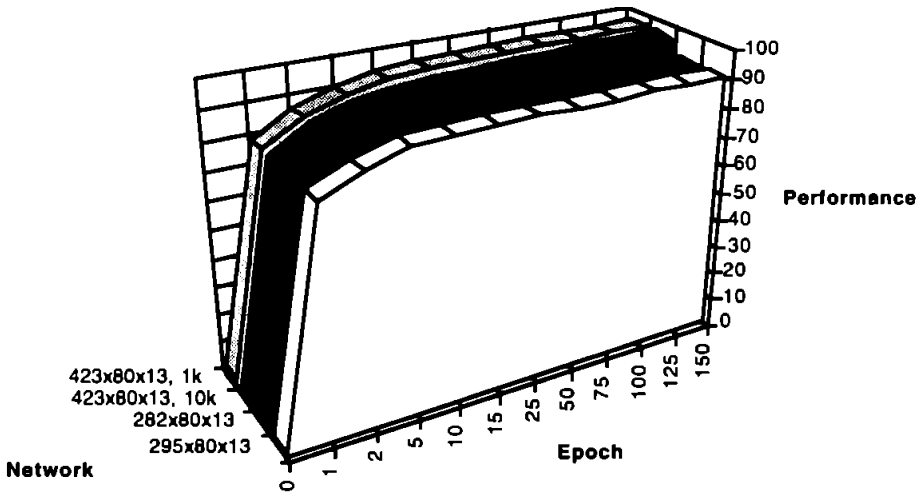


FIGURE 2: Performance of the nets on the complex task

The best result for a network which could classify more than nouns, adjectives, and verbs was 96.4% as shown in Figure 2. This was achieved by the <423,80,13> network, when trained using 30000 examples and tested (like all the other nets) on 1000 unseen examples. To evaluate the consistency of these figures, this net was also tested on an uncommonly large set of 10000 unseen examples. As could be expected when comparing the sizes of the training versus the test sets, this gave a slight decrease in performance, with a top result of 95.80%, as shown by the graph called "<423,80,13>, 10k".

Table 2 shows an example of network outputs. The clause *"(.) i södra Asien (har)"* ["(.) in Southern Asia (have)"] was fed to the <295,80,13> net together with the tags (following the ">" sign). As can be seen from the name *"Asien"*, it had a difficult time separating names from ordinary nouns.

TABLE 2: Example of network output

Categories:	noun	adjective	verb	preposition	adverb	determiner
pronoun	character	conjunction	number	name	sent. del.	inf. mark
Output (right):	0.000000	0.000000	0.000000	0.999984	0.000000	0.000001
0.000014	0.001059	0.000121	0.000000	0.000009	0.000042	0.000000
Right answer:	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Pattern:	.>IP I>PR SÖDRA>POSU					
Output (right):	0.000000	0.939394	0.000104	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000138	0.000002	0.000000
Right answer:	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Pattern:	I>PR SÖDRA>POSU ASIEN>PN					
Output (wrong):	0.999468	0.000000	0.000000	0.000000	0.000000	0.325023
0.000000	0.000031	0.000001	0.000000	0.000000	0.000000	0.000000
Right answer:	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
Pattern:	SÖDRA>POSU ASIEN>PN HAR>HVPS					

4. Discussion

In this section we will try to compare the results of the previous section with those that have been obtained using statistical and rule-based methods. First, however, we note that Veronis & Ide (1990) used an approach akin to a neural network in extracting lexical information from a machine readable dictionary. Their results were rather discouraging, in that they managed to identify the correct sense of a word (that already occurred in the dictionary) in only 71.74% of the cases. Nakamura *et al* (1990) investigated word category prediction using a neural network architecture called NETgram, a four layer architecture based on backpropagation. The grammatical categories of the preceding words were used to predict the category of the next word. They reported a word recognition rate of about 68%.

Recently a rule-based approach has achieved some extraordinary results (Voutilainen *et al* 1992). They report a classification rate of 99.7%. The downfalls of their method (and all rule-based ones) are that it is very time consuming to develop the rules and the system produced is highly language dependent. The main objection to their method is however that it also demands a very large lexicon (again making the approach highly language specific). The lexicon they used covered about 95% of all lexemes appearing in the texts, making the comparison of performance figures somewhat unfair.

Samuelsson (1994) suggests a method based purely on statistical evidence. With a success rate of 95.38%, it does not do as well as the method Voutilainen *et al* use, but on the other hand no external lexicon is needed and no language specifics are assumed. The best result was reported using a 4-gram, inspection of 6 letters, and syllable information. The test setting closely resembles that of the <423,80,13> net above, which reached a classification rate of 96.4%. For the same task the Xerox Parc system "Tagger" (Cutting *et al* 1992) based on a Hidden Markov Model (HMM) method also was able to classify 95% of the words correctly (Cutting 1994). Even though this comparison thus shows the neural net approach ahead by a margin, it indicates that the methods are virtually equivalent for the task at hand.

5. Conclusions and Future Work

We have described a series of experiments where different three-layered back-propagation network architectures were used for the task of recognizing unknown words for a natural language system. Two main tasks were performed: in the first the nets were to classify words by overall part-of-speech (noun, adjective or verb) only, while the second task involved a larger set of 13 possible output categories. The best results for the simple task were obtained by networks consisting of 204-212 input neurons and 40 hidden-layer neurons, reaching a classification rate of 93.6%. The best result for the more complex task was 96.4%, which was achieved by a net with 423 input neurons and 80 hidden-layer neurons. The results are overall rather promising and they are completely compatible with those achieved by purely statistical methods; however, they are still inferior to those reported by a rule-based approach, albeit on a somewhat different task.

A possible way to improve on the results could be to combine several networks, for example have we done some initial experiments using a self-organizing map of the Kohonen type. The idea was to use this map to transform the letters of the unknown word to the two dimensional map and then feed the coordinates of this map to a backpropagation network together with the grammatical categories of the surrounding words; however, this approach has not been very successful - yet. Early results indicate that this combination does not perform better than the backpropagation network which only used 3-gram. The map failed to capture the structure of the words. This approach is still being investigated though.

References

Cutting, D. 1994. *Porting a Stochastic Part-of-Speech Tagger to Swedish.* In Eklund (ed), *Nodalida'93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3-5 June 1993.* Stockholm.

Cutting, D., J. Kupiec, J. Pedersen and P. Shibun. 1992. *A Practical Part-of-Speech Tagger* pp 133-140, *Proceedings of the 3rd Conference on Applied Natural Language Processing*, Trento, Italy.

Eklund, Robert. 1994. (ed) *Nodalida'93 – Proceedings of '9:e Nordiska Datalingvistik-dagarna', Stockholm 3-5 June 1993.* Stockholm.

Ide, N.M. and J. Veronis. 1990. *Very Large Neural Networks Word Sense Disambiguation.* pp 366-368, *Proceedings of the 9th European Conference on Artificial Intelligence*, Stockholm, Sweden (also as pp 389-394, *Proc. 13th International Conference on Computational Linguistics*, Helsinki, Finland, Vol. 2).

Kohonen, T. 1984/1988. *Self-Organization and Associative Memory.* Springer-Verlag, Heidelberg, Germany.

McCulloch, W. S. and W. H. Pitts. 1943. *A Logical Calculus of the Ideas Imminent in Nervous Activity.* pp 115–133, BULLETIN OF MATHEMATICAL BIOPHYSICS, Vol. 5.

Minsky, M. and S. Papert. 1969. *Perceptrons: an Introduction to Computational Geometry.* MIT Press, Massachusetts.

Nakamura, M., K. Maruyama, T. Kawabata and K. Shikano. 1990. *Neural Network Approach to Word Category Prediction for English Texts.* pp 213-218, *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki, Finland, Volume 3.

Rayner, M., Å. Hugosson and G. Hagert. 1988. *Using a Logic Grammar to Learn a Lexicon.* pp 524-529, *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary. Also available as *SICS Research Report - R88001*, Stockholm, Sweden.

Rosenblatt, F. 1962. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanism.* Spartan Books, New York.

Rumelhart, D.E., G.E. Hinton and R.J. Williams. 1986. *Learning internal representations by error propagation.* PARALLEL DISTRIBUTED PROCESSING, Vols. 1 and 2, The MIT Press, Cambridge, Massachusetts.

Samuelsson, C. 1994. *Morphological Tagging Based Entirely on Bayesian Inference.* In Eklund (ed): *Nodalida'93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3-5 June 1993.* Stockholm.

Teleman, U. 1974. *Manual för grammatisk beskrivning av talad och skriven svenska* (in Swedish). Studentlitteratur, Lund, Sweden.

Widrow, B. 1962. *Generalization and information storage in networks of ADALINE neurons.* SELF-ORGANIZING SYSTEMS, Spartan Books, New York.

Voutilainen, A., J. Heikkila and A. Anttila. 1992. *Constraint Grammar of English.* Publication #21, Department of General Linguistics, University of Helsinki, Helsinki, Finland.

A Probabilistic Word Class Tagging Module Based On Surface Pattern Matching

Robert Eklund
Stockholm

Abstract

This paper[1] treats automatic, probabilistic tagging. First, residual, untagged, output from the lexical analyser SWETWOL[2] is described and discussed. A method of tagging residual output is proposed and implemented: the *left-stripping method.* This algorithm, employed by the module ENDTAG, recursively strips a word of its leftmost letter, and looks up the remaining 'ending' in a dictionary. If the ending is found, ENDTAG tags it according to the information found in the dictionary. If the ending is not found in the dictionary, a match is searched in ending lexica containing statistical information about word classes associated with the ending and the relative frequency of each word class. If a match is found in the ending lexica, the word is given graded tagging according to the statistical information in the ending lexica. If no match is found, the ending is stripped of what is now its left-most letter and is recursively searched in dictionary and ending lexica (in that order). The ending lexica – containing the statistical informaiton – employed in this paper are obtained from a reversed version of *Nusvensk Frekvensordbok* (Allén 1970), and contain endings of one to seven letters. Success rates for ENDTAG as a stand-alone module are presented.

1 Introduction

One problem with automatic tagging and lexical analysis is that they are never (as yet) 100 % accurate. Varying tagging algorithms, using different methods, arrive at success rates in the area of 94–99 %.[3] After machine analysis there remains an untagged residue, and the complete output may – somewhat roughly – be divided into three subgroups:

1 A group of unambiguously tagged words.

2 A group of homographs given alternative tags.

3 A residual group lacking tags.[4]

---

[1]This paper is an abbreviated version of my diploma paper in computational linguistics with the same title, presented in April 1993 at the department of linguistics, computational linguistics, Stockholm University.

[2]Karlsson 1990; Koskenniemi 1983a,b; Pitkänen 1992.

[3]See e.g. Church (1988), Garside (1987), DeRose (1988).

[4]There is a bulk of words which is never found in this group, preponderatingly those belonging to the closed words classes, since these normally are found in the lexicon.

Whereas the second of these groups is treated in Eriksson (1992), the task undertaken in this paper is to develop an algorithm for tagging material which has come through lexical analysis untagged.

The paper falls into the following areas:

First, untagged, residual output from the lexical analyser SWETWOL (Karlsson 1990; Koskenniemi 1983a,b; Pitkänen 1992) is described and analysed. This is done in order to pin down what input is, in one way or another, problematic to an automatic tagger. This is covered in section 3.

This paper presents a probabilistic tagger – henceforth ENDTAG – which tags according to statistics on the relations between final-letter combinations and word classes. The statistical information was obtained from the listings in NFO (Allén 1970) and collected in special ending lexica. This is described in section 4.

The ENDTAG module is presented in section 5. ENDTAG is based on what is here called the *left-stripping algorithm*, which recursively strips a word from its leftmost letter and compares the remaining ending[1] with the statistical information in ending lexica described in section 4.

The results of ENDTAG are evaluated in section 6.

2 Method

The untagged material used in this paper consists of residual files from the lexical analyser SWETWOL in Helsinki. SWETWOL was run on 831.289 words, whereof 10.988 came out untagged. Since SWETWOL yields output files of words on a word-for-word basis – thus ignoring (more or less) things like lexicalised phrases, particle verbs (ubiquitous in Swedish) and the like, words were only analysed one-by-one. A conjectural supposition is that a higher rate of accuracy is to be expected if context is also considered, as attempts with purely heuristic parsers show (cf. Källgren 1991b;c, Brodda 1983). On the other hand, it can be argued that there is palpable explanatory value in trying to find out how much information can be extracted from the words alone, neglecting their immediately adjacent 'text-mates'.

The success rate of any automatic tagger or analyser, *per se* and in comparison with other automatic taggers, is of course dependent on what tagset is being employed. The more general it is, i.e., the fewer the tags,

[1]The word 'ending' will throughout this paper denote any word final letter cluster, be this a grammatical suffix or not.

the more 'accurate' the output will be, due to the lack of more subtle subcategories. Since it was judged important that the tagset easily harmonise with already existing tagsets employed in other systems, the ending statistics were obtained from *Nusvensk Frekvensordbok*, NFO hereinafter (Allén 1970). I opted to adhere to the tagset employed therein, thus, the tags employed in this paper constitute a proper subset of the NFO tags.. It should be pointed out that NFO also contains tags for subcategories. The tagset employed by ENDTAG is shown in table 1.

TABLE 1 : The tagset employed by the ENDTAG module.

ABBREVIATION	WORD CLASS
ab	adverb
al	article
an	abbreviation
av	adjective
ie	infinitival marker
in	interjection
kn	conjunction
nl	numeral
nn	noun
pm	proper noun (proprium)
pn	pronoun
pp	preposition
vb	verb
**	non-Swedish unit
NT[1]	*Not tagged in NFO*

The ENDTAG module was implemented in COMMON LISP.

3 A Curt Description of the Untagged Output

In order to pin down what needs to be accounted for in tagging algorithms for arriving at better figures, one naturally has to scrutinise, with as great a punctilio as possible, the contents of that residual group of untagged words. I will here briefly list just a few observations made.[2]

In the untagged material, **proper** and **place nouns** abound! This is not really surprising, since they do not to any greater extent exhibit consistent morphological patterns.[3] It is also hard to list them all in the lexicon. Liberman and Church (1992) mention that a list from the Donnelly marketing organisation 1987 contains 1.5 million proper nouns (covering

[1]Since it was found that not all words in the computer readable version of NFO were tagged, an additional tag was created to render the format consistent. Hence, the tag 'NT' was added.

[2]For a more detailed account, the reader is referred to Eklund 1993.

[3]Of course *some* consistent patterns can be found. Thus the suffix *-(s)son* in Swedish typically denotes a surname, as in *Eriksson*, *Svensson* etc.

72 million American households). Since these have any number of origins, it is not feasible to cover them either with morphological rules or with a lexicon.[1]

Abbreviations were also common, which is more surprising, since all these should, it is assumed, have been expanded/normalised in the pre-processing. A related – and harder – problem concerns lexical abbreviations and **acronyms**.

Compounds constitute a notorious problem in all automatic processing of Swedish. Because they are legion, compounds constitute a very dire problem for any tagging module working on Swedish text. It might even be hard to decide where the compound border is located.

A related problem is encountered in what I call **complex compounds**. By that I mean compound words created in ways diverging from the 'normal' compounding of two ordinary words. One example of this is when more than two words are compounded. Instances of such compounds are:

Djursholms-Bromma-Lidingö-gängen
'The Djursholm-Bromma-Lidingö gangs'
karpatisk-balkansk-bysantiska
'Carpatian-Balcanian-Bysanthinian'
du-och-jag-ensamma-i-världen
'you-and-I-alone-in-the-world'

These clearly exhibit a *word-hyphen-word* pattern which could be formalized thus:

X-(Y-)*Z

These, I assume, would normally obtain the correct tag if one just looked at **Z** alone, and tagged accordingly. Compounds like these, I have found, were rather common in psychological terminology, where it also typically was used rather freely as to word class. A 'word' such as *du-och-jag-ensamma-i-världen* may be used as an adjective or a noun, for example.

---

[1] Something that could be considered here is majuscule heuristics, but this is not done without problems since upper case letters appearing in texts might indicate a wide variety of different phenomena. For example, the first letter of each sentence in a typical text, Roman figures, initials, titles and headings etc. Because of these problems, I chose to let the algorithm exempt majuscules altogether. For further discussion on majuscules, cf. e.g. Libermann & Church (1992), Eeg-Olofsson (1991:IV *et passim*), Källgren (1991b) and Sampson (1991).

A similar problem concerns what I call **slash compounds** like:

Dannemora/Österby
Hornstein/Voristan

... where the slash (/) separates two words according to the formalised pattern:

X/Y

Other phenomena occuring amongst the untagged residue were **professional/special terms, diacritica, archaisms** and **numbers** in various forms.

Another rather amusing, problem is posed by a word like

aaaaahh!

This word is of a recursive disposition which could be formalised thus:

a+h+!+

... where the plus sign denotes any number, equal to or greater than one, and not necessarily the same number in all three instances.

A large part of the untagged output was made up of **foreign words**, expressions and quotations et cetera. Interestingly enough, some of the suffixes used in certain languages are sufficiently unambiguous to permit a graded tagging in Swedish. Thus, some endings of Latin origin, *-ium*, *-ukt* or *-tion*, and some endings of Greek origin, *-graf*, *-lit*, *-ark*, *-skop* or *-logi* are highly unambiguous as to word class.

A problem harder to solve is that of **new words** being continuously created, old words given new interpretations, and then being used as members of other word classes. Thus, even a word like the conjunction *but* can not be considered a sure-fire case. In a phrase like

'*But me no buts!!*'

... 'but' first occurs as a verb in its imperative form, and then as a noun in the plural.[1] One must also point out that *all* words, irrespective of word class, might be used as nouns in a meta-linguistic way, for instance:

---

[1] A Swedish, idiomatic, counterpart would perhaps be *Menna mig hit och menna mig dit!*, the story being a speaker annoyed with a listener who interrupts by saying *but* all the time!

A 'green' would suit this phrase better!
Thou employest too many a 'lest' in thy prolegomenon, young esquire!

Lexicalized phrases typically receive the wrong parses, especially if they allow other constituents to be included 'inside' them. Since, as mentioned before, the module works with but a one-word window, lexicalized phrases cannot be properly accounted for by the module.

4 Obtaining the Ending Lexica

If we are to tag on a probabilistic basis, we need statistical information on the ending/word class relation. Hence, the first task was to create a number of ending lexica containing information as to word classes associated with particular endings. As mentioned earlier, the ending lexica were obtained from the lists in NFO (Allén 1970). NFO is a listing based on one million running words obtained from the material PRESS-65 and exists in computer-readable format. It might be pointed out that NFO is based exclusively on newspaper texts, and that other types of texts would perchance result in different ending lists. (Then again, results always depend on the input material used.)

Ending lexica were created with endings of 1–7 letters[1] – one lexicon per ending length – and word classes and their relative frequencies were calculated. Thus, the final format is as follows:

("ENDING" ((WORD-CLASS$_1$ PERCENTAGE$_1$) (WORD-CLASS$_2$ PERCENTAGE$_2$) (WORD-CLASS$_n$ PERCENTAGE$_n$)))

Word class frequencies are given with four decimals, and the word classes appear in falling order according to frequency. Thus, an authentic typical lexicon entry (from the three-letter ending lexicon):

*("ari" (("nn" 0.7802) ("ab" 0.1209) ("pm" 0.0934) ("**" 0.0055)))*

In other words, if the three final letters of a Swedish words are *-ari*, then there is a 78 % probability that the word is a noun, a 12 % probability that it is an adverb, a 9 % probability that is a proper noun, and finally, a 0.5 % probability that it is a foreign word.

The output files of the ENDTAG module look exactly the same apart from the first member of the list which will be the entire word, instead of as above, a final letter cluster.

[1]The number seven was chosen without any reason in particular.

The number of entries for each of the ending lexica is shown in table 2.

TABLE 2 : Numbers of entries in the ending lexica obtained from NFO.

	Number of letters in each ending lexicon						
	one	two	three	four	five	six	seven
Number of entries in lexica	43	669	3 936	13 176	26 494	38 464	46 179

One thing which cannot be bypassed is the extent to which the number of word classes associated with an ending decreases with the number of letters in the ending, i.e., the longer the final letter cluster, the fewer word classes associated with that ending. Statistics showing these relationships are illustrated in table 3.

TABLE 3 : Number of word classes associated with number of letters in endings (percentages). Zero percent area is marked with bold line.

	Number of letters in ending						
Number of word classes	one letter	two letters	three letters	four letters	five letters	six letters	seven letters
one	30.2	39.5	52.9	71.4	85.4	92.1	94.9
two	23.0	13.8	20.9	19.4	12.3	7.2	4.7
three	–	12.1	12.9	6.2	1.8	0.6	0.3
four	23.0	9.6	6.2	2.0	0.3	0.1	0.1
five	4.7	7.3	3.3	0.7	0.1	–	–
six	7.0	5.2	2.1	0.2	–	–	–
seven	–	4.0	1.0	0.1	–	–	–
eight	9.3	3.7	0.5	–	–	–	–
nine	2.3	1.6	0.2	–	–	–	–
ten	4.7	1.5	0.1	–	–	–	–
eleven	11.6	0.7	–	–	–	–	–
twelve	9.3	0.1	–	–	–	–	–
thirteen	9.3	0.7	–	–	–	–	–
fourteen	4.7	–	–	–	–	–	–
fifteen	2.3	–	–	–	–	–	–

A detailed description of the contents of the ending lexica will not be given here, but one example will perhaps serve as an indicator as to how the module works. Table 4 shows that probability rises as a function of increasing length for three noun declensions in Swedish.

TABLE 4 : Noun percentages (plural/definite/genitive) for Swedish noun declensions one, two and three.

	Paradigm according to the pattern *o / a / e* + suffix (i.e. the three first noun declensions in Swedish).		
Declensions	*-r* (plural)	*-r n a* (plural+definite)	*-r n a s* (plural+definite +genitive)
First declension	74.8	96.9	100.0
Second declension	26.5	97.8	98.4
Third declension	41.5	94.9	97.6

5 A Description of the Left-Stripping Algorithm

The tagging problem has been approached by many a linguist in many a way. Morphological models of Swedish have been provided by Hammarberg (1966), Kiefer (1970), Linell (1972, 1976), Cedwall (1977), Hellberg (1978)[1], Brodda (1979), Blåberg (1984), Eeg-Olofsson (1991:III), Ejerhed and Bromley (1986) and others. These works however, predominantly treat either very specific areas of Swedish morphology with varying degrees of minutae, or are generative models for Swedish word formation.

Probabilistic parsing as such, has been described by e.g. Sampson (1991) and Church (1987). As for tagging, probabilistic/statistical methods in general have been used by e.g. Johansson and Jahr (1982), Marshall (1987), Garside and Leech (1982), Church (1987) and Garside (1987) in the tagging of the LOB Corpus. Eeg-Olofsson (1991:I;IV) describes a statistical model for word-class tagging, and DeRose (1988) treats grammatical disambiguation by means of statistical methods. Johansson and Jahr's project aimed at improving the suffix lists developed for the Brown Corpus by Greene and Rubin (1971). They basically worked by means of a prediction of word classes in relation to grammatical suffixes, and to a certain extent also prefixes. Ejerhed (1988), Karlsson (1990), Källgren (1991a;b), Magnberg (1991) and Eriksson (1992) employ probabilistic methods for lexical analysis. Recent methods have been proposed by Samuelsson (1994) and Cutting (1994).

[1]Implemented by Ivan Rankin (1986).

The algorithm presented in this paper – the left-stripping algorithm – works by simple surface structure pattern matching. The concept is to strip a word of its leftmost letter, look for the resulting 'word' – i.e., the previous word sans its first letter – in a dictionary (e.g. SWETWOL for Swedish). If it is found, the word is tagged according to the dictionary, and the procedure is repeated with the next word. If it is not found, and the number of letters in the word is small enough to have a corresponding ending lexicon, i.e., the same number of letters, the word is looked for in that ending lexicon. If it is found in the ending lexicon, it is tagged, and the whole procedure is repeated with the next word. If it is not found, the word is stripped of what is now its leftmost letter, searched for in the dictionary et cetera. If no match is found even at the final (one) letter stage, the word is tagged thus:

("ENDING" ((NONE 0.0)))

The rationale behind this somewhat pleonastic design of the word class list is a desire to keep the format consistent. The flow chart in FIGURE 1 describes the module.

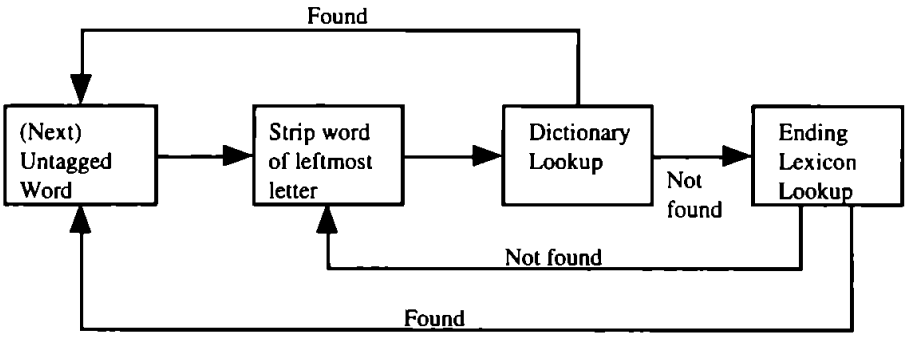


FIGURE 1 – Flow chart of the ENDTAG module.

As mentioned earlier, 'ending' here denotes the *n* final letters of a word, irrespective of whether these be grammatical suffixes, common combinations of any kind or unique word-final clusters. The dictionary lookup is likely to succeed before the ending lexicon, since the length of a complete word (normally) perforce exceeds the length of its ending.[1]

The module iterates over the untagged output list and strips the words recursively until a match is found in either the dictionary or the ending lexica. In the test run carried out here, no dictionary was employed, and the sub-routine intended to perform the dictionary lookup was foregone.

[1]In some instances, however, it might be hard to tell the difference between a word and its ending. Thus, in quoting John Lennon's *Give Peace A Chance*: *...Ragism, Tagism, / This-ism, that-ism, ism, ism.* ...it might be hard to tell the difference between the word and the ending in *ism*.

6 Results and Discussion

Since the output files of the module provide *graded* tagging, it is somewhat hard to discuss the results in terms of 'hits' or 'misses'. What could be discussed is how often the word class with the *highest* percentage is also the 'correct', word class. Although the module was not conceived as being used as a stand-alone module, it is of a certain interest to check its capabilities a such. Thus, a test run was carried out on 316.599 already tagged – and manually checked – words in the *Stockholm-Umeå Corpus* (Källgren 1991a). The leftmost member in the resulting output lists of ENDTAG were compared to the tags in SUC. The percentages are given in table 5.

TABLE 5 : Figures indicating the percentages of right tagging of words for different word classes.

WORD CLASS	PERCENTAGE
Infinitival marker	100
Nouns	93
Verbs	93
Prepositions	82
Adjectives	78
Adverbs	78
Conjunctions	69
Proper nouns	66
Pronouns	63
Interjections	37
Numerals	26
Abbreviations	16

One interesting feature of ending-list based tagging is the method's inherent capabilitites regarding the tagging of *new* words (cf. Greene & Rubin 1971). Since word formation obeys morphological rules, one may predict that neologisms and inflected loan words should be given rather accurate tags by the module.

One could also point out that one of the contributions of this work is the actual ending lexica *per se*. These have not been scrutinised in detail, but could presumably provide interesting information if studied.

Another point worth making is the module's limitations. Primo, it works on a *brute force* basis, rather than with linguistic *finesse*. The fact that it is not based on grammatical or morphological descriptions or models of Swedish, precludes generation, whence it follows that the module is not bi-directional, a lack we will have to make do with if we want to be able to handle foreign entries. Secundo, as already pointed out, the ending information in the ending lexica is perforce dependent upon the material on which they are based (in this case NFO). Tertio, tagging is graded. If an unambiguous tagging is desired, the module must succeed at lengths greater than (in most cases) three to four letters.

As a final remark, it could be said that no *one* tagging strategy, hitherto, has been able to solve this task fully. A combination of several different methods might increase success rates. A combination of a lexically based method (SWETWOL) with a statistically based method (ENDTAG), disambiguated by a module like the one described by Eriksson (1992) could enhance success rates in automatic word class recognition.

References

Allén, Sture. 1970. *Nusvensk frekvensordbok baserad på tidningstext (Frequency dictionary of present-day Swedish), 1. Graphic Words, Homograph Components, 2. Lemmas, 3. Collocations, 4. Morphemes, Meanings*. Almqvist & Wiksell International, Stockholm.

Blåberg, Olli. 1984. *Svensk Böjningsmorfologi. En tvånivåbeskrivning*. Unpublished Master's Thesis, Department of General Linguistics, University of Helsinki.

Brodda, Benny. 1979. *Något om de svenska ordens fonotax och morfotax: iakttagelser med utgångspunkt från experiment med automatisk morfologisk analys*. PILUS 38, December 1979, Stockholm University.

Brodda, Benny. 1983. *An Experiment with Heuristic Parsing of Swedish*. Stockholm University, Dept. of Linguistics.

Cedwall, M. 1977. *Semantisk analys av processbeskrivningar i naturligt språk*. Linköping Studies in SCIENCE AND TECHNOLOGY, Dissertations No. 18.

Church, Kenneth Ward. 1988. *A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text*. In *Proceedings of the Second Conference on Applied Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pp. 136–143.

Cutting, Douglas. 1994. *Porting a Stochastic Part-of-Speech Tagger to Swedish*. In Eklund (ed), *Nodalida '93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3–5 June 1993*. Stockholm.

DeRose, Steven. 1988. *Grammatical Category Disambiguation by Statistical Optimization*. In COMPUTATIONAL LINGUISTICS 14:1.

Eklund, Robert. 1994. (ed). *Nodalida '93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3–5 June 1993*. Stockholm.

Eklund, Robert. 1993. *A Probabilistic Word Class Tagging Module Based On Surface Pattern Matching*. BA Thesis, Dept. of Linguistics, Stockholm University.

Eeg-Olofsson, Mats. 1991. *Word-Class Tagging, Some computational tools.* Ph. D. Thesis including the following papers: *i. A probability model for computer-aided word-class determination, ii. Software Systems for Computational Morhpology – An Overview, iii. A morphological Prolog system for Swedish based on analogies, iv. Probabilistic word-class tagging of a corpus of spoken English.* Gothenburg University, institutionen för språkvetenskaplig databehandling.

Ejerhed, Eva and Hank Bromley. 1986. *A Self-extending Lexicon: Description of a World Learning Program.* in Karlson 1986, pp. 59–70.

Ejerhed, Eva. 1988. *Finding Clauses in Unrestricted Text by Finitary and Stochastic Methods.* In *Proceedings of the Second Conference in Applied Natural Language Processing*. Austin, Texas.

Eriksson, Gunnar. 1992. *Att två ord för att få en tolkning*. Dept. of Linguistics, Computational Linguistics, Stockholm University.

Garside, Roger. 1987. *The CLAWS word-tagging system.* in Garside, Leech and Sampson (1987), chapter 3.

Garside, J. L. and Geoffrey Leech. 1982. *Grammatical Tagging of the LOB Corpus: General Survey.* in Johansson 1982.

Garside, Roger, Geoffrey Leech and Geoffrey Sampson. 1987. *The Computational Analysis of English: a Corpus-Based Approach.* Longmans, London.

Greene, Barbara B. and Gerald Rubin. 1971. *Automatic Grammatical Tagging of English.* Providence, R. I., Department of English, Brown University.

Hammarberg, Björn. 1966. *Maskinell generering av böjningsformer och identifikation av ordklass*, pp. 59–70 in *Förhandlingar vid sammankomst för att dryfta frågor rörande svenskans beskrivning*. Sture Allén (ed.), Gothenburg University.

Hellberg, Staffan. 1978. *The Morphology of Present-Day Swedish.* DATA LINGUISTICA 13, Sture Allén (ed.), Department of Computational Linguistics, University of Gothenburg, Almqvist & Wiksell International, Stockholm.

Johansson. Stig. 1982. *Computer Corpora in English Language Research.* Norwegian Computing Centre for the Humanities, Bergen.

Johansson, Stig and Mette-Cathrine Jahr. 1982. *Grammatical Tagging of the LOB Corpus: Predicting Word Class from Word Endings.* In Johansson 1982.

Karlsson, Fred. 1986. Papers from the Fifth Scandinavian Conference on Computational Linguistics. Helsinki, December 11–12, 1985. Publications No. 15, Department of General Linguistics, Helsinki University.

Karlsson, Fred. 1990. *Constraint Grammar as a Framework for Parsing Running Text.* In Hans Karlgren (ed.), *Papers presented to the 13th International Conference on Computational Linguistics*, vol. 3, pp. 168–173, University of Helsinki, Department of General Linguistics.

Karlsson, Fred. 1992. *SWETWOL: A Comprehensive Morphological Analyser for Swedish.* pp. 1–45, NORDIC JOURNAL OF LINGUISTICS, Volume 15, Number 1, Scandinavian University Press.

Kiefer, Ferenc. 1970. *Swedish Morphology*. Skriptor, Stockholm.

Koskenniemi, Kimmo. 1983a. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Department of General Linguistics, University of Helsinki, Publication No. 11.

Koskenniemi, Kimmo. 1983b. *Two-Level Morphology for Word-Form Recognition and Production*. Department of General Linguistics, University of Helsinki, Publication No. 13.

Källgren, Gunnel. 1991a. *Storskaligt korpusarbete på dator. En presentation av SUC-korpusen*. In SVENSKANS BESKRIVNING 18, Lund University Press.

Källgren, Gunnel. 1991b. *Making maximal use of surface criteria in large-scale parsing: the MorP parser*. In Pilus 60, Dept. of Linguistics, Stockholm University.

Källgren, Gunnel. 1991c. *Parsing without lexicon: the MorP System*. European Chapter of the Association for Computational Linguistics, Berlin.

Lennon, John. *Give Peace A Chance*. Northern Songs, 1969.

Liberman, Mark Y. and Kenneth W. Church. 1992. *Text Analysis and Word Pronunciation in Text-to-Speech Synthesis*. In S. Furui and M. M. Sundhi (eds.), *Advances in Speech Technology*, Marcel Dekker, New York.

Linell, Per. 1972. *Remarks on Swedish Morphology*. Ruul 1, Department of Linguistics, Uppsala University.

Linell, Per. 1976. *On the Structure of Morphological Operations*. In LINGUISTISCHE BERICHTE 44/76, pp. 1–29.

Magnberg, Sune. 1991. *A Rule-Based System for Identifying Major Syntactic Constituents in Swedish*. Department of Linguistics, Stockholm University.

Marshall, Ian. 1987. *Tag selection using probabilistic methods*. In Garside, Leech and Sampson (1987), chapter 4.

Pitkänen, Kari. 1992. *SWETWOL / Major Changes in 1992*. Research Unit for Computational Linguistics, University of Helsinki.

Rankin, Ivan. 1986. *SMORF – an Implementation of Hellberg's Morphology System*. Department of Computer and Information Science, Linköping University.

Sampson, Geoffrey. 1991. *Probabilistic Parsing*. In Svartvik, Jan (ed.), *Directions in Corpus Linguistics 82*, Stockholm, 4–8 August 1991, Mouton de Gruyter, Berlin.

Samuelsson, Christer. 1994. *Morphological Tagging Based Entirely on Bayesian Inference*. In Eklund (ed), *Nodalida '93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3–5 June 1993*. Stockholm.

On Implementing Swedish Tense and Aspect

Björn Gambäck
Stockholm

Abstract

The paper addresses the problems encountered when implementing a system for the treatment of Swedish tense, mood and aspect. The underlying theory suffered from the same shortcomings as do most implementable linguistic theories: it was designed for English. To extend it to Swedish some aspects of the theory, but also the implementation had to be generalized to allow for a system which treats Swedish verb-phrase syntax and semantics in a uniform way. This paper is concentrated on how this treatment actually has been implemented in a large-scale natural-language processing system.

1. Introduction

The theories for the treatment of tense and aspect phenomena in various languages are so many that it almost seems like any linguist (or at least any slavist and aspectologist) worthy of the name ought to have her own; however, not many of these theories have had any major impact on computational linguistics, possibly partially because most natural language systems are written for English where the "problems" caused by tense and aspect (at least at the surface) are not so complicated as to warrant the spending of too much development time and partially because most NL-systems simply do not have a life-span long enough for the issue to reach the implementation agenda.

The paper concentrates on the problems encountered when implementing a system for the treatment of Swedish tense, mood and aspect. The underlying theory was designed for English, so some aspects of it had to be generalized in order to extend it to Swedish. However, most of the treatment was still relevant, given that Swedish is not a language where aspect is too complicated, either. This objection is not as serious as it may sound, since the generalized version of the theory also should be able to treat such aspectual languages as Polish and Russian: a claim which however is not defended in the paper, neither a main point of it. A full-detailed discussion of Swedish verb-phrases in general will also be left aside; they are treated in length by other authors, for example Andersson (1977) and Tjekalina (1991).

It should be noted that in the title of the paper, as well as in the text following, the term "tense-aspect information" (or just T/A) and the like will be taken to refer to a range of phenomena that in principle just have a few properties in common, namely that they are (to certain extent) visual in the surface syntax, but in general have to be interpreted at a deeper semantic level. This is mostly, but not necessarily, because they are discourse rather than sentence related.

Apart from the obvious "tense" and "aspect" from the title, another category of the same kind that immediately springs to mind is "mood", but this paper will also include "voice" as belonging to the same broad type, while it (admittedly rather arbitrarily) will exclude for example "negation". It will also exclude phenomena which might have some aspects in common with the ones mentioned, but which is outside the current state of the art, for example "metaphor". The term "tense-aspect" used here is for the lack of a better one, but should thus not be taken as defining just those two categories, or defining one category of those two concepts. Or indeed as defining any categories at all, reflecting what (Chatterjee, 1982, p 337) refers to as the *categorial paradox*:

> "... a semantic or grammatical category is one only in relation to other 'neighboring' categories, yet we have not succeeded in isolating or defining a tense/aspect category (giving it *gesamtbedeutungen*) in the most studied languages. (...) Further, even if we did, our category would be language-specific, and so would its interaction with other categories of the language. (...) Aspect being to some extent notational (i.e., an investigative concept) in all languages, a universalist pinning down of the category is impossible."

The phenomena lumped together here as "T/A" will be more or less manifest in different languages, so for example for Swedish "tense" is a rather obvious candidate for discussion, and so is "mood", while "aspect" seems to be far more controversial. There has even been claims that there is no such thing as aspect in Swedish (Jordan Zlatev, personal communication, 1993). The following text will hopefully show that such claims are not to be taken too seriously. A more relevant question is raised by (Gawrónska, 1992), who argues that aspect in English and Swedish is in practice not as relevant as the introduction of (or lack of introduction of) the definite article, thus giving the definite article in these languages a role rather complementary to that of aspect in for example Russian and Polish.

The rest of the paper is outlined as follows: the next section will introduce the natural-language processing system used, the Swedish Core Language Engine. Section 3 contains a discussion of the treatment of verb-phrase syntax and semantics in the grammar, while Section 4 gets

into some specific details of the implementation of the tense and aspect system.

2. The Swedish Core Language Engine

The Swedish Core Language Engine (S-CLE, Gambäck & Rayner, 1992) is a general-purpose natural-language processing system for Swedish which was developed from its English counter-part, the SRI Core Language Engine (CLE, Alshawi, 1992). The system is written purely in Prolog and based on unification as the main mechanism. The S-CLE is equipped with a sizable grammar for Swedish covering most common constructions in the language, including: questions (yes/no- and wh-), topicalized clauses, imperatives, passives, relative clauses, negation, cleft constructions, ellipsis, conjunction, noun-phrase and verb-phrase modification by preposition-phrases, adjectives and adverbs, various kinds of complex determiners, proper names, codes, dates and times, possessive constructions and about fifty different kinds of complements to verbs and adjectives. The grammar formalism is a feature-category type with declarative bidirectional rules, that is, the grammar can be used both for language analysis and for generation (it is just compiled in different ways depending on in what direction it is to be used)

A natural-language sentence that is input to the S-CLE is analysed to a logical-form like representation called QLF, Quasi-Logical Form, a conservative representation of the meaning of an input sentence based on purely linguistic evidence. The English and Swedish versions of the CLE have been used together to form a bidirectional translation system, transfer taking place at the QLF level (Alshawi *et al*, 1991), but the QLF can also be used as the basis for further (deeper-level) processing. Deriving a QLF from an NL-sentence involves the processing steps shown in Figure 1.

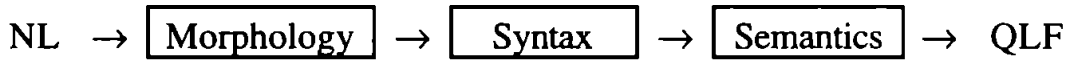


Figure 1: The analysis steps of the S-CLE

First morphological analysis locate the correct word-senses and inflected forms of the input string, then syntactic parsing and (compositional) semantic analysis derive the parse tree(s) and its corresponding QLF-representation. Later processing steps (e.g., reference resolution and quantifier scoping) will try to further instantiate the QLF, aiming at deriving a "true" logical form (context and application dependent).

The S-CLE lexicon

The lexicon of the S-CLE is rather elaborate, with the lexical entries containing information both about morphological inflection patterns, syntactical subcategorization patterns and some semantical restrictions on the type of arguments. The lexicon form chosen for verbs is the imperative (rather than the "normal" dictionary form, the infinitive) since this form constitutes the stem of most other inflections, so a verb like *gilla* ("like") can be defined as being of the first declension, subcategorizing for an NP (i.e., being a transitive verb) and having the restrictions that it is a physical, nonpropositional, located event obtaining between a human subject and an object which basically can be anything, thus:

```
lr(gilla,v_subj_obj(v1,n),gilla_3p).
sor(gilla_3p,[[physev,nonprop,located],[human],
[]]=>[prop]).
```

where `gilla_3p` is the semantic constant used to identify the verb *gilla*, `v_subj_obj` is the pattern of a regular transitive verb which passivizes, `v1` the first declension of a non-deponent verb (`n`), and the Prolog-type list at the end of the second line introduces the restrictions on the event itself, the subject, the object and finally on the overall statement produced (a proposition, `prop`).

This *implicit* verb entry is in turn expanded out automatically using explicit paradigm (prototype) syntactic and semantic entries for verbs of the `v_subj_obj` (transitive) type. Schematically,[1] the syntactic one is

```
paradigm('verb_subj_obj'(Conjugation,Deponent),
    v:[@conjugation(Conjugation), @deponent(Deponent),
       vform=impera, gaps=Gaps,
       subcat=[np:[gaps=Gaps]]]).
```

Where the notation is to be interpreted so that an element belonging to the `v_subj_obj` paradigm is a verb (i.e., has the *category name* `v`) which has a list of *feature-value pairs* associated with it. So, for example `subcat` (for subcategorization) is a feature of the verb having a value which in turn is a list consisting of just one element, an `np` (the object). That NP also has a list of feature and values; some of the values are unified with the corresponding values on the verb, the `gaps` feature (which holds a list of empty constituents found within the phrase) for one is thus shared between the verb and its object.

---

[1] All lexicon entries and grammar rules exemplified in this paper are simplified. Features and other information not relevant for the discussion at hand have been removed to improve readability.

Of the other features, the `vform` specifies that the verb-form found in the lexicon is the imperative, while `@conjugation` and `@deponent` are macros (introduced by the `@` operator). a phenomena to be further discussed below. For now it is enough to note that they instantiate some morphological features with values following the `v1` and `n` declarations from an implicit entry as the one for *gilla* shown above.

3. Verb-phrase syntax and semantics

As indicated by the title, this paper is mainly devoted to how a theory of Swedish verb-phrases actually was implemented. The next section will go into the some of the more specific implementation details, but we will start out with a discussion of the overall verb-phrase grammar rules, illustrated by the how the rules actually have been implemented.[1]

Syntax

From a theoretical view-point, the aim of this work is to establish a *uniform* treatment of Swedish verb-phrases of any kind, be it with or without modification or with different types of verbal complements. In order to reach that goal, a syntactic grammatical rule as the following is central:

```
vp:[gaps=G, vform=Vf]
 -->
v:[gaps=G, vform=Vf, subcat=Complements]
 +
Complements
```

This rule should be read so that the feature `subcat` on a verb in effect specifies the number of constituents to be found in a verb-phrase, since the rest of the right-hand side of the rule only is specified as being something which is unified with the value `Complements`. As we saw already in the previous section, the value of `subcat` for a particular verb is specified in its lexical entry, so if the verb found is "gilla", its subcategorization will be instantiated to be an NP.

Thus the above rule actually is a rule *schema*, replacing a multitude of verb-phrase formation rules which could have been written explicitly. The rule is both elegant in its simplicity and useful in that it helps in avoiding redundancy in the grammar and saves the grammar writer time:

[1]For a more consistent and implementation independent description of the theory, see Gambäck (1993).

the S-CLE contains some 50 different types of verbs all of which are treated with the same schema; writing a separate rule for each verb-type would of course have been possible, but hardly feasible.

Semantics

Looking at the semantic side the situation gets a bit more complicated; while main verbs still can be treated easily by a verb-phrase formation rule parallel to the single syntactic one, care has to be taken while treating auxiliaries.

The main verb case simply adds semantic information to the syntactic rule:

```
(V,
 vp:[@shared_tense_aspect(T,U)])
 -->
(V,
 v:[@shared_tense_aspect(T,U), arglist=Complements])
 +
Complements
```

Here, each constituent of the rule is a pair `(QLF,Category)`, where the `Category` holds the same information as in the syntactic case (i.e., consists of the category name followed by a list of feature-value pairs), while the QLF is the semantic information, a logical form fragment. Thus `arglist` is a feature performing exactly the same function as `subcat` above, but with the semantic information added. The value `V` for both the verb's and the verb-phrase's logical form indicates that the verb's semantic interpretation is passed up (by unification) to become the interpretation of the entire verb-phrase.

`@shared_tense_aspect` is a macro which (also by unification) passes the tense-aspect information up from the main verb to the verb-phrase. The rule does not explicitly take care of the semantic interpretations of the complements: this information is, however, simply unified into the verb's semantics in the lexicon.

Tense auxiliaries

Auxiliaries that change the tense of the verb-phrase (e.g., to past as *hade*, "had", or future as *ska*, "shall") must be treated separately from the main-verb case. As was shown above, both the semantic interpretation and the tense information for main verbs and for their "mother" verb-phrases are the same; however, in the auxiliary case, the semantic

interpretation of the mother verb-phrase should still be the one of the daughter verb-phrase, but the tense should be taken from the auxiliary, so this case of the rule becomes:

```
(V,
 vp:[@shared_tense_aspect(T,U)])
 -->
(empty,
 v:[@shared_tense_aspect(T,U), arglist=(V,vp:[])])
 +
(V,
 vp:[])
```

Where the auxiliary is shown to be a verb which subcategorizes for a verb-phrase with the same semantics as the mother VP, but itself carrying no semantic information proper (i.e., the QLF of the V is empty). The tense-aspect information of the daughter verb-phrase is left out from the rule, indicating that it should not influence the T/A of the mother; however, it may, but this should be treated in the lexical entry for the auxiliary.[1]

Modal auxiliaries

Modal auxiliaries complicate the picture somewhat: we need to treat two cases, one for finite and one for non-finite (i.e., infinite plus supine) verb forms, the difference being that the former (in Swedish, but not for example in English) can modify other modals as in a sentence like

Jag skulle vilja kunna flyga. *"I would like to be able to fly"*
(lit. "I should want could fly")

In examples like this one (where at least the *skulle vilja* construction is very common), finite modals behave quite a bit like tense auxiliaries; they do not affect the semantic content as such, but rather the modal information, which (as we shall see in the next section) can be taken to be part of the tense-aspect information, so that finite modals actually can be treated with exactly the same case of the verb-phrase formation rule as tense auxiliaries. Non-finite modals on the other hand behave just like ordinary verbs in the effect they have on the semantic interpretation

[1]It should be noted that in the actual implementation the auxiliary QLF may be non-empty (it can contain imformation about verb-modification by so called "mobile adverbs" – e.g., the negation marker *inte)* and is thus taken care of properly, anyhow. A full description of the implementation of negation would, however, hardly add to the present discussion and is thus (and for space considerations) left out here.

proper. They are thus treated with the same rule instance as the main verbs.

4. Implementational aspects

For an inflectional language like Swedish, where most of the tense and aspect information can be found in the suffix of the main verb, it is natural to view the tense-aspect information as forming a function of the affix. For the actual implementation, we represent it in the compositional semantics as a functor

```
verb(Tense,Aspect,Action,Mood,Voice)
```

where the information is filtered up from the verb-affix to the verb phrase. The arguments of `verb` will be explained further on; first, however, we should note that the choice of this functor is rather (but not completely) arbitrary. For a language such as Finnish where the aspect information is carried on the object rather than the predicate some other functor name of course should be chosen. Also, the number of arguments and their interpretation could certainly vary between languages (or between linguists and linguistic theories treating the same language), but in general we need a strategy as the one suggested by (Alshawi & Crouch, 1992): first a way to packet the tense-aspect information declaratively in the compositional semantics and then a way to unpack this information later on to determine the implicit points in time, etc., not shown in the surface form of the sentence. This "packaging" is the main function of the functor `verb`, whose arguments are in order:

`Tense`	the relation of the event to the present time of the speaker: past, present, or future.
`Aspect`	the relation of the event to the action time of the verb: perfective or imperfective.
`Action`	the way in which an event happens: progressive or non-progressive.
`Mood`	the speaker's view on the event: a modal, imperative, etc.
`Voice`	the relation of the meaning of the verb to the subject: active or passive.

Morphology

As noted above, the lexicon form chosen for the Swedish verbs is the imperative, since this form constitutes the stem of most other inflections. For tense and aspect purposes, however, the imperative is a bit peculiar:

it stands almost on the side of the entire tense-aspect system. Thus the lexicon contains stems for which the T/A information is only partially instantiated (viz., `verb(no,Pf,Pg,imp,A)`). The (normally) full instantiation is obtained by the inflection in morphology rules as the following

```
(@verb_semantics(Sense,TA,Event,Args),
 v:[@shared_tense_aspect(TA,U)])
  -->
(@verb_semantics(Sense,_,Event,Args),
 v:[])
  +
(suffix,
 suffix:[@shared_tense_aspect(TA,U)])
```

which shows that the mother verb is formed by adding a suffix to the daughter verb (i.e., the stem form). Just as in the sematic grammar rule above, each of the three components of the rule consists of two parts: the semantic information (here, a QLF fragment) and the category name followed by a list of feature-value pairs. The variables `TA` and `U` together carry the tense-aspect information: `TA` holding the T/A information proper, while `U` keeps track of the as-of-yet uninstantiated information. The T/A information from the suffix is passed up to the inflected verb by unification. This is also the only (semantic) information added by the suffix; the other parts of the mother-verb semantics come from the daughter, i.e., the sense name `Sense` (as `gilla_3p` above), the variable `Event` representing the event itself and the list of the verb's arguments' (e.g., objects') QLF-fragments, `Args`.

Suffixes

An example of a suffix entry is the one for the ending "-r", which is added to the stem of some verbs to form the present tense:

```
sense('-r',
suffix:[@pres_mainv(TenseAspect),
        vform=(fin/\present),
        synmorphv=(1\/3\/43),
        lexform='-r'],
suffix).
```

The value of the feature `vform` indicates that the inflected verb produced will be in present and finite form,[1] while the feature `synmorphv` restricts

[1]The symbols "/\" and "\/" functions as the normal logical "and" and "or" operators, respectively.

the syntactic morphological categories (i.e., verb declensions) for which the ending "-r" is appropriate. Here, they are verbs belonging to the 1st and 3rd declension as well as those belonging to the 4th declension., 3rd subgroup.

The first `'-r'` is the sense name of the suffix, while the second (the value of the feature `lexform`) is its actual realization in the surface string. In the same fashion, the first `suffix` is the rather arbitrary name of the suffix' category in the grammar, while the second holds the semantic content (the logical-form fragment) obtained from the suffix. The latter is in reality none at all (apart from the T/A information), so the second `suffix` is mainly a place-holder.

For the syntactic analysis of the system implemented, the logical form of the suffix entry could be completely uninstantiated; however, it is worth noting that given that the grammar is bidirectional, we do not want to leave the suffix' semantic content uninstantiated, the generation algorithm used in the S-CLE ("Semantic-Head-Driven Generation", Shieber *et al*, 1990) actually requiring all logical-form fragments to be instantiated.

Macros

For the purposes of this paper, the most important part of the suffix entry above is `@pres_mainv(TenseAspect)`, which actually is a macro call. The full macro definition used for present tense main verbs is

```
macro(pres_mainv([pres,no,no,no,y]),
      [tense=pres, perf=no, prog=no, modal=no, active=y,
       uninstTA=1([])]).
```

which shows that the tense-aspect information in reality is carried by a whole group of feature-value pairs, which together hold the same information as in the previously described `verb` functor. Thus the first feature-value pair indicates the present tense, the second shows the imperfective aspect and the third the non-progressive action. No specific modal information is added by this macro, but the voice of the verb has to be active. Since all the other five T/A features are instantiated, the final `uninstTA` feature just holds an empty (Prolog-type) list. The `@pres_mainv` macro is complemented with a number of macros for all the different inflectional forms of verbs and for both main verbs and auxiliaries; the main ones are listed in the appendix.

In the same fashion, the entries `@shared_tense_aspect` and `verb_semantics` in the verb-affixing rule above are also macro calls, their full definitions being

```
macro(shared_tense_aspect([T,Pf,Pg,M,A],U),
      [tense=T, perf=Pf, prog=Pg, modal=M, active=A,
       uninstTA=U]).

macro(verb_semantics(Sense,[T,Pf,Pg,M,A],Event,Args),
      @form(verb(T,Pf,Pg,M,A), Event,
            P^[P,[Sense,Event|Args]], _)).
```

The first macro is just a convenient way to address all the T/A features at once, while the second one gives the current version of the semantics chosen for event verbs. It is out of the scope of the present chapter to go into the details of the QLF formalism (the interested reader is referred to Alshawi, 1992), so we only note that the semantics of the verb is a `form` which includes the `verb` functor as defined above, the `Event` variable and the actual (body) semantics of the verb which is a lambda-abstraction[1] with the `Sense` name as a function whose parameters are the `Event` variable followed by the logical forms of the complements (`Args`).

[1] `^` is a type-writer version of the more common λ, so `P^[P,Q]` is equivalent to λP.Q(P).

References

Alshawi, Hiyan (ed.). 1992. *The Core Language Engine*. The MIT Press, Cambridge, Massachusetts.

Alshawi, Hiyan and Richard Crouch. 1992. *Monotonic Semantic Interpretation*. pp 32–39, *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, Newark, Delaware.

Alshawi, Hiyan, David M. Carter, Björn Gambäck and Manny Rayner. 1991. *Translation by Quasi Logical Form Transfer*. pp 161–168, *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, University of California, Berkeley, California.

Andersson, Erik. 1977. *Verbfrasens struktur i svenskan: en studie i aspekt, tempus, tidsadverbial och semantisk räckvidd*. Doctor of Philosophy Thesis, Åbo Akademi, Åbo, Finland (in Swedish).

Chatterjee, Ranjit. 1982. *On Cross-Linguistic Categories and Related Problems*. pp 335–345, *Tense-Aspect: Between Semantics & Pragmatics*. Paul J. Hopper, editor, John Benjamins, Amsterdam, Holland.

Gambäck, Björn. 1993. *Towards a Uniform Treatment of Swedish Verb Syntax and Semantics. Proceedings of the 16th Scandinavian Conference of Linguistics and the 8th Conference of Nordic and General Linguistics*, University of Gothenburg, Gothenburg, Sweden.

Gambäck, Björn and Manny Rayner. 1992. *The Swedish Core Language Engine*. pp 71–85, *Papers from the 3rd Nordic Conference on Text Comprehension in Man and Machine*, Linköping University, Linköping, Sweden. Also available as SICS Research Report, R92013, Stockholm, Sweden.

Gawrónska, Barbara. 1992. *An MT Oriented Model of Aspect and Article Semantics*. Doctor of Philosophy Thesis, Lund University, Lund, Sweden.

Shieber, Stuart M., Gertjan van Noord, Fernando C.N. Pereira, and Robert C. Moore. 1990. *Semantic-Head-Driven Generation*. pp 30–43, COMPUTATIONAL LINGUISTICS, Volume 16.

Tjekalina, Elena. 1991. *Om kategorierna modus och tempus i nutida svenska*. pp 139–155, SPRÅK & STIL, Volume 1 (in Swedish).

Appendix: Verb inflection macros

Imperatives (no tense)

```
macro(imperative([no,Pf,Pg,imp,A]),
      [tense=no, perf=Pf, prog=Pg, modal=imp, active=A,
       uninstTA=1([A,Pf,Pg])]).
```

Main verbs and VP complements

```
macro(pres_mainv([pres,no,no,no,y]),
      [tense=pres,perf=no,prog=no,modal=no,active=y,
       uninstTA=1([])]).
macro(past_mainv([past,no,no,no,y]),
      [tense=past,perf=no,prog=no,modal=no,active=y,
       uninstTA=1([])]).

macro(inf_mainv([T,no,no,M,y]),
      [tense=T,perf=no,prog=no,modal=M,active=y,
       uninstTA=1([T,M])]).
macro(perfp_mainv([T,yes,Pg,M,A]),
      [tense=T,perf=yes,prog=Pg,modal=M,active=A,
       uninstTA=1([A,T,Pg,M])]).
macro(perfp_intrans([past,yes,no,M,y]),
      [tense=past,perf=yes,prog=no,modal=M,active=y,
       uninstTA=1([M])]).
macro(perfp_transevent([past,yes,no,M,n]),
      [tense=past,perf=yes,prog=no,modal=M,active=n,
       uninstTA=1([M])]).
macro(perfp_transstate([pres,yes,yes,M,n]),
      [tense=pres,perf=yes,prog=yes,modal=M,active=n,
       uninstTA=1([M])]).
macro(presp_mainv([pres,Pf,yes,M,y]),
       [tense=pres,perf=Pf,prog=yes,modal=M,active=y,
       uninstTA=1([Pf,M])]).

macro(pass_mainv([T,Pf,Pg,M,n]),
       [tense=T,perf=Pf,prog=Pg,modal=M,active=n,
       uninstTA=1([T,Pf,Pg,M])]).

macro(supinev([T,yes,no,M,y]),
      [tense=T,perf=yes,prog=no,modal=M,active=y,
       uninstTA=1([T,M])]).
```

Modals (depends on the verb-form – the second argument)

```
macro(modal_tense_aspect(M,no,[no,Pf,Pg,M,A]),
      [tense=no, perf=Pf, prog=Pg, modal=M, active=A]).
macro(modal_tense_aspect(M,pres,[pres,Pf,Pg,M,A]),
      [tense=pres, perf=Pf, prog=Pg, modal=M, active=A]).
macro(modal_tense_aspect(M,past,[past,Pf,Pg,M,A]),
      [tense=past, perf=Pf, prog=Pg, modal=M, active=A]).
macro(modal_tense_aspect(M,inf,TenseAspect),
      [@inf_mainv(TenseAspect)]).
macro(modal_tense_aspect(M,supine,TenseAspect),
      [@supinev(TenseAspect)]).
```

Reasoning with a Domain Model

Steffen Leo Hansen
København

Abstract

A domain model is a knowledge base containing both domain specific and world knowledge. You may take the domain model to be both a universe of interest and a universe of problems. As a universe of interest the model contains all the information relevant and necessary for the intended use of the model as a store of information, a knowledge base. As a universe of problems the model represents a problem space and the relevant and necessary inferential tools needed by the model for the intended use as a problem-solving mechanism. Problem solving, in this case, means finding answers to queries about domain-specific knowledge. In this paper we shall discuss some fundamental problems related to the construction and use of a domain model called FRAME_WORLD.

1 Introduction

The domain model presented in this paper is thought of as a module in a knowledge system using a natural language interface to retrieve information in a database. As a module of the overall system the domain model serves the purpose of evaluating user queries with respect to domain-specific knowledge and that of generating appropriate arguments for subsequent SQL commands.

The domain-specific knowledge of the model comprises facts about domain-specific entities, their properties and possible relations between these entities, whereas world knowledge comprises information not represented in the domain but necessary for the model as a problem solver, e.g heuristics, general rules about causal or spatial relations and the like. The relevant rules and facts are used by an inference machine, not only to state information already explicitly at hand, but also to support the system in making implicit domain-specific knowledge explicit.

The knowledge representation schemes used in the domain model presented are a semantic network, frames, so-called model predicates and heuristics. In the following sections we shall present and discuss the implementation and intended use of FRAME_WORLD, first of all problems of reasoning with inferential structures given by virtue of a specific representation scheme.

2 The domain model

The knowledge in the domain model and the structure of the model depends entirely on the purpose it serves. As already mentioned, the model is thought of as a kind of filter, a means of controlling and checking the knowledge represented in the queries posed to the system by the user and either reject the query as a senseless one or compute and generate one or more arguments to be used in an SQL command to retrieve the required information.

The basis for building and constructing the domain model, therefore, is the set of possible and allowed queries to the system, like for instance: who is the colleague of X, how many people are employed in the sales department, or how much is the salary of X?

To answer questions of this sort you have to have access to both domain-specific and world knowledge. To know whether X and Y are colleagues, you have to have some rule telling you what it means to be colleagues and some means of checking if X and Y in our domain actually do fit this definition. If this is not the case, we do not want the system to react by simply answering 'No', but an output like: 'X is a customer, and Y is an employee'.

To this purpose the domain model needs information about entities and relations in the domain and in the world outside the domain as well as some kind of machinery that uses this knowledge for information retrieval and query answering.

Entities and relations between entities inside and outside the domain are represented as a network of nodes and links. The nodes in the net are conceptual entities, the knowledge primitives of the model. The links in the net either relate concepts as conceptual entities to each other or concepts as arguments of a semantic predicate to each other. The former kind of links are called conceptual links, the latter, the relational links, are called role relations.

The description of a node comprises both the set of incoming and outgoing links as structural information about the concept as well as the set of conceptual features characterising the specific concept in question. This description is implemented as a frame. The role relations, too, are mapped into frames such that for each concept and for each role relation in the net there will be a frame with the same name as a description of that particular knowledge unit.

3 Representation schemes

3.1 The network

Using a network for knowledge representation in a domain model seems obvious. Knowledge pictured as a network makes it possible to represent a conceptual hierarchy as a nice structure of nodes and links representing all available information immediately ready for use. All you need is the right algorithm extracting the information or transfering information from more to less general nodes of concepts. It seems to reduce knowledge retrieval to simply finding the right node or nodes and the right path connecting two or more nodes with each other.

It is, however, not as simple as that. Reasoning with a network presupposes a well-defined syntax and semantics of the net as discussed and emphasized in several papers (e.g. Woods 1987 & 1990, Thomasson/Touretzky 1991).

The idea of using networks as a representation scheme is that of making information attached to some node X accessible for other nodes connected to X. This property of a network is the fundamental principle of *inheritance* and *path-based reasoning*, and probably the most important reason for the popularity of this way of organizing knowledge and using a network as an inferential tool.

Inheritance means that information kept in a node X is inherited by a node Y if Y is connected to X. Path-based reasoning means infering conclusions by way of finding a correct path through the net, in most cases simply by computing the transitive closure of a set of links in the net (Thomason/Touretzky 1991:239). Let us illustrate these principles using a fragment of the domain net.

In this fragment (fig. 1) we have two different kinds of conceptual links labelled *ako* and *apo*, *a kind of* and *a part of* , and a relational link labelled *works_in* stating that an employee works in a department. Both the *ako* and *apo* relations are transitive relations, and without any further restrictions one might infer that a subordinate is a kind of legal person.

This conclusion is derived by simply computing the transitive closure of the links involved, but it not a valid one because it is based on two different and incompatible concepts: the concept *firm* as a subconcept of the superconcept *legal person*, a generic concept defined by a set of conceptual features, and the concept *firm* defined by the set of parts constituting it as a whole, one of which is a department.

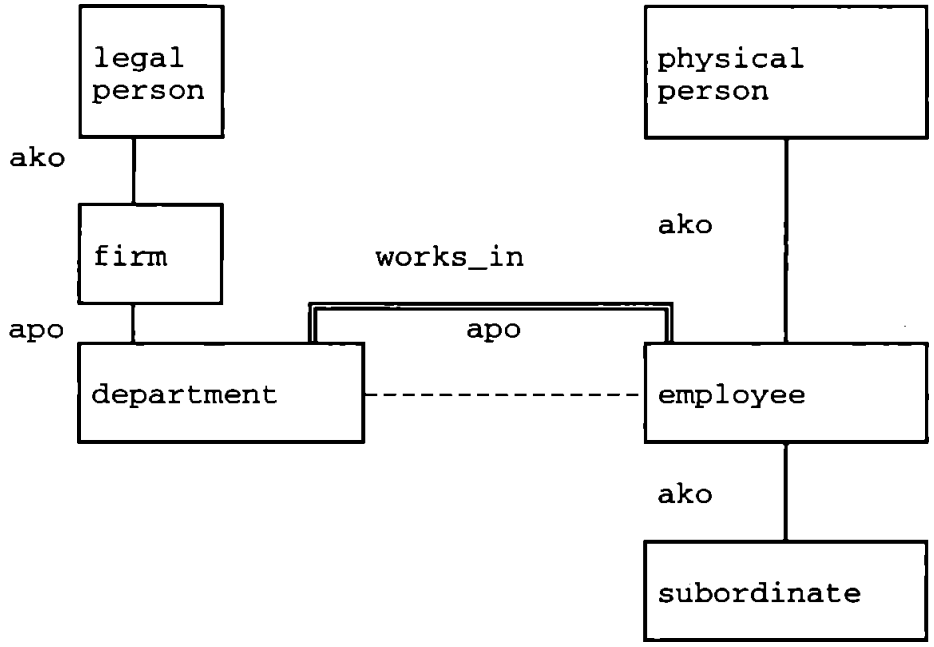


FIG. 1

To avoid conclusions like the one just presented we have to define both the syntax and the semantics of the net. The net in FRAME_WORLD consists of the following components:

(1) A set of nodes F = $\{C_{f1},...,C_{fn}\}$, generic concepts defined by a set of conceptual features,

(2) A set of nodes P = $\{C_{p1},...,C_{pn}\}$, part-whole concepts defined by a set of parts,

(3) A link type: L_{ako}, labelled 'ako',

(4) A link type: L_{apo}, labelled 'apo', and

(5) A link type: L_{role}, labelled with the name of the role.

A well-formed link in the net is a triple of one of the following types:

(6) $<L_{ako},C_{fi},C_{fj}>$

(7) $<L_{apo},C_{pk},C_{pl}>$

(8) $<L_{apo},C_{fm},C_{pn}>$

A well-formed path in the net is a structure of well-formed links. The interpretation of a well-formed link goes as follows:

(9) $L_{ako}(X)$ = Y: X < Y,
X is a subconcept of the superconcept Y,

(10) Lapo(X) = Y: X ¤ Y,
X is a part of Y.

Using these definitions we can reject the conclusion: a *subordinate* is a kind of *legal person* because the final link of the path: *<L_{ako},C_p,C_f>, the *firm* being a kind of *legal person* is not a wellformed link.

It is easy to see now how the definition of a well-formed link and of a well-formed path at the same time defines the inferential structure of the net as a sequence of well-formed links. The syntax of a well-formed link also defines the syntax of a well-formed query, and the interpretation of a well-formed query is the same as that of a well-formed link.

The link type L_{role} is not part of the inferential structure in the net. This link type is part of the definition of concepts and a means of associating concepts with thematic roles like

(11) $L_{deal\_with}(X)$ = Y: deal_with(X:actor,Y:locus)

3.2 Frames

The network, as demonstrated in the previous section, is a knowledge base mapping a conceptual hierarchy into nodes representing conceptual entities and links representing conceptual relations. These nodes and links are the knowledge primitives in the domain model. In addition, the network also keeps information about role relations associating concepts as arguments of a semantic predicate with thematic roles.

The description of the nodes and the role relations as objects of information is placed in the frames in the model. A structural description of a node comprises all incoming and outgoing labelled links in the traditional slot:filler structure, using the label of a link as slot and the value of a link as filler. The description of a generic concept, further, comprises the conceptual features defining the concept in a slot labelled *attributes.*

For each concept and for each role relation there will be a frame describing the entity in question. Role relations as knowledge objects are treated in the same way as conceptual entities, i.e. as structured objects of a taxonomic hierarchy. Based on the syntax adopted by the project all the

frames in the domain model are represented as Prolog terms like for instance:

```
frame( employee,
        [ ako-[ val physical_person],
          apo-[ val department],
          role-[ val work]
                                    ] ).
```

The concept *employee* is described as a kind of *physical person*, as a part of a *department* and as a valid argument in the role relation *work*.

```
frame( deal_with,
        [ ako-[ val process],
          roles-[ calculate deal_with(X,Y) ]
                                              ] ).
```

The role relation *deal_with* is described as a kind of *process* with a role structure to be computed by the procedure *calculate*. The possible values of X and Y are computed using the so called model predicates.

3.3 Model predicates

Model predicates were introduced by (Henriksen/Haagensen 1991) as a means of checking the validity of types of arguments. Thus the interpretation of the model predicate:

```
deal_with(FIRM,CUSTOMER)
```

defines the valid arguments of the semantic predicate *deal_with* to be of the type FIRM and CUSTOMER.

In FRAME_WORLD we have extended the function of model predicates to also associating types of arguments with thematic roles. In our domain model we have the following three instances of *handle_med* (eng. deal_with):

```
handle_med(actor:firma,locus:kunde)
handle_med(actor:firma,theme:vare)
handle_med(actor:kunde,locus:firma)
```

Instead of having a frame for each reading of the predicate the procedure *calculate* will compute the relevant role structure. The actual use and function of the model predicates will be demonstrated in section 4.4.

3.4 Rules

The core of the domain model as a reasoning system comprises the network and the frames. In addition, the model may use both the model predicates and a set of domain-independent rules as part of an inference procedure. The rules, representing general world knowledge, play a very important role in making implicit domain-specific knowledge explicit defining *where to look* and *what to look for* in the knowledge base.

For the present only three rules have been implemented defining the concepts *superior* and *colleague* and the role relation *an_employee_of.* These rules, however, illustrate the need for and use of world knowledge implemented as rules.

3.5 The inference machinery

The inference machinery of the model is a set of Prolog procedures. The strategy implemented is based on the principle of inheritance and path-based reasoning using build-in facilities of Prolog. The basic operation of the machinery is that of applying the interpretation of a link as a function to a node yielding as value another node. This is not the place, however, to go into details with the inference machinery. Let us, instead, take a look at how the domain model actually may be used and how it functions as a knowledge filter and generator in a question-answering system.

4 Reasoning with the domain model

In this section we shall focus on the intended use of the domain model. For the present, we can only show how to use the network, the frames, the model predicates and the rules as part of a reasoning system. This may, however, give you an idea of the intended performance of the model as a whole

4.1 The frames

The frame structure is utilized in two ways: (a) either to instantiate variables used by the inference machinery with values found in a frame, or (b) to find one or more frames matching a description:

(a) ?- frame(leder,**Slots**).

Slots = [ako-[val ansat], role-[val lede]]

?- frame(lede,**Slots**).

Slots = [ako-[val arbejde],
roles-[calc lede(_8210,_8211)]]

(b) ?- frame(**Name**,[ako-**Ako**,role-**Role**]).

Name = leder
Ako = [val ansat]
Role = [val lede]
...

4.2 The network

As you have probably already noticed, the structure of a frame as a description of a node is an encoded fragment of the network. The inference machinery uses this property of a frame in path-based reasoning. Actually, there is no network explicitly at hand in the domain model, but using the structure of the frames the inference machinery may generate one or more sub-nets computing the transitive closure of a link in the net:

? - get_frame(Name,ako-Ako).

Name = person
Ako = entity

Name = physical person
Ako = person

Name = employee
Ako = physical person
...

?- get_frame(Name,apo-Apo).

Name = department
Apo = firm

Name = employee
Apo = department

Name = manager
Apo = department
...

Generating hierarchies of this kind may at a later time be used as an instrument to check whether some inferred type value, say, *secretary*, is subsumed by some other type value, *employee*, and, consequently, a valid argument of the semantic predicate *work* as in: *work(secretary,department)*.

4.3 Inheritance

Inheritance normally means inheriting properties. This is also true of the domain model although inheritance in this case rather means structure copying (Winston 1974:263). The concept *physical person* in the net is defined by the features: *Navn*, *Adresse* and *CPR*. These features are representend in the corresponding frame in a slot labelled attributes and may be inherited by all subsumed concepts like

```
? - get_frame(sekretær,attr-Attr).

        Attr = [navn:NAVN,adresse:ADRESSE,cpr:CPR]
```

This is also true of role relations as features defining a generic concept:

```
? - get_frame(sekretær,[role-Role,roles-Roles]).

        Role = arbejde
        Roles = arbejde(actor:ansat,locus:firma)
```

In this case the role relation and the role structure is inherited from the superconcept *employee*.

4.4 Model predicates

The model predicates are potential inferential tools, tools to support the inference machinery as a means of controlling types and values instantiated by the inference machinery. These predicates may be used in three different ways:

(1) the procedure *calculate* called in a frame computes all possible role structures of a specific predicate:

```
?- get_frame(handle_med,roles-RoleStr).

        handle_med(actor:firma,locus:kunde)
        handle_med(actor:firma,theme:vare)
        handle_med(actor:kunde,locus:firma)
```

(2) given a specific conceptual entity as argument *calculate* computes the corresponding role structure:

```
? - get_frame(kunde,roles-RoleStr).

        handle_med(actor:firma,locus:kunde)
        handle_med(actor:kunde,locus:firma)
```

(3) the procedure *calculate* computes the role structure inherited from a subsuming argument type:

```
? - get_frame(sekretær,roles-Roles).

        arbejde(actor:ansat,locus:firma)
```

4.5 The rules

The rules in the domain model are implemented as Prolog rules. The definition of two colleagues, X and Y, presupposes that they are both in the same department and that they are both at the same level of employment, that is either subordinates or managers of a kind. The latter condition means that the persons in question as nodes in the net has to be either sister nodes or subsumed by the same superconcept, the former condition is implemented using a shared variable, *AFD*, in the call of the knowledge base. A simplified version of the actual rule, then, is:

```
kollega(X,Y):-
        get_frame(STX,ako-Ako),
        get_frame(STY,ako-Ako),
        table(X,STX,AFD),
        table(Y,STY,AFD),
        X \== Y.
```

Using rules like this one is one way of incorporating domain-independent knowledge in the domain model. The user of the system is not supposed to have any knowledge about the data structures in the knowledge base. If you don't want to tune the knowledge base to some specific application or to be usable for only a limited amount of users you will have to supply the domain model with several rules like the one just presented, changing general knowledge about the domain into domain-specific knowledge and making implicit knowledge explicit.

5 Summary

In this paper we have presented some principles and methods used to map domain-specific and world knowledge into a domain model called FRAME_WORLD. We also showed that, having access to knowledge about conceptual entities and relationships in the domain in question, this model may be used as part of a reasoning mechanism to both check and generate types and values as valid arguments of semantic predicates . The aim of using such a domain model is to facilitate the dialogue between the end-user and a knowledge database. FRAME_WORLD is still just a toy model, but yet a useful tool to investigate and test principles and methods underlying the construction and use of a domain model.

References

Henriksen, Lina and Lene Haagensen. 1991. *Speciale om domænemodellering*. Institut for Datalingvistik, HHK.

Thomason, Richmond and David S. Touretzky. 1991. *Inheritance Theory and Networks with Roles* In: Sowa (ed.): *Principles of Semantic Networks*, Kaufmann, pp. 231–267.

Østerby, Tom. 1992. *Kunstig intelligens, metoder og systemer*. Polyteknisk Forlag.

Winston, Patrick H. 1974. *Artificial Intelligence*, Addison-Wesley.

Woods, W.A. 1987. *What's in a link? Foundations for Semantic Networks*, in: Brachmann/Levesque (eds): *Readings in Knowledge Representation*, Kaufmann, pp. 217–243.

Woods, W.A. 1991. *Understanding Subsumption and Taxonomy: A Framework for Progress*. In: Sowa (ed.): *Principles of Semantic Networks*, Kaufmann, pp. 45–95.

Robust Parsing with Charts and Relaxation

Peter Ingels
Linköping

Abstract

This paper is a summary of my master's thesis[1] "Error Detection and Error Correction with Chart Parsing and Relaxation in Natural Language Processing" (Ingels 1992). Two methods are presented: The first is a chart-based parsing algorithm inspired by C. Mellish that generates error classifications and, when possible, error corrections to ill-formed input. The algorithm classifies missing words, spurious words, misspellings and substituted words. The second approach presupposes a unification-based grammar formalism. The idea is to extend the PATR formalism so that it can represent alternatives to feature-values. The alternatives can then be used to "carefully" relax constraints imposed by the grammar. Thus the alternatives can be used to abduce corrections in the face of unification failures. The paper also contains a discussion of a proposed project on robustness.

Introduction

NLU-systems that are to be employed in real-world applications need to be able to handle input that violates the expectations of the grammar encoded in them. The occurrences of ungrammatical, or ill-formed, input in such systems is so frequent that it can not be ignored or treated simplistically (e.g. *Sorry, couldn't parse that*).

An informal study of 20 dialogues taken from our own corpus of NLI-dialogues collected with wizard of Oz techniques showed that some 18% of the user utterances contained at least one error. The errors were classified as misspellings, segmentation errors and syntactic errors. It should be noted that the results of the investigation depicted below was collected by a cooperative human, and it is more than reasonable to assume that the number of errors would be higher if an actual system would be used.

66 of the 369 utterances were erroneous (18%)

misspelling	segm. error	synt. error	>1 error/utt.
25	16	21	4

[1]The thesis work was carried out at IRST (Instituto per la Ricerca Scientifica e Tecnologica), Trento, Italy. Oliviero Stock acted as my supervisor. Fabio Pianesi contributed significantly concerning relaxation (see below). I wish to thank them both.

The fourth column shows the number of user utterances that contained more than one error.

A system that can handle these and other types of errors is called robust. Robustness can be achieved in different ways, but it requires minimally that the system is able to localize and classify the deviance. There are many different plausible error typologies, the one below is influenced by (Veronis 1991). See also (Stede 1992).

	Performance	Competence
Lexical level	letter substitution letter insertion letter deletion letter transposition	wrong inflection segmentation error grapheme substitution
Syntactic level	word substitution word insertion word deletion word transposition	wrong agreement homophone punctuation error rule violation
Semantic level		presupposition violation reasoning error dialogue law violation conceptual error

Competence errors result from the failure to abide by, or lack of knowledge of, linguistic rules. Performance errors are technical errors made despite knowledge of the rules. The concepts of competence errors and performance errors can of course also be enlarged to encompass errors related to domain knowledge as well as linguistic knowledge.

The appropriate action taken by the robust system in face of ill-formed input is not solely dependent on the error classification. The application in which the system is used is also relevant. Applications range from language tutoring systems over grammar and style-checkers to machine translation and dialogue systems. Spoken communication is also a highly relevant area. So what is to be judged as an appropriate action in an error situation varies.

- The system can enter into a clarification dialogue with the user
- The system can present the user with an error diagnosis
- The system can present the user with a correction hypothesis
- The system can use the best correction hypothesis without bothering the user
- The system can save a partial interpretation of the user utterance
- There might not have been an error (bad coverage)

Two approaches will be presented in the following two sections. First a chart-based technique that can detect constituent errors such as misspelled words (segmentation errors excluded), missing constituents, spurious constituents and substituted words, then a relaxation scheme for detection of constraint violation errors is presented. The relaxation technique has only been partially implemented. The last section is devoted to a discussion on extensions and further research.

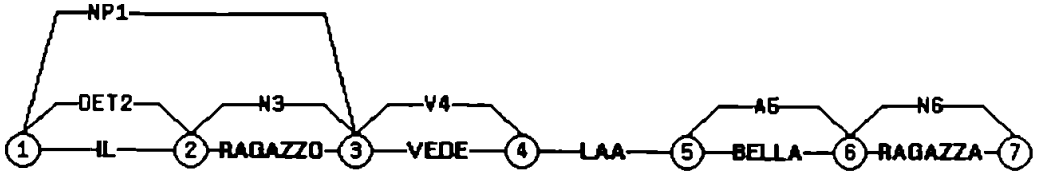
Constituent Errors

The techniques presented in this section rest on Mellish's paper "Some Chart-Based Techniques for Parsing Ill-Formed Input" (Mellish 1989). In his paper Mellish describes a variant of the chart parsing algorithm. His goal is to explore how far detection and classification of errors based purely on syntactic knowledge can lead. Thus he employs a CF-PSG (context-free phrase structure grammar) and the set of standard rules of chart parsing (combination and prediction of edges) is supplemented with a set of error hypothesis rules. These rules can detect and classify missing constituents, spurious constituents and substituted words. Actually he makes misspelling a special case of substituted word!

Mellish's algorithm invites to extensions and alterations and some improvements have also been made to the original algorithm. The improvements basically concerns the error hypothesis rules and some motivations will be accounted for in connection with the introduction of these rules. (There is no room here to present both versions and all considerations taken.)

The generalised chart parsing algorithm basically consists of two phases. First a standard bottom-up parser is supplied with the input. If the bottom-up parser fails the input is in some way ill-formed and recovery is attempted. Then a modified top-down parser is run on the input and the inactive edges left from the bottom-up phase. These inactive edges correspond to the complete constituents found in the bottom-up phase. One of the major differences between the modified top-down parser and the standard top-down parser is that the fundamental rule in the modified parser can incorporate constituents from either direction. In this way the fundamental rule can "narrow down" on an error-point. This scheme calls for a different way to represent an edge's needs and it also affects the top-down rule.

A schematic overview of the basic scheme is given below. The erroneous input in this example is 'Il ragazzo vede laa bella ragazza' ('The boy watches thee pretty girl').



Fgure 1: The chart after the bottom-up phase.

In figure 1, the chart is depicted as it looks when the second phase is ready to start. The superfluous active edges have been "cleaned away" and only the inactive edges that resulted from the bottom-up phase remain. The modified top-down parser would now behave something like:

Hypothesis:	need [S] 1->7
By top-down rule:	need [NP VP] 1->7
By fundamental rule with NP found bottom-up:	need [VP] 3->7
By top-down rule:	need [V NP] 3->7
By fundamental rule with V found bottom-up:	need [NP] 4->7
By top-down rule:	need [DET A N] 4->7
By fundamental rule with A and N found bottom-up:	need [DET] 4->5

This example gives a hint as to what the algorithm does. However, there are further complications. For example, there might be several errors in an input string and hence there must be a way to express multiple needs. If the input string in the example above instead was, 'Il ragazzo vede laa bella ragazza' ('The boy watches thee pretty giirl'), a need like the one below would be useful.

need [DET] 4->5 and [N] 6->7

Furthermore, there are "anchored" and "unanchored" needs. If a couple of consecutive constituents were sought for , say [NP VP] 1->7, and there is neither a complete NP nor a complete VP, this means that there is no way to tell where the two constituents meet. This is expressed with unanchored needs: need [NP] 1->*.

The "*" indicates a vertex in the chart that is not yet determined. Considering all this the general form for an edge will be as follows:

<C S->E needs cl_1 s_1->e_1, cl_2 s_2->e_2, ..., cl_n s_n->e_n>

Where C is a category, the cl_i are lists of categories (which will be shown inside square brackets), S and the s_i are positions in the chart and E and the e_i are positions in the chart or the special symbol "*". An edge of this type in the chart means that the parser is trying to find a constituent of category C, spanning from S to E. In order to do so it must then satisfy all the needs listed (cl_i s_i->e_i).

With this notation the two basic rules, the fundamental rule and the top-down rule, will have the following characteristics:

Top-down rule:
<C S->E needs [c_1,c_2,...,c_n] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>
c_1->RHS (in the grammar)

<c_1 s_1->e needs RHS s_1->e>
Where, if c_2,...,c_n is non-empty or e_1 = * then e = * else e = e_1
Precondition: e_1 = * or c_2,...,c_n is non-empty or there is no edge of category c_1 from s_1 to e_1

Fundamental rule:
<C S->E needs [c_1,...,c_{i-1},c_i,c_{i+1},...,c_n] s_1->e_1, cl_2 s_2->e_2, ...>
<c_i S_1->E_1 needs []>

<C S->E needs [c_1,...,ci_{-1}] s_1->S_1, [c_{i+1},...,c_n] E_1->e_1, cl_2 s_2->e_2, ...>
Precondition: $s_1 \leq S_1$ and (e_1 = * or $E_1 \leq e_1$)

These rules are sufficient to "narrow down" one error like in the example with 'Il ragazzo vede laa bella ragazza'. But since the interest is in the general case, where there can be an arbitrary number of errors in an input string, the parser is expected to by-pass the error-point in some way and to continue to search for possible additional errors. In this way all of an edge's needs will eventually get resolved. This is accomplished by the error hypothesis rules.

Garbage rule:
<C S->E needs [] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>

<C S->E needs cl_2 s_2->e_2, ..., cl_m s_m->e_m>
Precondition: $s_1 \neq e_1$

The garbage rule says that if all constituents of a particular need have been found, and a portion of that need's span is still not covered, this means that this uncovered portion of the chart contains words (or non-words) that should not be included in the parse. The C-constituent spans spurious words/non-words of the input string. That portion of the chart is

consequently disregarded and instead attention is focused on the next need. The garbage rule has not been altered from Mellish's version.

Missing word rule:
<C S->E needs [c_1,c_2,...,c_n] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>

<C S->E needs [c_2,...,c_n] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>
Precondition: c_1 is of lexical category and (s_1 = e_1 or (e_1 = * and (the word at s_1 is not of category c_1 or s_1 = the end of the chart)))

This rule hypothesizes missing word-errors. The rule differs from the corresponding rule in Mellish's algorithm in several respects. He allows for the c_1,c_2,...,c_n to be non-terminals and if s_1 = e_1 he can hypothesize the whole chunk c_1,c_2,...,c_n to be missing. This means that very blunt error classifications are produced, such as e.g. "missing [NP PP]". Furthermore the last clause of the precondition (e_1 = * and (the word at s_1 is not of category c_1 or s_1 = the end of the chart))) is not present in his version. This means that unanchored needs can not have missing constituents, which is an obvious weakness.

Unknown string rule:
<C S->E needs [c_1,c_2,...,c_n] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>

<C S->E needs [c_2,...,c_n] s_1+1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>
Precondition: c_1 is of lexical category and ($s_1 \neq e_1$ or e_1 = *) and s_1 < the end of the chart and the string at s_1 is unknown

Substituted word rule:
<C S->E needs [c_1,c_2,...,c_n] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>

<C S->E needs [c_2,...,c_n] s_1+1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>
Precondition: c_1 is of lexical category and ($s_1 \neq e_1$ or e_1 = *) and s_1 < the end of the chart and the word at s_1 is not of category c_1

The two last error hypothesis rules have only one counterpart in Mellish's version, namely the unknown word rule. With "unknown" words Mellish means both actual words that do not meet the present expectations and non-words (which obviously do not meet any expectations). With the present rules this distinction is respected. Thus the unknown string rule hypothesizes misspellings and the substituted word rule apply when the input contain a legitimate but misplaced word. However, note that transpositions require that the substituted word rule be applied twice, and so the relationship between the two transposed words is lost.

These extra rules will dramatically increase the parsing search space. In fact the search is exhaustive and obviously the hypothesizing of errors must be controlled in some way. This is done by means of heuristics. For each newly created edge a number of heuristics parameters will be calculated. These scores or penalties will determine an edge's priority compared to other newly created edges. The natural way to realise this procedure is to use the agenda. The agenda will thus be sorted according to the heuristics penalties with the most promising edge in the top position of the agenda. Functions described by Mellish include penalty so far (PSF, edges produced by the error hypothesis rules are penalized), mode of formation (MDE, the formation of unanchored edges are penalized) and several others. See Mellish (1989).

Constraint Violation Errors

This approach relies on the adoption of a feature-based - or unification-based grammar (UBG). The system, that has partially been implemented, makes use of a simple grammar encoded in the PATR-II formalism (Schieber 1986). In this paper the approach is merely sketched. For a full account see (Ingels 1992).

A technique for dealing with constraint violation errors is that of relaxation. This method is addressed in (Douglas & Dale 1992). In the paper D&D approach the problem by stating that some constraints are necessary and others are relaxable. If a unification fails some of the relaxable constraints can be relaxed. If the unification now succeeds a diagnosis of what was wrong with the input can be made. What is meant by relaxing a relaxable constraint in D&D's approach is simply not to incorporate any instantiation of the failed constraint in the resulting FS. In other words, dispose of the failed constraint altogether.

So with a sentence like *Do this cars have a good safety rating?* the resulting feature structure would not have a number feature with D&D&'s approach. A different approach would be to rely on the notion of the conflicting feature values as alternatives, or candidate values. In the example above, parsing *this cars*, the set of candidate values to the unification failure would be singular and plural. In this case other parts of the sentence can provide evidence for a plausible solution to the conflict. The idea is thus to capture the information implied by the unification failure.

The lexicon can also be used to record alternatives. E.g. the ill-formed Italian noun-phrase *la ragazzo* (the boy/girl) can be corrected as il ragazzo (the boy) or as la ragazza (the girl), while *la libro* (the book) only has one plausible correction since there exists no feminine

counterpart to the noun libro. (It should be noted here that alternatives are restricted to atomic values for reasons of complexity.)

The way to implement this scheme would be to explicitly represent the alternatives within the feature structure. So e.g. the Italian definite article *la* would have as value for gender ({f},{f m}), saying that the actual value for the feature gender is feminine although relaxable. The relaxability property is conveyed by the non-empty second component which also explicitly enumerates the possible alternatives to be used in case of unification failure. Non-relaxability is indicated by having the empty set (Ø) as the second component (no alternatives). The Italian noun *libro* could be relaxable having ({m},{m}) as value for gender. The unification (set intersection by pairs) of *la* and *libro* would then produce as value for gender (Ø,{m}), indicating a unification failure (Ø) and the singleton alternative masculine {m}, here functioning as a correction hypothesis.

The natural way to incorporate this scheme with Mellish's algorithm would be to consider only unification proper in the first phase. I.e. do not consider alternatives, look only for well-formed sentences, in the bottom-up phase. Then allow for relaxation in the second, error hypothesizing phase.

A Project on Robustness

A central aspect of the thesis work, presented briefly in the two preceding sections, is that assumptions of error occurrences are made explicitly. In the case of Mellish's algorithm errors are recorded in the chart edges since the error diagnosis is due to the expectations of a particular edge. Also assumptions regarding alternative interpretations of feature structures are explicitly represented. We believe this to be a practicable path to follow in the project too.

To keep track of alternative assumptions/interpretations a reasoned chart parser will be used. For a good survey of reasoned chart parsers see (Wirén 1992). A reasoned chart parser is a chart parser where dependencies between edges are explicitly recorded. With this framework the likelihood of alternative interpretations can be judged with reference to the assumptions on which they rest. In his dissertation Wirén suggests the reasoned chart parser to be integrated with an ATMS-based problem solver to support also such assumptions that can not be represented as chart edges. This setting will be used in the project as a general formal framework for studying diagnosis and interpretation of ill-formed input. Alongside with this we will gain knowledge of the error types occurring and their relative frequency. Another thing that should be empirically

investigated is the question regarding what action is appropriate in different error situations. This includes preventive actions.

The empirically collected information will then, together with the formal framework, serve as a basis for an implementation of a robust and reasonably fast interpreter, eventually to be integrated in the BILDATA-system. The BILDATA-system is the (written) dialogue system in our current project 'Dynamic Natural Language Understanding' (Jönsson 1993) .

Some of the questions relating to the implementation resulted from the thesis work.

Although my version of Mellish's algorithm makes the error classification more fine-grained than Mellish himself does, the error classification is inadequate. Transpositions and segmentation errors e.g. can not be dealt with in a straight forward manner. The reason being that an error hypothesis is kept local in an edge. That is not a problem as long as errors are discovered incrementally, one at a time, but when several constituents or input fragments are affected by a single error, there is a problem. This also raises the question whether there are any profitable alternatives to the two stage process suggested by Mellish. Maybe one should look out for 'lower level errors' (segmentation errors, misspellings,...) already in the first phase or in a third intermediate phase?

When should the system give up trying to parse the ill-formed input? Presently the system can parse everything (, you can put your elbow on the keyboard and the system will eventually come up with a diagnosis of what went wrong). The subtle question reads: how distorted can an utterance be and yet be understandable? What are the criteria for stating that the input is simply rubbish?

Another problem is the systems inability to discriminate between competing correction hypotheses. One reason for this is obviously that the system uses only syntactic information in the diagnosis process. Should semantic constraints be an integral part of the grammar and used as a filter in the parsing process?

References

Ingels, Peter. 1992. *Error detection and error correction with chart parsing and relaxation in natural language processing*. Master's Thesis, Department of Computer and Information Science, Linköping University, LiTH-IDA-Ex-9269.

Jönsson, Arne. 1993. *Dialogue Management for Natural Language Interfaces — An Empirical Approach.* Linköping Studies in Science and Technology, Dissertation No. 312.

Douglas, Shona and Robert Dale. 1992. *Towards Robust PATR.* In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-92)*, Vol. II: 468–474.

Mellish, C. S. 1989. *Some Chart-Based Techniques for Parsing Ill-Formed Input.* In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pp. 102–109.

Schieber, Stuart M. 1986. *An Introduction to Unification-based Approaches to Grammar.* P. University of Chicago Press, Chicago, Illinois, USA.

Stede, Manfred. 1992. *The Search for Robustness in Natural Language Understanding.* ARTIFICIAL INTELLIGENCE REVIEW, Vol. 6 No. 4: 383–414.

Véronis, Jean. 1991. *Error in Natural Language Dialogue Between Man and Machine.* INTERNATIONAL JOURNAL OF MAN-MACHINE STUDIES, Vol. 35, No. 2.

Wirén, Mats. 1992. *Studies in Incremental Natural-Language Analysis.* Department of Computer Science, Linköping University. Linköping Studies in Science and Technology, Dissertation No. 292.

1From Semantic Representations to SQL Queries

Per Anker Jensen, Bodil Nistrup Madsen
Annie Stahél, Carl Vikner
København

Abstract

Our paper discusses problems which arise when trying to translate a semantic representation into a SQL database query, and more particularly the encoding of yes/no-questions, and the evaluation of the existential and the universal quantifier.

1 The project

Our project investigates the problems which arise in creating a natural language interface to the database query language SQL. The basic layout of our system is a stepwise progression from a natural language expression via its semantic representation to a SQL query. The reason for choosing SQL as the database query language is that it is widely used in conventional database systems, like ORACLE for instance.

In our talk, we will be concerned specifically with the problems which arise when trying to translate a semantic representation into a SQL query, and more particularly with three problems which are of special importance for a natural language interface, namely the encoding of yes/no-questions, and the evaluation of the existential and the universal quantifier.

2 Semantic representations and restricted quantification

The semantic representation we employ is in the form of restricted quantification (cf. Jensen & Vikner (1992, I: 137–48)). An example with one quantifier is shown in (1):

(1) exists(x,customer1a(x),complain1a(x))
'En kunde klager'
'A customer complains'

Formulas like this one consist of four components as indicated in (2):

(2)

exists	(x,	customer1a(x),	complain1a(x))
↑	↑	↑	↑
QUANT	VAR	RESTRICTION	ASSERTION

A formula with the format shown in (2) we call a 'quantifier structure'. By 'predicate structure' we refer to expressions such as those which make up the restriction and the assertion in (2). These consist simply of a predicate followed by a number of arguments depending on the arity of the predicate. In this example, the meaning of the head noun of the subject, i.e. *kunde* ('customer'), is represented as the predicate structure in the restriction slot of the formula, and the intransitive VP *klager* ('complains') is represented as the predicate structure in the assertion slot.

It should be mentioned that neither the restriction nor the assertion is necessarily a predicate structure. Rather, they may contain any well-formed formula, hence also quantifier structures. An example of this is shown in (3), which contains a quantifier structure in the assertion slot.

(3) exists(y,product1a(y),
all(x,customer1a(x),complain2a(x,y))
'Alle kunder klager over en vare'
'All customers complain of a product'

The point of using restricted rather than unrestricted quantification, as is customary in classical predicate logic, is that when using restricted quantification, only a subset of the individuals in the domain is quantified over, namely those individuals which satisfy the restriction. This comes out clearly in the evaluation algorithms we propose for the quantifiers.

We want the evaluation of the quantifiers *exists* and *all* to be taken care of by the rough algorithms in (4) and (5), respectively, which ensure that only the individuals satisfying the restriction are considered when evaluating the assertion.

(4) **Evaluation of formulas of the form: exists(x,restriction(x),assertion(x))**

1. Find all the individuals that satisfy the restriction
2. IF at least one of the individuals found in step 1 satisfies the assertion
   THEN the formula evaluates to true
   ELSE the formula evaluates to false.

(5) **Evaluation of formulas of the form: all(x,restriction(x),assertion(x))**

1. Find all the individuals that satisfy the restriction
2. IF all the individuals found in step 1 satisfy the assertion
   THEN the formula evaluates to true
   ELSE the formula evaluates to false.

3 Database tables and semantic predicates

For the purpose of this paper we have designed a small toy database, whose tables are shown in (6). The database contains the names of four customers and three types of products and, in the table *cp*, are registered the complaints made by some of the customers.

(6) Tables in the database:

c (customers)

NO	NME
1	Hansen
2	Jensen
3	Madsen
4	Sørensen

p (products)

NO	TYP
1	television set
2	video recorder
3	video camera

cp (complaints)

NO	CNO	PNO
1	3	1
2	2	1
3	3	2
4	1	1

When we want to evaluate a semantic representation with respect to this database, we have to relate the predicates of the semantic representation to the tables in the database. Following Grosz et al. (1987:222), this is done by means of a set of definitions of the predicates in terms of database tables as shown in (7).

(7) Definitions of semantic predicates in terms of database tables:

```
complain1a(Nme) ←
        c(Cno,Nme),
        cp(_,Cno,_)

complain2a(Nme,Pno) ←
        c(Cno,Nme),
        cp(_,Cno,Pno)
```

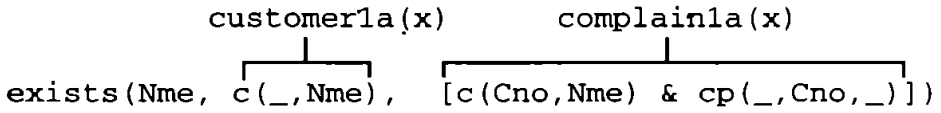
customer1a(Nme) ←
    c(_,Nme)

product1a(Pno) ←
    p(Pno)

television_set1a(Pno) ←
    p(Pno,'television set')

The predicate structures *customer1a(x)* and *complain1a(x)* in the semantic representation in (2) can now be replaced by their respective definitions. By doing so we obtain the expression shown in (8), which we call the tabular representation:

(8)



4 SQL and yes/no-questions

The SQL language[1] offers a facility for retrieving information from a database, namely the so-called SELECT queries. SELECT queries come in two types. A set-valued type and a number-valued type.

(9)

```
SELECT * FROM...          ┐
                          ├ set-valued type
SELECT c.NME FROM...      ┘

SELECT COUNT(*) FROM...      number-valued type
```

In the set-valued type, the SELECT list, i.e. the expression between the token *SELECT* and the token *FROM*, is a sequence of column specifications. The value of this type of queries is the set of tuples of values in the indicated columns of the rows which satisfy the condition of the query. To represent the value of such a query one can use the set notation proposed by Pirotte (1978:414), as shown in (10):

(10) { (x,y) | t(x,y) }

[1]For details of the SQL language, see for instance Date (1990), Ørum (1990) or *SQL Language Reference Manual. ORACLE* (1990).

In this notation the SELECT list *(x,y)* is written in front of a tabular formula containing the corresponding free variables. If we have a query of the form (11):

(11) SELECT c.NO, c.NME FROM c;

we can represent its value by means of the set expression in (12):

(12) { (NO,NME) | c(NO,NME) }

In the kind of number-valued type which is relevant to the subject of this paper, the SELECT list consists of the expression *COUNT(*)*. The value of a query of this form is the number of rows in the table which satisfy the condition of the query.

Numbers and sets (of tuples) are the only two kinds of possible answers to SQL queries. That is, unlike for instance Prolog, SQL does not support yes/no-questions directly. Therefore, we have to somehow trick it into doing so. Our stratagem consists in making use of the built-in SQL table DUAL. DUAL is a table with one column and one row with the value *X*. We begin all queries which encode a yes/no-question by the expression *SELECT COUNT(*) FROM DUAL*. Such a query yields the answer *1* if the condition is satisfied, and *0* otherwise. So, this gives us the equivalent of a yes/no-question facility.

The content proper of the yes/no-question is encoded by means of a SELECT subquery. This is shown in example (13), where the content 'Hansen complains' is encoded in the condition in the innermost WHERE clause, which checks the occurrence of a customer name *Hansen* whose customer number appears in a row in the complaints table. This SELECT subquery is made part of the WHERE clause of the outermost SELECT statement by means of the operator EXISTS. In this way we get a condition which comes out true – and thus triggers a *1* as the final answer – only in the case where the value of the SELECT subquery is nonempty.

(13)

a. *Hansen klager*
'Hansen complains'

b. Semantic representation:
complain1a(Hansen')

c. Tabular representation:
c(Cno,'Hansen') & cp(_,Cno,_)

d. SQL query:

```
SELECT COUNT(*) FROM DUAL
WHERE EXISTS
  (SELECT * FROM c, cp
  WHERE c.NME = 'Hansen'
  AND c.NO = cp.CNO);
```

5 Existential quantification

Turning next to existential quantification, Pirotte (1978: 419) has it that one can transform a formula containing the existential quantifier into an equivalent set expression, and thus remove the existential quantifier.[1] For instance (14.a) can be transformed into (14.b):

(14) a. exists(x,p(x),q(x))
b. { x | p(x) & q(x) } ≠ ∅

We use a transformation like the one in (14) as the basis for translating existentially quantified formulas into SQL queries. Thus the existentially quantified semantic representation in example (15.b and c) is transformed into the SELECT subquery in example (15.d) which encodes a set expression corresponding to the lefthand side of (14.b).

(15)

a. *En kunde klager*
'A customer complains'

b. Semantic representation:

```
exists(x,customer1a(x),complain1a(x))
```

c. Tabular representation:

```
exists(Nme,c(_,Nme),
                [c(Cno,Nme) & cp(_,Cno,_)])
```

d. SQL query:

```
SELECT COUNT(*) FROM DUAL
WHERE EXISTS
      (SELECT c.NME FROM c, cp
      WHERE c.NME LIKE '%'
      AND c.NO = cp.CNO);
```

[1]For other discussions of the elimination of the existential quantifier in database queries, see e.g. Minker (1978: 110), Dilger & Zifonun (1978: 395-400), Pereira (1983: 21), Steiner (1988: 186-87).

The LIKE-condition of the inner WHERE clause is redundant and is deleted by optimization.

Note that the *EXISTS* of the outer WHERE clause corresponds, not to the existential quantifier, but to the symbols "$\neq \varnothing$" of (14.b). So, in our treatment the existential quantifier disappears altogether. However, it would be possible to encode the quantifier *exists* by the SQL-operator *EXISTS*. In example (15) this would give a SQL query identical to the one shown. But in more complicated cases, i.e. examples containing multiple occurrences of quantifiers, this would result in a considerable number of subqueries (cf. Madsen & Stahél (forthcoming)).

6 Universal quantification

For the encoding of universal quantification we use the SQL operator MINUS, as shown in example (16). MINUS takes as its arguments two SELECT queries of the set-valued type and maps them onto the set-theoretic difference between the value of the first and the value of the second. The idea is that if we want to find out if all customers complain of some product, as in example (16), we find the difference between the set of customers and the set of complainers. If the resulting set is empty, then all customers complain, and so the initial query should receive a positive answer. That is, in the case of universal quantification, the subquery is a MINUS construction, and the value of this subquery must be the empty set for the outermost SELECT query to yield the value *1*. That is why the condition of the outermost WHERE clause is constructed by means of the expression *NOT EXISTS*.

The MINUS operator must be given comparable sets as arguments. In our example (16) these are sets of customer names determined by the SELECT list *c.NME* figuring in both SELECT expressions. This column designation is the encoding of the variable bound by the universal quantifier in the semantic representation.

(16)

a. *Alle kunder klager over en vare*
'All customers complain of a product'
= 'Each customer complains of some product'

b. Semantic representation:
all(x,customer1a(x),
exists(y,product1a(y),complain2a(x,y)))

c. Tabular representation:

```
all(Nme,c(_,Nme),
        exists(Pno,p(Pno,_),
                [c(Cno,Nme) & cp(_,Cno,Pno)]))
```

d. SQL query:

```
SELECT COUNT(*) FROM DUAL
WHERE NOT EXISTS
        (SELECT c.NME FROM c
        MINUS
        SELECT c.NME FROM p, c, cp
        WHERE p.NO = cp.PNO
        AND c.NO = cp.CNO);
```

The MINUS solution to universal quantification is analoguous to the analysis advocated by the theory of generalized quantifiers, which states the truth conditions of an expression of the form in (17):

(17) all N VP

as shown in (18):

(18) $[\![\text{all N VP}]\!] \equiv [\![\text{N}]\!] \subseteq [\![\text{VP}]\!]$

(cf. Barwise & Cooper (1981: 169), Thomsen (forthcoming)). The subset statement in (18) is equivalent to a statement in terms of set-theoretic difference of the form given in (19):

(19) $[\![\text{N}]\!] - [\![\text{VP}]\!] = \varnothing$

And this again is exactly what we have encoded by means of the MINUS construction.

Until this point the encoding of existential quantification has been relatively easy, and we have avoided the burden of keeping track of variables bound by the existential quantifier. However, if we have a semantic representation with a universal quantifier in the scope of an existential quantifier, as in example (20), such recklessness is no longer admissible. In example (20), the existential quantifier binds the variable designating the product, i.e. *y* or *Pno*. This variable appears again inside the scope of the universal quantifier. The point is that, for the formula to be true, it must be possible to find one particular value for this variable such that all customers complain of the product which has this number. Therefore we have to fix values for the variable outside the scope of the universal quantifier. This is done by giving the existential SELECT subquery in example (20) the SELECT list *p.NO* and repeating this

column specification in the universal SELECT subquery in the last AND clause, where it is required to be identical to *cp.PNO*.

(20)

a. *Alle kunder klager over en vare*
'All customers complain of a product'
= 'There is a product which all customers complain of'

b. Semantic representation:
exists(**y**,product1a(y),
all(x,customer1a(x),complain2a(x,**y**)))

c. Tabular representation:
exists(**Pno**,p(Pno,_),
all(Nme,c(_,Nme),
[c(Cno,Nme) & cp(_,Cno,**Pno**)]))

d. SQL query:

```
SELECT COUNT(*) FROM DUAL
WHERE EXISTS
  (SELECTS p.NO FROM p
   WHERE NOT EXISTS
     (SELECTS c.NME FROM c,
   MINUS
     SELECT c.NME FROM c,
   WHERE c.NO = cp.CNO
   AND p.NO = cp.PNO));
```

universal subquery

existential subquery

Thus, only in cases like this one, where the existential quantifier has a universal quantifier in its scope, do we have to keep track of an existentially bound variable.

7 Conclusion

To summarize, the table DUAL is used to encode yes/no-questions, the existential quantifier may be eliminated and the universal quantifier is encoded by means of the MINUS operator.

References

Barwise, Jon and Robin Cooper. 1981. *Generalized Quantifiers and Natural Language.* LINGUISTICS AND PHILOSOPHY, 4, 159–219.

Date, C.J. 1990. *An Introduction to Database Systems.* Volume I, Fifth Edition, Addison–Wesley, Reading, Massachusetts.

Dilger, Werner and Gisela Zifonun.1978. *The Predicate Calculus-Language KS as a Query Language.* In Gallaire and Minker, pp. 377–408.

Gallaire, Hervé and Jack Minker (eds). 1978. *Logic and Databases*, Plenum Press, New York.

Grosz, Barbara J., Douglas E. Appelt, Paul A. Martin and Fernando C.N. Pereira. 1987. *TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces.* ARTIFICIAL INTELLIGENCE 32, 173–243.

Jensen, Per Anker and Carl Vikner. 1992. *Natursprogsbehandling og unifikations-grammatik*, I–II. Department of Computational Linguistics, Copenhagen Business School.

Madsen, Bodil Nistrup and Annie Stahél (forthcoming). *Database structure and SQL-queries for a natural language interface project.* LAMBDA 19, 1993.

Minker, Jack. 1978. *An Experimental Relational Data Base System Based on Logic.* In Gallaire and Minker, pp. 107–21.

Pereira, Fernando. 1983. *Logic for Natural Language Analysis.* Technical Note 275, SRI International, Menlo Park, California.

Pirotte, Alain. 1978. *High Level Data Base Query Languages.* In Gallaire and Minker, pp. 409–36.

SQL Language Reference Manual. ORACLE. 1990. Oracle Corporation.

Steiner, Juraj. 1988. *Relational theory of queries.* DATA AND KNOWLEDGE ENGINEERING 3, pp. 181–96.

Thomsen, Hanne Erdman (forthcoming). *The Semantic Values of NPs.* LAMBDA 19.

Ørum, Henning. 1990. *Oracle-håndbogen.* Borgen, Copenhagen.

Clustering Sentences — Making Sense of Synonymous Sentences

Jussi Karlgren, Björn Gambäck and Christer Samuelsson
Stockholm

Abstract

The paper describes an experiment on a set of translated sentences obtained from a large group of informants. We discuss the question of transfer equivalence, noting that several target-language translations of a given source-language sentence will be more or less equivalent. Different equivalence classes should form clusters in the set of translated sentences. The main topic of the paper is to examine how these clusters can be found: we consider — and discard as inappropriate — several different methods of examining the sentence set, including traditional syntactic analysis, finding the most likely translation with statistical methods, and simple string distance measures.

1 Introduction

The idea that there is a one-to-one correspondence between sentences in one language and sentences in another is obviously ridiculous to anyone who has tried to translate between any pair of languages. When translating, the aim is not to find **the** correct translation but **a** correct one. For almost any sentence in a source language several sentences in the target language will do: there will not be one good sentence but a set of them, more or less synonymous or **homeosemous** (H. Karlgren, 1974). What a translator (or an information retrieval intermediary) tries to do is to produce a transfer equivalence, i.e., a sentence or a sequence of sentences with a similar or identical pragmatic effect.

This is a decision problem when translating, and an evaluation problem when done. As will be shown below, even for trivially simple source language sentences and utterances there will be a large number of corresponding target language sentences. It would be useful to find a simple method of ranking sentences in such a set to use when evaluating the translation produced by a machine translation (MT) system.

Historically there has been little emphasis on evaluation in the machine translation community, and although that is now starting to change, the methods proposed are often quite *ad hoc*. The strategy chosen for a particular evaluation of course depends on the reasons for the evaluation; or more specifically on who the evaluator is. Developers of MT-systems, end-users and prospective buyers will by necessity evaluate systems in

different ways. Following for example Way (1991) MT evaluation strategies are divided into three broad classes:

Typological Evaluation is a developer-oriented strategy aiming at specifying which particular linguistic constructions the system handles satisfactorily and which it does not.

Declarative Evaluation is the strategy commonly used when assessing human translators work; scoring the output with respect to various quality dimensions (such as accuracy, intelligibility and style).

Operational Evaluation is the way end-users and MT-system buyers normally evaluate the systems: measuring how cost- and time-effective a particular system is when used in a specific translation environment.

The principal tool for typological evaluation is a **test suite**, a set of sentences which individually represent specified constructions and hence constitute performance probes. Most work on MT-system evaluation has been concerned with how such a test-suite should be composed, e.g. (King & Falkedal, 1990, and Gambäck *et al*, 1991a, 1991b); however, the methods outlined in this paper follow the declarative evaluation track. Previous methods along this path have normally been "hand-crafted", or based on existing (labour-intensive) methods for the evaluation of human translators' work (Balkan, 1991). Both Thompson (1991) and Su *et al* (1992) have independently worked on automating the process. They present methods for evaluating translation quality based on statistical measurements of a candidate translation against a standard set using simple string-matching algorithms, i.e., ideas quite akin to the ones below.

The rest of the paper is outlined as follows: in the section following we describe an experiment with obtaining a set of translated sentences from a large group of informants. In section 3 we discuss what conclusions can be drawn from the experiment, the key questions being what the structure of the sentence set is and if the set contains clusters. The main topic of the section is how clusters can be found: we consider several different methods of examining the sentence set, including traditional syntactic analysis, finding the most likely translation with statistical methods, and simple string distance measures. Section 4, finally, sums up the previous discussion and points to other possible research directions.

2 Empirical Evidence

In order to find out the extent of divergence of translations, the sentence space, we distributed twelve randomly chosen sentences from a corpus of 4021 spoken English sentences to 1100 Swedish computer scientists. We

received 73 answers. The translations were inspected by a professional Swedish translator, and all but a few were considered quite acceptable in a situation corresponding to the one in which they were given. The sentences distributed are shown in table 1 below. They were all in the air traffic information domain, or ATIS, the corpus used by the US government to evaluate the performance of different spoken language understanding systems (Boisen & Bates, 1992).

TABLE 1: Sentences distributed

1	Atlanta to Oakland Thursday.
2	Give me flights from Denver to Baltimore.
3	Which companies fly between Boston and Oakland.
4	Show me all flights from Pittsburgh to Dallas.
5	Show me the names of airlines in Atlanta.
6	What's the cheapest flight from Atlanta to Baltimore.
7	I want to fly from Baltimore to Dallas round trip.
8	Show all flights and fares from Denver to San Francisco.
9	List round trip flights between Boston and Oakland using T W A.
10	What are the flights from Dallas to Boston for the next day.
11	And the ground what is the ground transportation available in the city of Philadelphia.
12	I need a flight leaving Pittsburgh next Monday arriving in Fort Worth before ten a m.

Even the simplest sentence in the test set proved surprisingly divergent: number 1 was translated to twelve different Swedish sentences. For number 12, and the longest sentence in the test set, we received 68 different translations, all of them judged as "good" by the professional translator. Table 2 sums up how the sentences as a whole were translated.

TABLE 2: Summary of responses

Sentence	translations	good	different	most common
1	73	72	12	27
2	74	72	61	4
3	68	66	19	39
4	69	67	36	7
5	73	68	43	9
6	70	68	37	7
7	72	65	27	25
8	70	65	50	10
9	71	71	12	27
10	70	70	62	3
11	68	55	66	2
12	68	68	68	1

A natural choice for a goal translation is to pick the most common translation. For the first sentence in the test set this would give us an appropriate result, the most common translation occurring 27 times; however, for more elaborate sentences this cannot always be done, as shown by sentence number 12. To pick the most typical one, we need to rank the translations. Tables 3 and 4 in the appendix show such frequency ranking for sentences 1 and 5, respectively.

3 What does the study mean?

So, for seventy informants, we received up to seventy non-pathological translations of non-pathological sentences. The question is what the structure of the sentence set is. Are all the sentences synonymous, or does the divergence reflect polysemy on the sentence level? If the sentence set is synonymous, are the sentences just variations over a homogenous space, or are the discernible strategies on some level that can be identified? In effect, what we are asking is if the sentence set contains clusters, or are equidistant, as in figure 1.

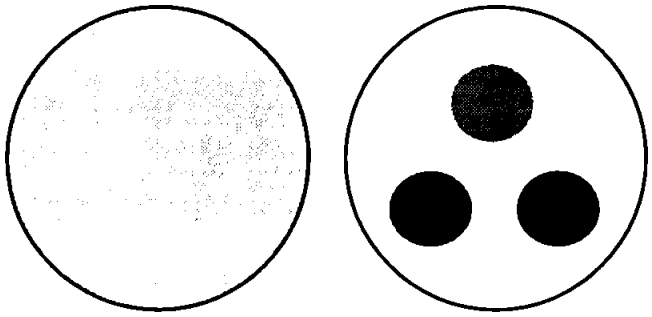


FIGURE 1: Two sentence sets, with equidistant sentences resp. clusters

We will in the following sections consider several different methods of examining the sentence set to find clusters or strategies. First we examine finding the most likely translation with statistical methods, then simple string distance measures, before moving on to traditional syntactic analysis. In passing, we first note that a methodological question that needs to be addressed in a study of this type is whether there is a correct answer to be found as regards the structure of sentence sets. One way of doing this is to ask test subjects to group sentences manually. We have not done this in this small study, but trusted our own judgment as to the likeness between sentences.

3.1 The most likely translation

One obvious way of picking the most typical candidate translation is to choose the most likely one. This is done by comparing the probabilities of the candidate strings. In order to do this, we need a probabilistic language model, i.e., a method of assigning a probability to each string. A simple, but very successful, probabilistic langugage model is the bigram model. In the general case, the probability of a word string $w_1,...,w_n$ is calculated recursively:

$$p(w_1,...,w_n) = p(w_n \mid w_1,...,w_{n-1}) \ldots p(w_1,...,w_{n-1}) =$$

$$= \prod_{k=2 \to n} p(w_k \mid w_1,\ldots,w_{k-1})$$

The bigram model approximates the factors $p(w_k \mid w_1,\ldots,w_{k-1})$ with the factors $p(w_k \mid w_{k-1})$ — only the word immediately preceeding the current word is taken into account, while the rest of the preceeding string is discarded. Thus, to calculate the string probability all that is used is the probability of each word given any predecessor (bigrams are treated in more detail by e.g. Jelinek, 1990) This gives us the bigram approximation of the string probability of the word string $w_1,\ldots,w_n$:

$$p(w_1,\ldots,w_n) \approx \prod_{k=2 \to n} p(w_k \mid w_{k-1})$$

The probabilites $p(w_k \mid w_{k-1})$ are calculated from the relative frequencies of word pairs in the set of candidate translations corresponding to a sentence in the source language:

$$p(w_k \mid w_{k-1}) \approx f(w_{k-1},w_k) \,/\, f(w_{k-1})$$

A different set of probabilities is derived for each source sentence using only the various candidate translations. After all, we are trying to find the most likely translation of this particular sentence. Instead of comparing the probabilities directly, we compare their logarithms, the logarithm function being monotonously increasing. Multiplying probabilities amounts to the same thing as adding their logarithms. Thus

$$\ln\{p(w_1,\ldots,w_n)\} \approx \sum_{k=2 \to n} \ln\{p(w_k \mid w_{k-1})\}$$

In order not to penalize longer word strings, the sum is normalized by the string length, giving us the following norm $|w_1,\ldots,w_n|$ of the string $w_1,\ldots,w_n$.

$$|w_1,\ldots,w_n| = -\,1 / n \cdot \sum_{k=2 \to n} \ln\{p(w_k \mid w_{k-1})\}$$

The minus sign is included to make the norm positive and give more likely sentences smaller norms. This means that $\exp\{-\,|w_1,\ldots,w_n|\}$ is the geometric mean of the probability of each word w_k in its context, or in other words, its likelihood of occurrence. The probability of a word string $w_1 \ldots w_n$:

$$p(w_1 \ldots w_n) = p(w_n \mid w_1 \ldots w_{n-1}) \cdot p(w_1 \ldots w_{n-1}) =$$
$$= \prod_{k=2 \to n} p(w_k \mid w_1 \ldots w_{k-1}) \approx \prod_{k=2 \to n} p(w_k \mid w_{k-1})$$

Noting that both w_1 and w_n are sentence delimiters (eos), the probability of the sentence *"Atlanta to Oakland Thursday"* is

$$p(\text{eos,Atlanta,to,Oakland,Thursday,eos}) \approx$$
$$\approx p(\text{Atlanta} \mid \text{eos}) \cdot p(\text{to} \mid \text{Atlanta}) \cdot p(\text{Oakland} \mid \text{to}) \cdot$$
$$\cdot\, p(\text{Thursday} \mid \text{Oakland}) \cdot p(\text{eos} \mid \text{Thursday})$$

For the simpler sentence, the bigram statistics produce a similar ranking as do the simple counts of occurrence — not very surprising. Table 3 of the appendix show the bigram rankings for source sentence 1 together with the likelihood (frequency) of the translated target sentences. Table 5

shows that the bigram rankings manage to separate the different translations of sentence 12, a sentence for which pure frequency measures gave no information at all.

3.2 String Distance Methods

Simple string distance measures are designed to match strings of characters rather than strings of words; however, they can be modified to fit these measures as well. Wagner & Fischer (1974) and Lowrance & Wagner (1975) define string distance measures based on primitive string correction operations: replace, delete, insert, and swap. If there is a sequence of edit operations to construct A from B, and N_R, N_D, N_I and N_S are the number of replacements, deletions, insertions and swaps needed in this sequence to convert A to B, and W_R, W_D, W_I, and W_S are costs associated with the operations respectively, the cost of constructing B from A will be the minimum of the following function:

$$D(A,B) = N_R \cdot W_R + N_D \cdot W_D + N_I \cdot W_I + N_S \cdot W_S$$

The distance from string A to string B is defined as the cost of the least cost edit sequence. The measurements were applied to the words as they appeared in the text giving edit distances both character by character and word by word.

After computing the distances between sentences, we need to examine which one of the strings is the most typical. There are standard methods for this type of analysis: we use agglomerative hierarchical clustering, i.e., we assume the sentences all are in separate clusters and repeatedly join the closest pair of clusters until we only have one cluster left. We calculated distance between clusters using two strategies: **complete linkage** and **single linkage** as illustrated in figures 2 and 3.

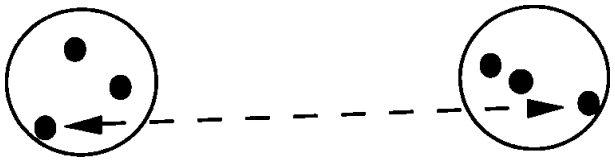


FIGURE 2: Distance between clusters using complete linkage measures

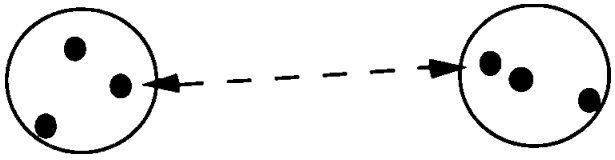


FIGURE 3: Distance between clusters using single linkage measures

In the first case the closest pair of clusters is defined as those where the distance between **furthest** neighbours is minimized, and in the second case as clusters where the distance between **closest** neighbours is minimized. We found that complete linkage gave us a faster clustering, using less steps, and that single linkage yielded a larger number of derivational steps. For most of the experiments, a large number of steps provided more information, so we used single linkage as the preferred strategy.

The results are displayed in the dendrograms in figure 4 in the appendix below (with translations numbered as in tables 3 and 4). In the single linkage based dendrogram for sentence 5 (at the right of figure 4), the two closely related sentences

"visa alla bolag representerade i atlanta"

"vilka flygbolag finns representerade i atlanta"

(translations 21 and 23) are shown to be in different clusters, which naturally is not the desired result.

3.3 Traditional Syntactic Analysis

Consider the following sentence and its translations:

Show me the names of airlines in atlanta.

Vilka flygbolag finns i Atlanta
Vilka flygbolag flyger på Atlanta
Vilka flygbolag trafikerar Atlanta

The three translations correspond to two different syntactic types, and two different propositional contents, whatever way their meaning is analyzed. However, the division by syntactic criteria is different from the division by semantic criteria. Syntax is not the right analysis level to examine complete sentences, since it is concerned with intra-clausal relations, which tend to lose their relevance when larger discourse segments are examined (J. Karlgren, 1993). The aim is to find a level of description with an adequate granularity.

4 Simple Methods: How and Why They Fail

Both statistical and word identity metrics only utilize local information on relatively scarce data. While these types of method are simple to implement, they give relatively little of use for the level of processing we

are interested in. Syntactic analysis does not help immediately, as shown by the examples in section 3.3 above. One way to alleviate the arbitrariness of the analysis would be to enlarge the classes of objects studied, by both lexically based methods that equate classes of words — synonym classes, or near synonym classes.

Another way to condense the data better would be to use "demi-structural methods", which add some structure to the text by constructing surface constituents of a relevant level, like complete NP:s and PP:s to perform the analysis. With the advent of reliable surface syntax analysis components (as the ones of, e.g., Voutilainen & Tapanainen, 1993), this could be done with relative little trouble. The idea of leaving certain troublesome grammatical properties to the top level, to be handled by rules of a different type rather than resolving all on the bottom seems to be fruitful.

References

Balkan, Lorna. 1991. *Quality Criteria for MT.* Technical Report. University of Essex, Colchester, England.

Boisen, Sean and Madeleine Bates. 1992. *A Practical Methodology for the Evaluation of Spoken Language Systems.* pp. 162–169, *Proceedings of the 3RD Conference on Applied Natural Language Processing.* Trento, Italy.

Gambäck, Björn, Hiyan Alshawi, Manny Rayner, and David Carter. 1991a. *Measuring Compositionality of Transfer . 1st Meeting of the International Working Group on Evaluation of Machine Translation Systems.* Les Rasses, Switzerland.

Gambäck, Björn, Hiyan Alshawi, Manny Rayner, and David Carter. 1991b. *Measuring Compositionality in Transfer-Based Machine Translation Systems . The ACL Workshop for Evaluation of Natural Language Processing Systems.* University of California, Berkeley, California.

Jelinek, Fred. 1990. *Self-Organizing Language Models for Speech Recognition.* pp. 450–506, READINGS IN SPEECH RECOGNITION. Morgan Kaufmann, San Mateo, California.

Myers, Eugene W. 1986. *An O(ND) Difference Algorithm and its Variations.* pp. 251–266, ALGORITHMICA, 1. Springer-Verlag, New York, New York.

Karlgren, Hans (ed.). 1974. *Homeosemi.* KVAL PUBLIKATION 1974:1. Skriptor, Stockholm, Sweden.

Karlgren, Jussi. 1993. *Syntax in Information Retrieval.* In *Proceedings from the 1st NorFa Doctoral Symposium on Computational Linguistics.* Department of General and Applied Linguistics, Copenhagen University, Copenhagen, Denmark.

King, Maggie and Kirsten Falkedal. 1990. *Using Test Suites in Evaluation of Machine Translation Systems.* pp. 211–216, In *Proceedings of the 13th International Conference on Computational Linguistics,* Volume 2. Helsinki, Finland.

Lowrance, Roy and Robert A. Wagner. 1975. *An Extension of the String-to-String Correction Problem.* pp. 177–183, JOURNAL OF THE ACM, Volume 22, Number 2.

Su, Keh-Yih, Ming-Wen Wu, and Jing-Shin Chang. 1992. *A New Quantitative Quality Measure for Machine Translation Systems.* pp. 433–439, In *Proceedings of the 14th International Conference on Computational Linguistics.* Nantes, France.

Thompson, Henry S. 1991. *Automatic Evaluation of Translation Quality Using Standard Sets. 1st Meeting of the International Working Group on Evaluation of Machine Translation Systems.* Les Rasses, Switzerland.

Voutilainen, Atro and Pasi Tapanainen. 1993. *Ambiguity Resolution in a Reductionistic Parser.* In *The 6th Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht, Holland.

Wagner, Robert A. and M. J. Fischer. 1974. *The String-to-String Correction Problem.* pp. 168–173, JOURNAL OF THE ACM, Volume 21, Number 1.

Way, Andrew. 1991. *Developer Oriented Evaluation of MT Systems.* TECHNICAL REPORT, University of Essex, Colchester, England.

Appendix

TABLE 3: Translation frequencies and bigram probabilities for 1: *"Atlanta to Oakland Thursday"*.

0	27	0,86	atlanta till oakland på torsdag
1	18	0,79	från atlanta till oakland på torsdag
2	12	0,66	atlanta till oakland torsdag
3	1	0,62	från atlanta till oakland torsdag
4	4	0,53	atlanta oakland på torsdag
5	1	0,45	från atlanta till oakland på torsdagen
6	2	0,45	från atlanta till oakland torsdagar
7	1	0,40	atlanta till oakland
8	2	0,34	atlanta oakland torsdag
9	1	0,25	atlanta oakland kommande torsdag
10	1	0,19	på torsdag från atlanta till oakland
11	1	0,10	torsdag atlanta till oakland

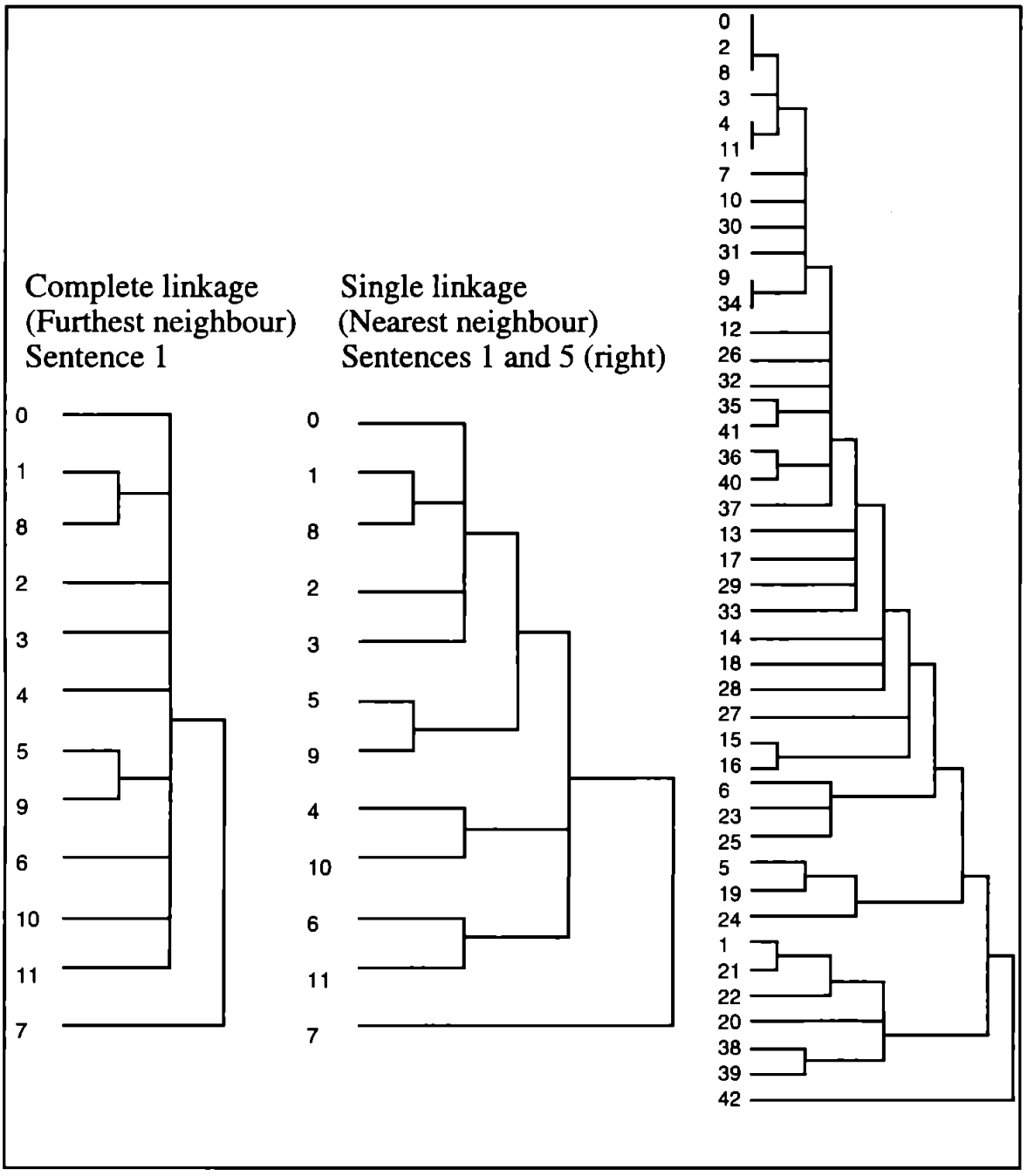
TABLE 4: Translation frequencies for 5: *"Show me the names of airlines in Atlanta"*.

0	10	visa mig namnen på flygbolagen i atlanta
1	6	vilka flygbolag finns i atlanta
2	6	visa mig namnen på flygbolag i atlanta
3	3	visa mig namnen på alla flygbolag i atlanta
4	3	visa namnen på flygbolagen i atlanta
5	2	vilka flygbolag flyger på atlanta
6	2	visa alla flygbolag i atlanta
7	2	visa mig flygbolagen i atlanta
8	2	visa mig namn på flygbolag i atlanta
9	2	visa mig namnen på flyglinjer i atlanta
10	2	visa namnen på alla flygbolag i atlanta
11	2	visa namnen på flygbolag i atlanta
12	1	ge mig namnen på flygbolag representerade i atlanta
13	1	ge mig namnen på flyglinjerna i atlanta
14	1	jag vill veta namnen på flygbolag i atlanta
15	1	kan jag få namnen på flygbolag i atlanta
16	1	kan jag få se namn på flygbolag i atlanta
17	1	vad är namnen på flygbolagen i atlanta
18	1	var god visa namnen på flyglinjerna i atlanta
19	1	vilka bolag flyger på atlanta
20	1	vilka bolag har kontor i atlanta
21	1	vilka flygbolag finns representerade i atlanta
22	1	vilka flygbolag trafikerar atlanta
23	1	visa alla bolag representerade i atlanta
24	1	visa alla flygbolag som flyger på atlanta
25	1	visa mig all flygrutter i atlanta
26	1	visa mig alla namn av flygbolag i atlanta
27	1	visa mig bolagen som flyger på atlanta
28	1	visa mig name på alla flyglinjer i atlanta
29	1	visa mig namnen av luftlinjer i atlanta
30	1	visa mig namnen för flygbolag i atlanta
31	1	visa mig namnen på bolagen i atlanta
32	1	visa mig namnen på de flygbolag som finns i atlanta
33	1	visa mig namnen på flyglinjer till atlanta
34	1	visa mig namnen på flyglinjerna i atlanta
35	1	visa mig namnen på flygrutterna i atlanta
36	1	visa mig namnen på linjer i atlanta
37	1	visa mig namnet på alla flygbolag i atlanta
38	1	visa mig vilka bolag som finns i atlanta
39	1	visa mig vilka flygbolag som finns i atlanta
40	1	visa namn på linjer i atlanta
41	1	visa namnen på flygrutter i atlanta
42	1	visa upp flyglinjer som avgår från atlanta

TABLE 5: Bigram probabilities for sentence 12

0,60	jag behöver en flight från Pittsburgh nästa måndag som är framme i Fort Worth före klockan tio
0,60	jag behöver en flight från Pittsburgh nästa måndag som anländer i Fort Worth före tio på förmiddagen
0,58	jag vill ha en flight från Pittsburgh nästa måndag som anländer i Fort Worth före klockan tio
0,58	jag behöver flyga från Pittsburgh nästa måndag och komma fram till Fort Worth före tio på morgonen
0,56	jag vill ha ett flyg från Pittsburgh nästa måndag som är framme i Fort Worth före klockan tio på morgonen
	...
0,27	jag behöver en biljett från Pittsburgh framme i Fort Worth innan tio nästa måndag
0,26	jag söker en flight till Pittsburgh nästkommande måndag som beräknas vara framme före klockan tio på morgonen
0,19	nästa måndag behöver jag flyga från Pittsburgh till Fort Worth så att jag anländer före klockan tio

FIGURE 4: Dendrograms for two different clustering methods, sentences 1 and 5.



Semiotics at Work: Technical Communication and Translation in a Multilingual Corporate Environment

Arne Larsson, Espoo
Magnus Merkel, Linköping

Abstract

In the paper an attempt is made to find a unifying approach to the study of the translator's praxis, assuming that translation is guided by certain, recognizable, semiotic processes. Computational, corpus-based methods intended to aid in the research of large text bases are introduced. Alignment of text segments from files in different languages contained in a corpus, where these text files are known to be mutual translations is described. Text encoding in order to allow comparison of the results of translation studies performed by different scholars is also demonstrated. One goal is to establish qualitative and quantitative variables, on the sentential as well as the textual level, which would permit generalizations about the concrete procedures performed by professional translators in authentic work situations, e.g. in multi-lingual corporate environments.

Empirical, descriptive methods

Today large amounts of texts sit on the hard disks of computers in companies and organizations, but exact, empirical, detailed, descriptive information telling us what translators actually do when they translate is not abundant. A natural solution to this dilemma is the collection of evidence from existing texts included in aligned bilingual corpora. The purpose of text alignment is to establish *version complexes*[1], i.e. sets of corresponding elements in the source and target texts.

The Alignment Tool (LinAlign)

At the Department of Computer Science at Linköping university an Alignment program was developed in 1993[2]. The program (called LinAlign) creates translation memories of a source and target text, that is, it links a sentence in the original with a corresponding sentence in the target document. There are different techniques to accomplish the alignment of segments. Most notable has been the statistical approach,

[1]This concept was introduced by Wollin (1981), followed by Platzack (1983).
[2]The major part of the programming has been done by Bernt Nilsson.

which the LinAlign tool also adheres to. The best-known statistical algorithm is the one developed by Gale & Church (1991). LinAlign uses a much simpler method than Gale's & Church's program, but in a test described below its performance is equal to theirs, if not better.

The algorithm is based on three assumptions of the source and target texts.

1. The source and target texts are similarly ordered.
2. If two sentences in one text are combined to one sentence in the other text, it is always adjacent sentences that have been joined.
3. The alignment is based on paragraph and sentence lengths (number of characters).

Apart from 1–1 relations, LinAlign also handles 1–2 and 2–1 relations (one source sentence – two target sentences, two source sentences – one target sentence).

Below is a sample of the output from the LinAlign program:

```
f1:21.1 Specify the amount of time before you receive
  messages about printer problems.
f2:21.1 Ange efter hur lång tid ett meddelande rörande
  skrivarproblem ska visas.

f1:22.1 Select the default printer.
f2:22.1 Välj standardskrivaren.

f1:23.1 The following sections explain how to perform
  each of these tasks.
f2:23.1 Följande avsnitt förklarar hur du vidtar dessa
  åtgärder.
```

The code before each segment gives information about each document and its respective paragraph and sentence ordering. In the example above `f1:21.1` indicates that the segment is taken from the target language (f1), the 21st paragraph (`:21`) and the first sentence of that paragraph (`.1`)

To illustrate the way the algorithm works when there are an unequal number of sentences in the corresponding paragraphs, let us consider the example below. The first part of the example is a help text that is described if LinAlign is run in Debugging mode.

```
searching for sentences to join...
f1:444.1 & f2:444.1 + f2:444.2 = 6
f1:444.2 & f2:444.2 + f2:444.3 = 38
 -> f2:444.1 + f2:444.2
f1:444.1 To cancel a selection, you can use mouse or
  keyboard techniques, or the Select Files command.
f2:444.1 Du använder musen eller tangentbordet för att
  avbryta markeringar.
f2:444.2 Du kan också använda kommandot Markera filer.
```

```
f1:444.2 You can cancel one selected file or a group of
   selected files.
f2:444.3 Du kan avbryta markeringen av såväl en enskild
   fil som hela grupper.
```

The example describes paragraph 444 of a particular translation text, showing both the source and target texts. In the English text there are two sentences, but in the Swedish there are three target sentences. The help text above the translation pairs helps us to understand the way the algorithm works. The program has to determine whether it is the first and the second Swedish sentence that should be joined as the translation of the first English sentence, or if the second and third Swedish sentences should be taken as the translation of the second English sentence. Based on the number of characters in the sentences the different options are compared and the one with the closest match is selected. In the example above LinAlign values the cost of regarding the first and second Swedish as the translation of the first English sentence as the cheapest alternative (i.e. the shortest "sentence distance") and therefore these two sentences are joined in a 1–2 relation.

The sentence distance measure is computed by the following formula:

sentence distance = P(l1 + l2 – olfactor)

where P is the proportional measurement of the two texts,
l1 is the length of sentence 1 measured in characters,
l2 is the length of sentence 2 measured in characters,
and *olfactor* is the overlap factor that is used to capture the fact that two sentences joined together becomes longer than a corresponding single sentence (default value is 15)

Alignment test

A test of the LinAlign tool when run on a manually translated text, showed that out of 624 sentences, it failed on only four sentences. The test was done on an English-Swedish corpus consisting of a chapter from a manual for a computer program. Church & Gale (1991) reported that their tool when tested on a similarly sized English-French material failed on 22 sentences out of 621. It is of course impossible to draw any conclusions on the quality of the tools from such small and different test materials.

However, one interesting factor found when we analyzed the source text with a tool for measuring recurrence was that 23 sentence types were repeated between 2 to 19 times in the text. (The Recurrence Analyzer is developed at the same department as LinAlign and results from analyses of technical documentation can be found in Merkel (1992).) A recurrence test on the target text revealed that out of these 23 sentence types 20 had

been translated with consistent translations. The three sentence types (all with the frequency 2) that had different translations could have had consistent translations, without impeding readability. In the following example, the three source sentences are shown together with the alternative target sentences.

Recurrent source sentences with different translations:

1. The options available in the dialog box below may vary, depending on the network you are using.
1a. Vilka alternativ som finns i dialogrutan nedan beror på vilket nätverk du använder.
1b. Tillgängliga alternativ i dialogrutan beror på vilket nätverk du använder.
2. Select the port you want to assign the printer to.
2a. Markera den port du har anslutit skrivaren till.
2b. Välj vilken port du vill ansluta skrivaren till.
3. Select the port you want to use.
3a. Välj den port du vill använda.
3b. Markera den port du vill använda.

In other words, there was nothing special in the context that demanded variation. It was just what the translator had chosen at a certain point in the translation process, unaware of the fact that the exact sentence occurred at a different text segment.

It would be interesting to take this analysis methodology one step further by analyzing the variation in the target text on a much larger scale. For example, how widespread are these phenomena in different types of text? Furthermore, to what extent can segments with explicit cohesive markers (Halliday & Hasan 1976, Källgren 1979) be reused in different local contexts in, for example, technical documentation and legal treaties? And will consistent use of memory-based translation make certain translations "worse" in the aspects of text binding? These are questions that can only be answered if huge masses of translated texts are aligned and analyzed in detail.

Language independence

Two text fragments from a technical manual in Finnish and Swedish were also aligned using the LinAlign tool, thus demonstrating the language independence of this statistical method. The actual aligned segments are similar to the English-Swedish ones above. For reasons of space, they will not be reproduced here.

Other alignment methods

Morphology-based alignment is used in a computer-based workstation for the lexicographer (Picchi et al. 1992). Aligned parse trees from a

dependency grammar parser were proposed for machine translation purposes in Sadler (1989). A method for alignment of words as well as sentences was presented in Kay & Röscheisen (1993).

Text encoding

When the alignment of the Finnish and Swedish texts was completed, the text fragments were marked up according to the function of the *primary sentential constituents*. The following abbreviations were used (for additional details and examples, see Platzack 1983: 249 ff, Larsson 1993):

FV	Inflected verb (finit verb)	**OO**	Direct object
IA	Content adverbial (innehållsadv)	**SP**	Predicate NP (subjektiv predikatsfyllnad)
IO	Indirect object	**SS**	Subject
IV	Infinitives (infinit verb)	**SA**	Added clause

Operations performed by the translator

Eight different types of operations, which the translator may apply were identified by Wollin (1981), viz. addition, convergence, deletion, divergence, functional modification, mixing, structural identity, and transposition (Platzack 1983: 256 ff.). Four of these operations were used in the text fragments (1) and (2) below:

(1) *Structural identity*

Ennen asennustyön aloittamista	**IA**	**IA**	Innan installationsarbetet påbörjas
tehdään	**FV**	**FV**	utarbetas
asennussuunnitelma	**OO**	**OO**	installationsplan
noudattaen tässä kirjassa ja asennettavien laitteiden käyttökirjoissa annettuja ohjeita.	**IA**	**IA**	med ledning av föreliggande handbok och anvisningarna i handböckerna för den teleutrustning som skall installeras.

(2) *Addition (SA), functional modification and transposition (OO=>SS)*

Apuna suunnitelman teossa	**IA**	**SS**	Bifogade planeringsblanketter
voidaan	**FV**	**FV**	kan
käyttää	**IV**	**IV**	användas
liitteenä olevia suunnitelmalomakkeita.	**OO**	**IA**	som hjälp vid utarbetande av planen
		SA	(jfr bilagorna).

An important distinction is the one between obligatory versus optional operations. Here, operations that are absolutely necessary for the formation of *grammatical structures* in the target language are called

obligatory (e.g. insertion of articles and prepositions in the Swedish, which are non existing in the Finnish; 'correct' order between primary constituents and word order within constituents), otherwise they are called *optional.*

Professional translators' performance

Focusing on the syntactic level alone will render a somewhat shallow picture of the complicated processes of translator performance.[1] Therefore, certain *textual variables* were used to supplement the *sentential variables* (i.e., the operations on version complexes outlined above) in order to achieve enhanced explanatory power.

The following textual variables were chosen, because contrastive studies of Finnish and Japanese, vs. Anglo-American writers indicate specific problematic differences (Kohl et al. 1993, Mauranen 1992, Ventola 1992) with respect to 1) reference items, reference chains and text coherence; 2) theme and rheme, thematic progression, choice of connectors; 3) reflexive expressions, 'text about text'; 4) signals of propositional relationships: making the point, stating opinions; 5) types of strategic moves; 6) culture as discourse. Similar differences can be expected between other language pairs as well.

The choice was also guided by the existence of methods for the successful study of the variables involved (Källgren et al. 1977, Källgren 1979, Sigurd 1987).

Linguistic preferences govern translation

As can be seen from the semantic network (table 1), no less than six of the referents are implicit in the Finnish source text (marked *if* in P5, P6, P7, P10, P11 and P13), versus two implicit items in the Swedish target (marked *is* in P10 and P14, one of which is the predicate *är* 'is, are').

The Finnish tendency toward implicitness together with late introduction of referents (Mauranen 1992: 109) explain fairly well, why the translator has made use of the structure changing operations *addition, functional modification* and *transposition* in (2). The reason for these manipulations is that the target language community requires more explicit referents and prefers an earlier introduction of these.

---

[1]Platzack (1983:266) stressed the need for obtaining additional information concerning the mutual influences between various properties of the languages involved in the translation, and the frequency of application of different operations.

Today, we have evidence to the effect that the appropriate use of textual connectors will make a text easier to read, more logical, more convincing, and add to the writer's credibility (Mauranen 1992: 187). Computer tractable, well structured thesauri will facilitate decisions about *what* to actually add in order to achieve enhanced *connectivity*.

Translation as choice and change

Text-linguistic methods will not only provide explanations of the translators use of certain operations, but also facilitate a systematic approach to active text planning and organization during the creative phase of writing or translation.

The underlying *elementary propositions* for the above technical manual fragments can be presented as a *semantic network*[1] where paths representing various, *logically possible* texts involving the factoms can be drawn.

TABLE 1: Semantic network for the technical manual text fragments (1) and (2).

		1	2	3	4	5	6	7	8	9	10	11	12	13
P1	Man börjar installations-arbetet (IA)		fs ⇳ ⇳	①										
P2	Man gör en installations-plan (IP)	●	⇳ ⇳ ⇳	fs ⇳ ⇳										
P3	P2 föregår P1	fs	(fs)	(fs)①										
P5	Handboken (HB)*if* ger ledning		●		fs		(f) ⇳ ⇳							
P6	HB har anvis-ningar (A)*if*					fs	⇳ ⇳	①						
P7	HB/A gäller (tele)*if*utrust-ning (TU)						fs		①					
P8	Man installerar TU							fs		①				
P9	P2 föregår P8								fs	①				
P10	(HB)*is*/*if* har planerings-blanketter (PB)				●					s				(s) ⇳⇳
P11	(PB)*if* ger hjälp					①				f	s		(fs)	⇳
P12	Hjälpen gäller P2						①				f	s	⇳⇳	⇳⇳
P13	(HB)*if* har bilagor										①	f	⇳ ⇳s	⇳⇳
P14	Bilagorna (är)*is* PB										①		f	(s)

f	Finnish text fragment	*if*	implicit in the Finnish text	
s	Swedish text fragment	*is*	implicit in the Swedish text	
⇳	Reader (translator) needs to backtrack			
①	Suggested paths through the network	●	The point we want to make	

[1]See Källgren (1979), Larsson (1993), Sigurd (1977, 1987), Wintraecken (1990) for additional details.

Building new texts using paths through a network

In planning and creation, paths may be chosen, which are considered optimal for the communicative task at hand. Unnecessary propositions may be left out. Below, the symbol ① represents suggested paths, and the black symbol ❶ the points we want to make. Two new versions in Swedish and two in English of the technical manual fragments are presented with the propositions reordered to avoid backtracking, and explicit referents inserted:

First draft (Swedish):
(P2)Gör en installationsplan (P5)med ledning av anvisningarna, (P3, P1)innan du börjar arbeta med installationen. (P10)Planeringsblanketterna (P11)hjälper dig (P12)att göra upp planen. (P6)Anvisningarna finns i handboken (P7)för teleutrustningen som (P9)skall (P8)installeras. (P13, P14)Se bilagorna.

A more official version may be needed (changes relative to the first draft are marked using underlining.):
(P2)Gör alltid upp en installationsplan (P5)med ledning av gällande anvisningar, (P3, P1)innan arbetet med installationen börjar. (P10)Nokia har tagit fram planeringsblanketter (P11)som hjälp_ (P12)vid upprättandet av installationsplanen. (P6)Anvisningarna finns i handboken (P7)för teleutrustningen som (P9)skall (P8)installeras. (P13, P14)Se bilaga 14-21.

We might even dare an attempt at an English version:
(P2)Make a plan of the installation (P5)according to the instructions, (P3, P1)before you start working on the installation. (P10)Forms (P11)help you (P12)make the plan. (P6)Instructions are in the manual (P7)for the telecommunications equipment (P9)under (P8)installation. (P13, P14)Please, refer to the Appendix.

Which we might want to edit later:
(P2)We strongly recommend, that you create a plan of the facilities (P5)according to the instructions given by the manufacturer, (P3, P1)before you start working on the setup of the equipment. (P10)Forms (P11), which help you (P12)with the installation planning, (P13, P14)are provided in Appendix 14-21. (P6)Instructions are in the manual (P7)for the telecommunications equipment (P9)to be (P8)installed.

Conclusions

Aligned bilingual corpora can tell exactly *what* the translator does in terms of concrete syntactic operations. Text-linguistic methods explain *why* these operations were used. Moreover, operational and text-linguistic approaches facilitate systematic planning and organization of texts in a multi-lingual corporate environment. As a result, these methods form a useful complement to the goal oriented principles of "translatorisches Handeln" and "skopos" (Holz-Mänttäri 1982, Vermeer 1989). Future work will be focused on 1) automatic text alignment, 2) automatic tagging/parsing of aligned texts, 3) application of international standards, e.g. SGML, 4) tools for translators and writers.

References

Chesterman, A. 1989. *Readings in Translation Theory.* Loimaa.

Gale W. A. and K. W. Church. 1991. *A program for aligning sentences in bilingual corpora.* In: PROCEEDINGS FROM ACL–91, pp 177–184, Berkeley.

Halliday M. A. K. and Ruqaiya Hasan. 1976. *Cohesion in English.* Longman, London.

Holz-Mänttäri, J. 1982. *Vom translatorischen Handeln.* Turku.

Kay, M. and M. Röscheisen. 1993. *Text-Translation Alignment.* COMPUTATIONAL LINGUISTICS, Volume 19, Number 1, pp. 121–142, MIT Press.

Kohl J., R. Barclay, Th. Pinelli, M. Keene, J. Kennedy. 1993. *The Impact of Language and Culture on Technical Communication in Japan.* TECHNICAL COMMUNICATION, First Quarter 1993.

Källgren, G. 1979. *Innehåll i text.* ORD OCH STIL 11. Studentlitteratur, Lund.

Källgren, G., B. Sigurd and M. Westman. 1977. *Experiment med text.* Akademilitteratur, Stockholm.

Larsson, A. 1993. *Semiotic Aspects of Technical Translation and Communication.* In FACKSPRÅK OCH ÖVERSÄTTNINGSTEORI. VAKKI-SEMINARIUM XIII, Vasa universitet, Institutionen för språk, Vasa (forthcoming).

Mauranen, A. 1992. *Cultural differences in academic rhetoric. A textlinguistic study.* Dissertation, Birmingham.

Merkel, Magnus. 1992. *Recurrent Patterns in Technical Documentation.* RESEARCH REPORT, DEPT. OF COMPUTER AND INFORMATION SCIENCE, Linköping University.

Platzack, Ch. 1983. *Sex översättningar till svenska av Lewis Carolls "Alice in Wonderland".* In Engwall, G. and af Geijerstam, R. (eds), *Från språk till språk,* Studentlitteratur, Lund.

Picchi, E., C. Peters and E. Marinal. 1992. *The Pisa Lexicographic Workstation: The Bilingual Components.* In Tommola, H., K. Varantola, T. Salmi-Tolonen and J. Schopp (eds), *EURALEX '92 PROCEEDINGS I–II.* Papers submitted to the 5th EURALEX International congress on Lexicography. Tampere.

Sadler, V. 1989. *Working with Analogical Semantics: Disambiguation Techniques in DLT.* Dordrecht.

Sigurd, B. 1987. *High resolution referent grammar (HRG) in analysis and generation of text.* In WORKING PAPERS, Lund University, Dept. of Linguistics, Lund.

Ventola, E. 1992. *Reference and Theme and the Interplay of these two textual systems.* In Nyysönen, H. and L. Kuure (eds), *Acquisition of language – acquisition of culture.* AFINLA YEARBOOK. Jyväskylä.

Vermeer, H. J. 1989. *Skopos and commission in translational action.* In Chesterman, A., (ed.).

Wintraecken, J. 1990. *The NIAM information analysis method: theory and practice.* Dordrecht.

Wollin, L. 1981. *Svensk latinöversättning. I. Processen,* Dissertation, Lund.

Pragmatics Through Context Management

Joakim Nivre
Göteborg

Abstract

Pragmatically based dialogue management requires flexible and efficient representation of contextual information. The approach described in this paper uses logical knowledge bases to represent contextual information and special abductive reasoning tools to manage these knowledge bases. One of the advantages of such a reasoning based approach to computational dialogue pragmatics is that the same rules, stated declaratively, can be used both in analysis and generation.

1 Introduction

The purpose of the present paper is to illustrate an approach to computational dialogue pragmatics that has been developed within the ESPRIT project PLUS (A Pragmatics-based Language Understanding System, ESPRIT P5254).[1] The purpose of this project was to build a dialogue system for information-seeking (in the domain of the Yellow Pages), and the basic idea of the project was to improve on existing systems by making heavy use of pragmatics, i. e. by enabling the system to make systematic use of contextual information in interpretation, planning and response generation.[2]

In the PLUS system, contextual information is stored in a set of **knowledge bases**, represented as logic programs. The most important of these knowledge bases is the Discourse Model, which contains the information derived from the ongoing dialogue. The process of dialogue management, i. e. of interpreting the user's contributions, planning the system's actions, and generating appropriate responses, is then conceived as the process of maintaining the contextual knowledge bases (in particular, the Discourse Model) through such knowledge base operations as querying, updating, checking (and restoring) consistency, etc. From a computational point of view, then, the management of dialogue can be seen as a side effect of the process of maintaining the contextual

---

[1]The PLUS consortium consists of CAP GEMINI INNOVATION (France), CAP debis (Germany), ITK (The Netherlands), Omega Generation (Italy), UMIST (England), LIMSI (France), University of Bristol (England), University of Göteborg (Sweden). The Swedish part of the project has been funded by Teleannons AB and NUTEK.

[2]For an overview of the PLUS project, see Black et al (1991).

knowledge bases (cf. Gallagher et al 1992). In short, **dialogue management** is reduced to **context management**.

In what follows, I will attempt to illustrate the PLUS approach to dialogue management by means of a few simple examples. I will begin by giving a brief overview of the PLUS system (section 2). After that, I will present an extremely simplified version of the Discourse Model, containing only the features that are absolutely necessary to illustrate the basic process of dialogue management (section 3). Finally, in section 4, I will try to show how the process of dialogue management can be implemented through the management of contextual knowledge bases using a set of special abductive update procedures.

Needless to say, the work presented here draws extensively on collaborative work within the PLUS project. Most of these debts are acknowledged through references cited throughout the text. In addition, the paper is based on (so far unpublished) work carried out together with Jens Allwood and Björn Beskow after the completion of the PLUS project. It is also worth mentioning that the discussion of PLUS work contains many simplifications and omissions, mainly due to limitations of space. For a more complete account of the PLUS approach to pragmatics and dialogue management, the reader is referred to Bunt and Allwood (1992), Nivre et al (1992), Bunt et al (1992) and Bego et al (1992).

2 Overview of the PLUS System

The PLUS system is meant to be a prototype for an information dialogue system using typed terminal input. It consists of three main components:

- A Dialogue Manager (DM)
- A Natural Language Engine (NLE): parser and surface generator
- An application database (the Yellow Pages for the prototype)

The Dialogue Manager is the heart of the system. It receives parsed user input from the NLE, it queries the application database and it generates system responses which are converted into output strings by the surface generator. The Dialogue Manager itself can be broken down into three components:

- A World and Application Model
- A Discourse Model
- A Knowledge Base Management System (KBMS)

The World and Application Model is a static knowledge base containing general world knowledge as well as information about the application

database. The Discourse Model is a dynamic knowledge base which is built up and modified during the course of a dialogue. The KBMS, finally, is a set of procedures for managing the knowledge bases (querying, updating, consistency checking, etc.).

The tasks of the Dialogue Manager include interpretation of user contributions (given the output of the parser), planning of system actions (such as querying the database), and generation of system responses (which are fed to the surface generator). These tasks are referred to collectively as **dialogue management**.

The pragmatics-based approach of PLUS entails that dialogue management be based heavily on contextual information. The contextual information includes information in the World and Application Model (static context) as well as information in the Discourse Model (dynamic context). In this paper, I will concentrate exclusively on the use of information in the Discourse Model.

3 A Simple Discourse Model

The contextual knowledge bases in the PLUS system are implemented as logic programs. In this section, I will outline a very simple Discourse Model to illustrate the basic principles of this approach. (For the specification of the actual PLUS Discourse Model, see Bunt et al 1992.)

3.1 Dialogue History

In order to keep track of the dialogue history, we need to record (at least) the following aspects of each contribution (or "utterance") in the dialogue:

- Contributor (or "speaker")
- Verbatim form
- Grammatical structure
- Semantic (propositional) content
- Communicative function (illocutionary force)

These aspects can be specified by simple facts of the following form:

(1)
```
contribution(N,Agent).
verbatim(N,String).
gram_structure(N,Structure).
prop_content(N,P).
comm_function(N,CF).
```

The simple atomic formulas in (1) can be taken to represent a context where the Nth contribution to the dialogue was made by `Agent`, having the verbatim form `String`, the grammatical structure `Structure`, the propositional content `P`, and the communicative function `CF`.[1]

3.2 Attitudes

Both the interpretation of user contributions and the planning of system actions (including dialogue contributions) normally require reasoning about propositional attitudes, such as beliefs and intentions, attributed to the user and the system. For example, a context where the system believes some proposition `P`, where the user doesn't know whether `P`, and where the system wants the user to believe `P` can be represented as follows:

(2)
```
bel(system,P).
¬know_wh(user,P).
want(system,bel(user,P)).
```

3.3 Rules and Constraints

So far, we have only considered simple facts (i. e. atomic formulas and their negations). However, the Discourse Model must also contain **rules** (universally quantified conditionals) defining relations between different types of contextual information. For example, rules of the following kind may be proposed to capture the relations between communicative functions (such as `state` and `ask`) and the propositional attitudes underlying these communicative functions:

(3)
```
comm_function(N,state) ← contribution(N,A),
                         gram_structure(N,S),
                         ¬interrogative(S),
                         prop_content(N,P),
                         bel(A,P),
                         want(A,bel(B,P)),
                         interlocutor(A,B).
```

(4)
```
comm_function(N,ask)   ← contribution(N,A),
                         prop_content(N,P),
                         ¬know_wh(A,P),
                         want(A,know_wh(A,P)),
                         interlocutor(A,B).
```

[1] In addition to these aspects of contributions, the real PLUS Discourse Model also includes information about such things as topic, focus and discourse referents (cf. Bunt et al 1992).

The first rule can be read as saying that a non-interrogative contribution with propositional content `P` is a statement (or has the communicative function `state`) if the contributor believes `P` and wants the interlocutor to believe `P`. In the same vein, the second rule says that a contribution with propositional content `P` is a question (or has the communicative function `ask`) if the contributor doesn't know whether `P` but wants to know whether `P`.

The rules in (3–4) define positive relationships between different types of facts (i. e. if the clauses in the antecedent are true, then the consequent is also true). However, we also have a need for negative **constraints** stating that a certain conjunction of clauses cannot be simultaneously true in the knowledge base. In order to do this, we introduce a special predicate `inconsistent`, occurring in the consequent of such constraints. For example, the following are constraints saying that the Discourse Model is inconsistent if it contains a contribution without a verbatim form, a contribution without a grammatical structure, a contribution without a propositional content, or a contribution without a communicative function.

```
(5) inconsistent    ←  contribution(N,Agent),
                       ¬verbatim_form(N,String).

(6) inconsistent    ←  contribution(N,Agent),
                       ¬gram_structure(N,Structure).

(7) inconsistent    ←  contribution(N,Agent),
                       ¬prop_content(N,P).

(8) inconsistent    ←  contribution(N,Agent),
                       ¬comm_function(N,CF).
```

The joint effect of these constraints is that every contribution is required to have a verbatim form, a grammatical structure, a propositional content as well as a communicative function in order for the Discourse Model to be consistent.

3.4 Reasoning Tools

The Discourse Model (and the other contextual knowledge bases) in the PLUS system are implemented as logic programs. The KBMS tools developed for the management of these knowledge bases support standard operations of asserting and retracting facts from a knowledge base, querying the knowledge base (to find out if a goal is a consequence of the knowledge base) and checking that the knowledge base is consistent.

In addition, special procedures for abductive updates have been developed and implemented (cf. Guessoum and Gallagher 1992). The two main predicates of these procedures are `insert` and `delete`, which can be characterised in the following way:

- The call `insert(P,KB,Trans)` returns the list `Trans` of transactions (asserts and retracts) that would make `P` a consequence of the knowledge base `KB`.

- The call `delete(P,KB,Trans)` returns the list `Trans` of transactions (asserts and retracts) that would ensure that `P` is no longer a consequence of the knowledge base `KB`.

A special use of these update procedures is the deletion of the special predicate `inconsistent` from the Discourse Model. As we will see in the next section, the insertion of new facts into the Discourse Model will often violate constraints in the Discourse Model, temporarily giving rise to states where the Discourse Model is "inconsistent" in the sense that the formula `inconsistent` is a consequence of the knowledge base. The normal way for the system to deal with this problem is to attempt to remove the inconsistency through an abductive update, i. e. by deleting the formula `inconsistent`. This move may then introduce new inconsistencies which have to be deleted through further updates and so on.

4 Dialogue Management

By means of a few simple examples, I will now try to outline how the process of dialogue management can be implemented through the use of abductive update procedures to maintain a contextual knowledge base of the kind described in the preceding section. I will subdivide the process of dialogue management into three subprocesses:

- Interpretation of user contributions
- Planning of system actions
- Generation of system responses

It is important to note, however, that this is an analytic division which is made primarily for purposes of exposition and which does not correspond in any straightforward way to "system modules". The basic computational process is the same in all three cases, and many of the rules and constraints involved apply across several subprocesses.

4.1 Interpretation

Whenever the user types some input (and hits the return key) the Discourse Model is updated by inserting facts of the following form (where N is some number and String is the verbatim form of the user input):

(9) `contribution(N,user).`
`verbatim_form(N,String).`

Since there are no rules in the Discourse Model which allows the system to prove such facts, they will simply be asserted into the Discourse Model. (In other words, user contributions and their verbatim form can only be observed, they can never be inferred, neither deductively nor abductively.)

Asserting these facts into the Discourse Model will make the knowledge base inconsistent, because of the following constraints (cf. section 3.3):

(6) `inconsistent ← contribution(N,Agent), ¬gram_structure(N,Structure).`

(7) `inconsistent ← contribution(N,Agent), ¬prop_content(N,P).`

(8) `inconsistent ← contribution(N,Agent), ¬comm_function(N,CF).`

In order to make the Discourse Model consistent again, the system must prove that the Nth contribution from the user has a certain grammatical structure Structure, a certain propositional content P, and a certain communicative function CF. In a PLUS-like system, the first goal will be resolved by calling the parser, which will instantiate the variable Structure to a grammatical feature structure containing, among other things, a compositional semantic analysis of the input. From this semantic analysis, together with contextual information already stored in the Discourse Model, the system will then attempt to derive a propositional content P for the contribution in question.

Let us now consider in a little more detail how the analysis of communicative function can proceed. In order to make the Discourse Model consistent again, the system must prove the goal comm_function(N,CF), for some CF. As noted above, the Discourse Model contains rules relating to communicative function, such as the following (cf. section 3.3):

```
(3) comm_function(N,state) ← contribution(N,A),
                             gram_structure(N,S),
                             ¬interrogative(S),
                             prop_content(N,P),
                             bel(A,P),
                             want(A,bel(B,P)),
                             interlocutor(A,B).

(4) comm_function(N,ask)   ← contribution(N,A),
                             prop_content(N,P),
                             ¬know_wh(A,P),
                             want(A,know_wh(A,P)),
                             interlocutor(A,B).
```

The important point about these rules is that the attitude goals cannot be proven deductively in the Discourse Model but have to be abduced (if they are compatible with the rest of the system's knowledge). For example, when the system tries to insert that a certain user contribution is a statement, the update procedures will propose as a possible transaction to assert (i. e. to abduce) that the user believes the propositional content and wants the system to believe the same thing.

If there is no conflict with the rest of the information in the Discourse Model, these attitude facts can be assumed, representing an interpretation of the communicative function of the user's contribution. If there **is** conflicting information (the system may know on other grounds that the user does not believe the propositional content), then the abduction is blocked and the system will continue to search for another interpretation. If all interpretations are blocked in this way, the system will be forced to initiate a repair (asking the user what she meant, whether she has changed her mind, etc.).

4.2 Planning

As we have seen above, the interpretation of a user contribution will typically result in the abduction of a set of user attitudes (beliefs, goals, etc.) in order to restore the consistency of the Discourse Model. However, the addition of these user attitudes will normally generate new inconsistencies, because of constraints relating user attitudes to system attitudes. For example, if the system is meant to be ideally cooperative, then it seems reasonable to require that any goal of the user is also a goal of the system (with certain restrictions that I will not go into here). A cooperativity constraint of this kind would have the following form:

```
(10) inconsistent ← want(user,P),
                    ¬want(system,P).
```

The presence of this constraint in the Discourse Model will guarantee that as soon as the system has inferred that the user has a certain goal, the system will try to insert that it has the same goal. This insertion will generate further system goals, such as the goal to find a certain piece of information in the database and give it to the user, etc. In this way, planning of system actions can be carried out by the same basic process of maintaining the Discourse Model through abductive updates that was used for the interpretation of user contributions.

4.3 Generation

A very basic requirement on a cooperative dialogue system, is that it should generate a response to every contribution from the user. This requirement can be implemented by adding the following constraint to the Discourse Model:

(11) `inconsistent` ← `contribution(N,user),`
`¬contribution(N+1,system).`

This constraint will ensure that the addition of a user contribution to the Discourse Model is always followed by the addition of a system contribution. Moreover, once the new contribution has been added, the constraints in (5–8) will come into play again and will drive the generation process further until a fully specified system contribution has been generated. In this way, the same constraints are used to drive both interpretation and generation.

Furthermore, the rules relating to communicative functions can also be exploited both in interpretation and in generation. Suppose, for example, that the Nth contribution has been interpreted as a question by the user with propositional content `P`. Suppose further that the system knows `P` to be the case (`P` may be a fact in the application database, such as the fact that a certain company has a certain phone number), and that we have the following (not too implausible) rules in the Discourse Model:

(12) `want(A,bel(B,P))` ← `want(A,know_wh(B,P)),`
`P.`

(13) `bel(system,P)` ← `P.`

We can then prove the following facts in the Discourse Model:

(14) `bel(system,P).`
`want(system,bel(user,P)).`

Now, given the constraint in (11), the system will sooner or later be forced to add a new system contribution to the Discourse Model:

(15) `contribution(N+1,system).`

And in order to satisfy the constraint in (8), the system must then be able to prove `comm_function(N+1,CF)` for some `CF`. If we consider the rules (3–4) and the facts in (14), we see that it may be possible for the system to prove `comm_function(N+1,state)`, thus generating an answer to the question, but not `comm_function(N+1,ask)`, which would result in a new question. Without going into all the details, the important point is that the same rules and constraints apply both in interpretation and generation.

5 Conclusion

In the present paper, I have tried to illustrate a certain approach to dialogue management based on knowledge base representations of contextual information and reasoning tools incorporating abductive updates. There are still many open problems in relation to the use of abductive reasoning, but I nevertheless think that the approach presented here is interesting enough to merit further attention. Hopefully, I am not alone in thinking this.

References

Bego, H. (ed.), B. Beskow, H. Bunt, A. Derain, L. Horel, K. Jokinen, W. Kraaij, D. Pernel and G. Tabuteau. 1992. *Pragmatic Knowledge in PLUS. Part III: Pragmatic Rules*, Review Report. ESPRIT-II-P5254 PLUS.

Black, W. J. (ed.), J. Allwood, H. C. Bunt, F. J. H. Dols, C. Donzella, G. Ferrari, J. Gallagher, R. Haidan, B. Imlah, K. Jokinen, J.-M. Lancel, J. Nivre, G. Sabah and T. J. Wachtel. 1991. *A Pragmatics-based Language Understanding System.* In INFORMATION PROCESSING SYSTEMS. RESULTS AND PROGRESS OF SELECTED PROJECTS IN 1991, ESPRIT, Brussels.

Bunt, H. and J. Allwood. 1992. *Pragmatics in PLUS.* Deliverable D2.3., ESPRIT-II-P5254 PLUS.

Bunt, H. and C. Godin, (eds.), K. Jokinen, W. Kraaij, R. Meyer, J. Nivre and D. Pernel. 1992. *Pragmatic Knowledge in PLUS. Part II: Discourse Model.* Review Report, ESPRIT-II-P5254 PLUS.

Gallagher, J., A. Guessoum, R. Haidan and R. Meyer. 1992. *Reasoning Tools and Pragmatic Reasoning.* Internal Report, ESPRIT-II-P5254 PLUS.

Guessoum, A. and Gallagher, J. 1992. *The PLUS Update Procedures.* Internal Report, ESPRIT-II-P5254 PLUS.

Nivre, J. (ed.), A. Cavalli, A. Guessoum, T. Lager, R. Meyer and N. Underwood. 1992. *Pragmatic Knowledge in PLUS. Part I: World and Application Model.* Review Report, ESPRIT-II-P5254 PLUS.

On GB Parsing and Semantic Interpretation

Torbjørn Nordgård
Bergen

Abstract

The paper shows how sentences containing scope ambiguities can be assigned syntactic and semantic structures by means of sloppy deterministic processing techniques only. The semantic framework is Discourse Representation Theory, and the sloppy deterministic parser is described in Nordgård (1993). Of primary concern for the article is the transition from syntactic structures to discourse representation structures (DRSs).

Introduction

In Discourse Representation Theory (DRT) nominal constituents in a syntactic tree are substituted by variables when the tree is translated to a semantic expression which is interpretable wrt. a model, cf. Kamp & Reyle (1992). The variables and the "reduced" trees are crucial parts of *Discourse Representation Structures* (DRSs). Consider example (1) and the DRS (2), which results from (1), assuming that sentence (1) is the first utterance in a context:

(1) Peter likes Mary

(2)

x, y
Peter(x)
Mary(y)
x likes y

The content of the box in (2) constitutes a DRS. The first line is the *variable list*, and the other expressions are the *conditions* of the DRS. The expression *x likes y* is a shorthand for the syntactic representation of (1) where *x* and *y* have replaced **Peter** and **Mary**, e.g. [$_S$ x [$_{VP}$ [$_V$ likes] y]]. The variables introduced in a DRS δ have scope over expressions inside δ and all other DRSs "contained" in δ.

Another example is given by (3) and the corresponding DRS in (4)[1].

(3) A man smokes

(4)

x
man(x)
smokes(x)

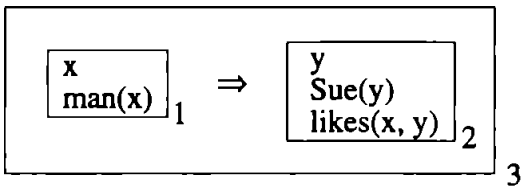
[1]The reader is referred to Kamp & Reyle (1992) for elaboration of this analysis of indefinites.

Universal Quantification

Universal quantification is exemplified in (5). According to Kamp & Reyle we want a DRS like (6):

(5) Every man likes Sue

(6)



Note that this representation consists of three distinct DRSs, as indicated by the subscripts 1–3. DRS_1 and DRS_2 are subordinate to DRS_3, which is easily seen from the box notation. One might wonder how condition *likes(x,y)* in DRS_2 has access to variable *x* in DRS_1. Kamp & Reyle state that a condition α in some DRS_i has access to variables declared in some DRS_{ii} if DRS_i is subordinate to DRS_{ii}, or DRS_i and DRS_{ii} are connected by "⇒", DRS_i on the righthand side of "⇒" (this is a simplification of the terms 'subordinate' and 'accessibility'; see Kamp & Reyle (1992) for a detailed exposition).

The translation of universally quantified NPs is performed by a construction rule:

(7)

Triggering configuration $[_\gamma [_{NP} [_{Det}$ every $]\ [_{N'} [_N\ \alpha\ ]]]\ [_\phi \ldots\ ]]$ or $[_{\varphi P} [_{\varphi'} [_\varphi \ldots\ ]\ [_{NP} [_{Det}$ every $]\ [_{N'} [_N\ \alpha\ ]]]]]$

$\varpi \supseteq \varpi'$, $\varpi \in$ **CON/DRS$_0$:**

Introduce in CON/DRS$_0$: New condition $DRS_1 \Rightarrow DRS_2$ where DRS_1 and DRS_2 are empty

Introduce in U/DRS$_1$: new discourse referent *u*

Introduce in CON/DRS$_1$: $\alpha(u)$

Introduce in CON/DRS$_2$: New condition χ, where χ is the result of substituting *u* for $[_{NP} [_{Det}$ every $]\ [_{N'} [_N\ \alpha\ ]]]$ in ϖ.

Delete ϖ from DRS$_0$.

CON/DRS$_n$ is an abbreviation for the set of conditions in DRS_n, and *U/DRS$_m$* is a shorthand for the universe of DRS_m, i.e. the variables declared in DRS_m. We assume that (7) applies as soon as the triggering configuration is detected by the syntactic parser.

Scope Ambiguities

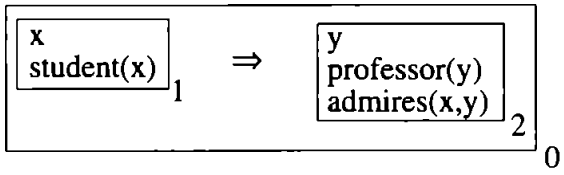
Sentence (8) is an example of scope ambiguity:

(8) Every student admires a professor

The sentence can either mean that every student admires a particular professor (the wide scope reading of the indefinite phrase), or it can mean that the students admire different professors (the narrow scope reading of the indefinite).

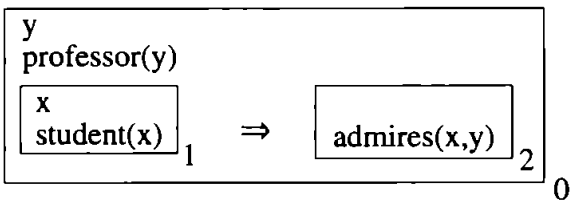
DRT, as presented here and in Kamp & Reyle, assigns a DRS like (9) to this sentence, assuming a top-down left-to-right translation to semantic representations:

(9)



The construction rule for indefinites refers to the "current" DRS, and the current DRS is DRS_2 when the translation takes place. Thus, the wide scope interpretation of the existential phrase is lost. This reading should be represented as

(10)



Williams' Analysis of Scope Ambiguities

Williams (1986, 1988) proposes a scope theory without quantifier raising in Logical Form. This theory is interesting for the design of natural language processing systems because it avoids operations on phrase structure (LF movements are operations on phrase structure). Williams assumes that "a quantification structure consists of four elements: the quantifier, the variable, the scope and the restriction on quantification" (Williams 1988:136). A restriction is for instance **man** in **every man**; the variable is an empty category or a quantifier *in situ*. In examples like (11) the quantifier is *in situ* and occupies the variable position:

(11) John saw everyone

In sentence (12) the quantifier binds an empty category in the variable position:

(12) What$_i$ did John see e$_i$

Since a quantifier like **everyone** in (11) doesn't move in Williams's system, its scope must be defined by other means than c-command, which is the standard way of defining the scope of moved quantifiers. Williams assumes that the node *S* (S=InflP) restricts the scope of a quantifier.

Consider now the scope analysis of sentence (8) without QR:

(8) [$_S$ Every student [$_{VP}$ admires a professor]]

Node S dominates both *every student* and *a professor.*[1] The two quantifiers thus share scope. The scope ambiguity follows straightforwardly if we assume that scope orderings are underdetermined when two or more quantifiers are included in the same scope domain.

Deterministic Processing and Parallel Syntactic and Semantic Structure Building

In this section I will try to show that certain processing techniques and principles developed in Nordgård (1993) are useful in the computation of scope ambiguities in a GB/DRS approach, together with a scope analysis without LF-movements like Williams'. The parser described in Nordgård (1993) is deterministic, sloppy deterministic, to be precise.[2] It cannot destroy or "forget" structure it has created. Information can, however, be added to its left context, e.g. indices and new constituents. Importantly, such a parser does not waste time on non-well-formed structural representations, and, consequently, it is efficient.[3]

In the examples below I will assume some familiarity with Nordgård (1993). To recapitulate very briefly, the system has the following important properties: The analysis starts out with a sentential template, e.g. [$_{CP}$ [$_{XP}$] [$_{C'}$ **X$_j$** [$_{IP}$ [$_{NP}$] [$_{I'}$ **e$_j$** [$_{VP}$ [$_{V'}$ [$_V$ **e$_j$**]]]]]]]. Positions in boldface, i.e. Spec-CP, Spec-IP, Head-IP and Head-VP, will be considered during the parsing process, and positions without empty

[1] Assume, for the moment, that the structure of (8) is [$_S$ [every student] admires [a professor]].

[2] A sloppy deterministic machine can output a set of analyses for some input string as long as each analysis is computed deterministically. A "standard" deterministic device is only allowed to produce excactly one result.

[3] If the search space is huge then efficiency decreases, of course. For discussion, see Nordgård (1993), chapter 7.

categories will be instantiated by lexical material. New positions are added on the basis of properties of lexical items introduced into the tree (subcategorized constituents), or non-subcategorized constituents discovered in the input string (adjuncts). The parser's "attention" in the tree is governed by a stack of queues of waiting positions. Positions are represented as integers referring to unique nodes in the tree (cf. (i) below), and they are organized in queues. These queues are in turn organized in a stack. This organization enables the parser to delay parts of the analysis until substructures are analyzed properly. The details are irrelevant in the examples to be discussed below. Finally, the parser makes use of procedural instructions ("heuristic rules") when deciding what to do in a given state (trace insertion, PP attachment, and so on). See Nordgård (1993) for a comprehensive discussion of this parsing system.

In what follows I would like to explore whether DRSs can be built deterministically, and in particular whether scope ambiguities can be captured by deterministic techniques.[1] The most important ideas are as follows:

- DRSs are created in parallel with the syntactic analysis
- Quantifier indices percolate upwards in the tree
- The scope of a quantifier *in situ* is determined by the node where its percolated index is terminated

An Example

Let me show the effects of these ideas by an outline of the syntactic and semantic analysis of sentence (13):

(13) Jens beundrer enhver professor
Jens admires every professor

Stages in the analysis will be represented as a triple containing the "remaining" string items, the structural representation built "so far", and the DRSs derived from the structural representation "so far":

(14) a. Input string, b. Tree structure, c. DRS(s)

First the clausal template is initialized (the second line of (i), see below). Each node has a unique identifier (a number attached to the left, e.g. $_3$C') which makes it possible to refer to them in DRSs. Assume that the main DRS is empty in this example:

[1]Of course, scope ambiguities must rely on *sloppy* deterministic techniques.

i. *Jens beundrer enhver professor*
[1CP [2**XP**] [3C' 4$\mathbf{X}_j$ [5IP [6**NP**] [7I' 8e_j [9VP [10V' [11**V** 12e_j]]]]]]]
Empty main DRS

The next step is the syntactic analysis of the content of Spec-CP, which turns out to be an NP containing the proper name **Jens**.[1] When the analysis of Spec-CP is completed, information can be put into the DRS. The parser's attention will now be at Head-CP.

ii. *beundrer enhver professor*
[1CP [2NP Jens] [3C' 4$\mathbf{X}_j$ [5IP [6**NP**] [7I' 8e_j [9VP [10V' [11**V** 12e_j]]]]]]]

x, Jens(x), #1#, x:#2#

The notation used in the DRS calls for some comments. Numbers enclosed by "#" refer to nodes in the tree. The expression x:#2# means that variable x is connected to the position in the tree where node 2 is. If the variable prefix is absent, #n# refers to the "current tree". For the moment this can be taken as simply a notational convenience which replaces the entire tree in Kamp & Reyles notation, but later in this section it will be demonstrated that this notation opens for a flexible account of scope ambiguities.

Next the verb is attached to Head-CP; an empty category is inserted in Spec-IP, and the subcategorized argument of **beundrer** is inserted in the tree. The remaining input string is analyzed as the object of **beundrer**:

iii. ∅
[1CP [2NP Jens]$_i$ [3C' 4beundrer$_j$ [5IP e_i [6NP e_i] [7I' 8e_j
[9VP [10V' [11V 12e_j [13NP [Det enhver] [N' [N professor]]]]]]]]]
x, Jens(x), #1#, x:#2#

Construction rule (7) can be applied, and the result is

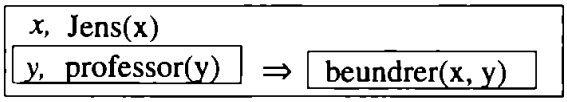
iv. ∅
[1CP [2NP Jens]$_i$ [3C' 4beundrer$_j$ [5IP e_i [6NP e_i] [7I' 8e_j
[9VP [10V' [11V 12e_j [13NP enhver professor]]]]]]]

x, Jens(x),		
y, professor(y)	⇒	#1#, x:#2#, y:#13#

[1]This is not an appropriate occasion for introducing the operations of the parser. The relevant heuristic rules are described in Nordgård (1993).

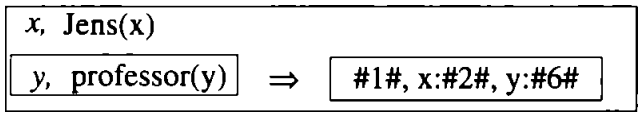
The DRS in the standard shorthand notation:

(15)



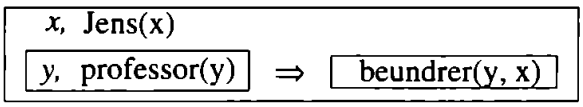
As is well-known, sentences like (13) are ambiguous in a verb second language like Norwegian. In addition to the reading described above it also has an interpretation where **Jens** is the object of **beundrer**. Since the parser assigns both syntactic representations to the string, distinct DRSs will be created:

v. ∅
[1CP [2NP Jens]i [3C' 4beundrerj [5IP [6[NP [Det enhver]
[N' [N professor]]]] [7I' 8ej [9VP [10V' [11V 12ej [13NP ei]]]]]]]]]



In standard notation:

(16)



Hence, the system does output the desired set of DRSs when input sentences are structurally ambiguous.

Scope Ambiguities and Index Percolation

Let us now turn to scope ambiguities:

(17) Enhver student liker en professor
every student likes a professor

As in the previous example, this clause is structurally ambiguous. Space considerations do not permit a discussion of both syntactic readings and their semantic implications. We will consider only the reading where *enhver student* is the subject.

When *enhver student* has been attached to Spec-CP, the main DRS, assumed to be initially empty in this example, is to be modified:

(18) *liker en professor*
[1CP [2NP enhver student] [3C' 4$\mathbf{X}_j$ [5IP [6NP] [7I' 8e_j
[9VP [10V' [11V 12e_j]]]]]]]

x, student(x)	⇒	#1#, x:#2#

Attachment of the verb and projection of the object position are as in the previous example. When the object is syntactically analyzed, the narrow scope reading of the existential phrase is obtained:

(19) ∅
[1CP [2NP enhver student]$_i$ [3C' 4liker$_j$ [5IP [6NP e_i] [7I' 8e_j
[9VP [10V' [11V 12e_j] [13NP [Det en] [N' [N professor]]]]]]]]

x, student(x)	⇒	y, professor(y) #1#, x:#2#, y:#13#

If the DRS in (18) is the only one available, the wide scope reading of the object is lost. To obtain both interpretations *index percolation* comes into play. Recall from Williams' system that scope ambiguity arises when two quantifiers share scope domain. The idea to be elaborated here is that scope ambiguity arises as soon as an index percolates to a position in the tree which dominates (or equals) another quantifier. If the index of the object percolates to CP in (19), the indefinite takes scope over the subject.[1]

Suppose that the index of quantified phrases can percolate up to any maximal projection whose head has semantic content, where 'semantic content' is to be understood as 'ability to assign thematic roles'. Note that the syntactic analysis assumed here implies that the index can percolate to CP in root clauses because the verb moves to Head-CP. Thus, the index of the existential phrase will at least percolate to VP, but since VP does not dominate node #2 no new interpretation can be derived. When index #13 reaches CP the following configuration (20) results:

(20) [**1CP:13** [2NP enhver student]$_i$ [3C' 4liker$_j$ [5IP [6NP e_i] [7I' 8e_j
[9VP [10V' [11V 12e_j] [13NP [Det en] [N' [N professor]]]]]]]]

x, student(x)	⇒	y, professor(y) #1#, x:#2#, y:#13#

[1] A possible alternative is that scope ambiguity arises as soon as some percolating index dominates an EC bound by another quantifier. If so, percolation to IP is sufficient in the example under consideration.

Now the index of the object is connected to a position (node #1) which dominates the position held by the other quantifier, i.e. node #2. If a DRS is created in this configuration another scope interpretation results *provided that the constituent referred to by index #13 is translated first:*

(21)

y, professor(y)
#1#, y:#13#

In this representation the variable *y* has scope over the entire clause, assuming that its structural representation is the same:

(22)

y, professor(y)		
x, student(x)	⇒	#1#, x:#2#, y:#13#

Even though we are talking about distinct DRSs representing different scope orderings, the DRSs share some information. For DRS (20) and DRS (22) the global DRS existing prior to the analysis of the clause is common. By assuming an empty discourse in the examples under discussion this point is perhaps not so obvious, but it must nevertheless be taken into consideration. Assume that each node in the tree has a corresponding DRS. We need not be concerned about how the correspondence is made technically, but the corresponding DRS should be a copy of the "current" DRS prior to the analysis of the daughters of the relevant node. Thus, the corresponding DRS of node #1 in (20) is a vacuous structure because the initial DRS was assumed to be empty. Given these assumptions, consider state (ii) from the processing example above:

ii. *beundrer enhver professor*
[$_{1}$CP [$_{2}$NP Jens] [$_{3}$C' $_{4}$**X**$_{j}$ [$_{5}$IP [$_{6}$**NP**] [$_{7}$I' $_{8}$e$_{j}$ [$_{9}$VP [$_{10}$V' [$_{11}$**V** $_{12}$e$_{j}$]]]]]]]

x, Jens(x), #1#, x:#2#

$_{1}$CP: Empty DRS
$_{2}$NP: Empty DRS
$_{3}$C': | *x*, Jens(x), #1#, x:#2# |

Information is not put into any DRS until a node yielding such information is analyzed. Note, in particular, that the DRS connected to a node which "triggered" some information is not affected by "its own semantic information". That is, the DRS connected to node 2 is not modified by the semantic content extracted from node 2. This information is passed onto node 3. The last DRS created in an analysis is one of possibly more final results.

Given these assumptions, it is fairly straightforward to build alternative DRSs for sentences with scope ambiguities: If the index percolation process shows that a percolated index i of some quantifier Q is attached to some node X which dominates another quantifier P, a new DRS can be made as a corresponding DRS of X. In the new DRS the variables introduced by Q are introduced. To preserve determinism, the index percolation process must take place deterministically. If we adhere to the tree searching strategy developed in Nordgård (1993), the process can informally proceed as follows: Whenever a relevant quantified expression Q is detected, check whether there is a c-commanding quantifier P higher up in the tree.[1] If so, put an index of Q on the maximal projection MP dominating P. Create a new DRS based on the DRS connected to MP. Introduce the variables introduced by Q here. Restart the analysis from this point.[2]

Applied to the example above, the processing starts up with node 1CP again, but now the corresponding DRS contains the information in (21). Provided that the same variable is not introduced again when node 13 is translated, the wide scope analysis of the object phrase is achieved.

Conclusion

This paper has demonstrated that the processing system developed in Nordgård (1993) can be related to DRT in a way which preserves the deterministic nature of the syntactic parser. In particular, scope ambiguities can be handled by deterministic techniques. I believe this is an important result because it shows that scope ambiguities do not enforce guessing algorithms.

[1]This search can be accomplished by deterministic finite state machinery, cf. Nordgård (1993, chapter 7) for discussion.

[2]This strategy presupposes that a copy of the remaining string elements is stored together with the DRSs of the nodes. One might object that it seems unnecessary to perform the analysis once more. This is presumably true.

References

Kamp, Hans and Uwe Reyle. 1992. *From Discourse to Logic*. Ms, University of Stuttgart.

Nordgård, Torbjørn. 1993. *A GB-Related Parser for Norwegian*. Peter Lang Inc., Bern.

Nordgård, Torbjørn. 1992. *On Existential NPs, Multiple Interrogation and Scope*. Unpublished ms., Department of Linguistics and Phonetics, University of Bergen.

Williams, Edwin. 1986. *A Reassignment of the Functions of LF*. In LINGUISTIC INQUIRY 17.2.

Williams, Edwin. 1988. *Is LF Distinct from S-Structure? A Reply to May*. In LINGUISTIC INQUIRY 19.1.

Methods and Tools for Corpus Lexicography

Ole Norling-Christensen
København

Abstract

A survey is given of some technical aspects of the theory and practice of building and using text corpora for dictionary making. The survey builds on newer, especially Anglo-Saxon, literature, as well as on the experience of the editorial team of The Danish Dictionary.

1. Introduction

The work on the mainly corpus based, 6 volumes dictionary of contemporary Danish[1] was initiated in September 1991. Since then, a 40 mil. words (i.e. tokens) corpus of all kinds of general language has been collected, and each of the c. 40.000 text samples has been annotated according to a rather elaborated text typology.

In parallel, methods for reuse of existing lexical sources have been developed, and a database, The Word Bank (Duncker, forthcoming) of morphological, morphosyntactic, semantic and contextual information on more than 300.000 words (i.e. lemmas) has been extracted/constructed from the machine readable versions of some standard printed dictionaries, supplemented with corpus evidence. Work on word class tagging of the corpus is (September 1993) in its initial phase, as is the writing of dictionary entries.

2. Types of corpus - types of tool

As pointed out by the Text Encoding Initiative (TEI26 1993: 3), the term *language corpus* is used to mean a number of rather different things. However, for TEI, as well as for the purpose of this paper, the only distinguishing feature of a corpus that really matters is that its components have been selected or structured according to some conscious set of design criteria. A similar corpus definition is given by Atkins & al. (1992: 1) who, partly building on earlier work by Quémada, distinguish four types of machine readable text collection:

[1] Den Danske Ordbog, hereinafter called DDO.

- *archive*: a repository of readable electronic texts not linked in any coordinated way;
- *electronic text library (ETL)*: a collection of electronic texts in standardized format with certain conventions relating to content etc, but without rigorous selectional constraints;
- *corpus*: a subset of an ETL, built according to explicit design criteria for a specific purpose; and
- *subcorpus*: a subset of a corpus, either a static component of a complex corpus or a dynamic selection from a corpus during on-line analysis.

This distinction mirrors an increasing grade of refining of the textual material; and as different tools are needed for the different levels of refinement, as well as for getting from one level to the next one, the distinction shows useful for classifying corpus tools as well. It is, thus, evident that computational tools are needed not only for corpus analysis, but also for the creation and preprocessing of corpora. And the more information in the form of linguistic as well as extralinguistic annotation according to the design criteria that is added to the text samples of the corpus during the preprocessing, the more sophisticated analyses can be made.[1]

Obviously, the tools used for preprocessing and the tools used for analysis must be compatible, by which is meant, among other things, that the analysis tools must conform to the format of the preprocessed texts and be able to make use of the complementary information of the annotations. Useful guidelines for defining such formats can be found in the SGML standard as it is utilized by the Text Encoding Initiative (TEI P2, 1992-).

2.1. From text archive to text library

The text archive may include all kinds of word processor and typesetter files. Changing them into the standardized format of a text library implies not only homogenizing the character set, but also making up one's mind about which of the features, represented by formatting codes in the files, should be preserved, and which not. It will, for instance, hardly be relevant to keep information on specific typefaces; on the other hand, it might be useful to keep some generic information on e.g. headlines and emphasis (leaving out information on whether the emphasis was represented by italics, underlining, boldface or small caps). For this job, text converting programs are needed. They may be supplied as functions

[1] One should, however, keep in mind that tagging a corpus according to e.g. a grammatical theory is likely to make the corpus less usable for testing other theories. Further, there is the risk of circularism: the evidence you get out of your corpus may be, more or less, only what you yourself did put into it.

of the source word processing program; but if this is not known or not available, rather much text specific programming may be needed.

At this stage, each text should also be annotated with at least the directly available bibliographical information, preferably in the form of an SGML header element. One may also chose even now to include text typological information like genre, subject and sender-reciever relationship, sociological information like sex, age and education of the author(s), linguistic information on e.g. language variants. Whether these annotations are made now or during the subsequent phase of corpus making, they have to be made in a standardized form and, whenever possible, with their values taken from a limited set of options. Only then, they will be useful for computational processing. Some kind of syntax and content checking device is, therefore, needed during the process of annotation.

2.2. From text library to corpus

Making one or more corpora out of a text library implies selecting in a balanced way samples of the library texts, in order to fit the specific purpose of the corpus. If information on text types etc. of the library is available in standardized, electronic form, such selection may be done more or less automatically. If this is not the case, it is now time for making these annotations. For checking the balance according to different criteria (i.e. annotations) and combinations of criteria, a statistical tool is needed for computing the relative sizes of texts belonging to different classes.

In addition to the annotations related to selectional criteria, the scope of which will typically vary from the entire corpus down to an entire text sample, the corpus design criteria may also define kinds of additional information that must be added at lower levels, the scope being individual sentences, phrases, words, or even parts of words. The object of these kinds of annotation will normally be some kind of computational and/or computer-aided analysis; consequently, the annotation system has to be thoroughly formalized.

3. A standard format for corpus samples[1]

The international standard SGML (ISO 8879, 1986) for generic description of textual structures and for marking up the texts accordingly, is used by The Danish Dictionary for describing and tagging not only the

[1] An earlier version of this section was part of Norling-Christensen 1992.

dictionary but also the corpus. Readers who are not familiar with SGML and with terms like DTD, element, attribute, entity reference, may consult the brilliant introduction in (TEI 1990: 9-32).

For the corpus an SGML document type *CorpusEntry* has been defined. It provides a suitable form for registration of the necessary (extralinguistic) information *about* the text as well as a means for unambiguous tagging of those (linguistic) features of the text proper that we have decided to represent in the corpus. Each sample (element *CorpusEntry*) consists of: A *Header*, that contains information on the kind and provenience of the text, and the *Text* proper. In the language of SGML:

```
<!DOCTYPE   CorpusEntry  [
<!ELEMENT   CorpusEntry     ( Header, Text ) > ] >
```

3.1. The Header

For designing the Header and deciding which information types should form part of it, we found much inspiration in Atkins (1992). The Header of each corpus entry is divided into two main parts, *viz.* information on the source (SourceInfo), and information on the text sample proper (TextDescription). *SourceInfo* consists of an unambiguous identification (TextGroup/TextNumber), notes on restrictions of use imposed by the supplier of the text (private or confidential texts), information on those who produced the text (LanguageUser = authors, speakers), and on title, publisher, date of origination, and location (*e.g.* page number). There is one element *LanguageUser* for each person involved in the production of a given text sample. Especially in spoken language there usually will be more than one. The element describes the person's name, role (*e.g.* interviewer or interviewee), sex, education, occupation, year and place of birth, and language variant (*i.e.* standard or regional Danish). The element *TextDescription* gives an account of the language type (general or special), whether it is written or spoken, and public (reception) or private (production), the age relation between sender and receiver of the text (adult-adult, adult-juvenile, adult-child, juvenile-adult, etc.), medium (book, newspaper, television etc.), genre, subject field, size of the sample.

The full structure and contents of the header can be explained in the following way. An interrogation mark (?) after an element name means that the element is facultative, *i.e.* it shall only be there if it is relevant, and if the information in question is known. The plus (+) after LanguageUser means that there may be one or more of these elements in a single header.

Header
  SourceInfo
    TextGroup *Unambiguous identifier of a group of (related) corpus entries*
    TextNumber *Serial number inside the text group*
    Restriction?
      RestrictA *Proper names in text must be altered: "Y[es]"/"N[o]"*
      RestrictB *Text must only be used for the dictionary: "Y[es]"/"N[o]"*
      Expiration *of Restriction B, e.g. "1998"*
    LanguageUser+ *(one element for each author/speaker)*
      Role? *Esp. when more language users are involved; e.g. "teacher", "pupil"*
      Identification? *A unique three character string, referred to by SpeakerTurns in the Text*
      LastName? *If known*
      FirstName? *If known*
      ***Sex** "m"/"f"/"u[nknown]"*
      Education? *if known*
      Occupation? *if known*
      ***YearOfBirth** a number between 1880 and 1990*
        Precise? *"?", if not known exactly*
      PlaceOfBirth? *if known*
      ***LanguageVariant** "standard"/"regional"*
    TextTitle? *if any*
    VolTitle? *Name of Anthology, Newspaper, Magazine, etc., if any*
    Publisher? *Book publisher or Radio or TV station, if any*
    Date
      Day? *if known*
      Month? *if known*
      Year *number between 83 and 92*
        Precise? *"?", if not known exactly*
    Location? *e.g. Section, page, column of Newspaper; (Vol.,) page of book*
  TextDescription
    ***LanguageType** "general"/"special purpose"*
    ***Written_Spoken** "written"/"spoken" or one of two intermediate types*
    ***Aspect** "reception"/"production"*
    ***AgeRelation** "child-child"/"child-juvenile"/"child-adult"/../"adult-adult"/"unknown"*
    ***Medium** taken from a list of 12 different media, e.g. book, journal, radio, film*
    ***Genre?** taken from a list of 124 partly medium-dependent genres, like novel, letter, comic*
    ***Subject?** taken from a list of 64 different subject areas, like biology, literature, physics*
    Size *Number of words (tokens) in this text sample*

The elements marked by an asterisk (*) above are standardized descriptors that play a special role in corpus search and analysis. For each of the descriptors a restricted list of legal values is defined. Different text types, and corresponding subcorpora, can be defined in terms of one or more of these descriptors, *e.g.* "Women born before 1940 speaking to

children" or "Newspaper texts on politics". Besides, the decriptors are used for studies of the distribution of all kinds of linguistic features over the different text types.

3.2. The Text

The structure of the *Text* element depends on whether it consists of written language or of (transcribed) spoken language. Written language is split up into paragraphs (the element *p*) that are subdivided into sentences (the element *s*). Sentences are mostly non-tagged strings of characters[1] (the SGML category #PCDATA); these may, however, be interspersed with elements of special types of text, viz. the elements *Highlighted* and *Note* . The tag *Highlighted* covers all kinds of accentuation in the original text: underlining, boldface, italics, spacing, bigger or deviant fonts; *Note* are foot- or endnotes.

Spoken language is normally not cut into paragraphs; instead, they may be divided up into speaker turns. Most of the spoken texts are conversations or interviews with more persons involved. Consequently, the header contains two or more instances of the element *LanguageUser*. Each of them contains in the subelement *Identification* a different three letter string. Each element *SpeakerTurn* contains an attribute *id* that refers to the *Identification*. The *SpeakerTurn* element consists of #PCDATA interspersed with entity references like {hesitation} representing non-verbal sounds like 'eh', 'mmm'; {pause}; {uf} that represents a passage that was incomprehensible to the transcriber; {laughter}; and with the elements *Comment* (the transcriber's "stage directions" that are not part of the speech), and *Doubtful*: a word or passage that the transcriber was not sure about.

4. Use of the corpus

4.1 Two problems: abundance and scarcity

The lexicographer working with corpora runs into two basic problems: the theoretical problem of the significance of sparse or none instances of some linguistic phenomena, and the practical problem of being flooded with too many instances of others. The former problem can only be solved by making the corpus even larger, or by relying on sources external to the corpus. To cope with the latter, however, computational

[1]The ongoing word class tagging splits the sentences up into single words, each with a word class attribute, leaving only the interpunction untagged.

tools are needed in order to structure the flood; without such tools, large corpora will not be of much use.

4.2. Interactive analysis

"Structuring the flood" can be seen as the repetitive process of asking ever more specific questions to the material. The basic, first question is "Give me all the instances of the lemma X". The following questions include contextual restrictions which can be made the more precise the more annotated the corpus is. The questioning is repeated until some characteristic behaviour of the word (lemma) crystallizes. Once such behaviour (e.g. one meaning; one valency frame) has been recognized and described by the lexicographer, the instances of it are thrown away, and the procedure is repeated for the rest of the instances.

4.3. Statistical analysis

There is, however, one class of important questions that cannot be meaningfully asked just to the immediate context of the instances of a lemma. Exploration of the collocational behaviour of a word is not possible without some knowledge of the corpus as a whole. The mere observation that one word seems to be frequently occuring in the neighbourhood of another word does not in itself indicate a collocational connection between the two, neither does a seemingly infrequent occurrence indicate the absence of such connection. Only a statistical calculation that takes into account the total numbers of occurrence of the words in question can give a reliable indication.

In Church (1991) three statistical methods for collocational studies are discussed. They all ought to be part of toolboxes for corpus analysis, even though rather big corpora are needed in order to make reliable statistics. "Mutual Information" reveals positional interdependence between two words by comparing the observed frequency of a co-occurrence to the calculated frequency for co-occurrence by chance. "Scale Statistics" calculates the mean and the standard deviation of the distance between such pairs. The more sophisticated "T-score" test looks for significant differences between the immediate neighbourhood of two different words, typically pairs of near synonyms like "strong"/"powerful" or "his"/"her". The observed neighbouring words, e.g. in the position immediately to the right of the two, are ranged on a scale spanning from those having greatest affinity to one of the synonyms, through those which are neutral, to those with greatest affinity to the other synonym.

4.4. Subcorpora

In so far as the individual text samples that make up the corpus have been annotated with text typological information etc. (cf. section 3.1.2.1) it shall be possible to use (boolean combinations of) the annotations for the selection of subsets like e.g. "texts on science or medicine written by women born before 1950". The result may be a new corpus of its own; but a flexible corpus management system will also allow for creating temporary virtual subcorpora by inserting filters between the corpus and the user. The full range of analytical methods specified for a corpus must also be appliable to a virtual subcorpus as well as to the collections of corpus examples (e.g. sentences or KWIC lines) that result from searches and subsequent annotating and sorting.

5. Computational tools

As much as possible of the Header-information, as well as the identification and tagging of the entity references and subelements of the Text proper, is made (semi)automatically. This means, that for each group of texts of a given provenience or type, a customized conversion program is written. A toolbox of Borland Pascal units, called DICONV, programmed by the author, makes such programming fast and efficient; it was originally made for conversion and adjustment of dictionary texts for a publishing house. Not only does the program convert a given wordprocessor format into our standard format; in some cases it also makes use of the authors' idiosyncratic ways of marking those features we are interested in. In other cases these features are marked up manually, using word processor macros. The rest of the header information is keyed in using a customized database application.

5.1. Grammatical tagging

The only "syntactical" tagging that is done for the moment is the delimiting of paragraphs and sentences. For this, Jann Scheuer has written a program that analyses surface information like Newline, interpunction, and the use of uppercase letters. The biggest problems in delimiting sentences are the well known ambiguities of the full stop character (abbreviation mark, ordinal number mark, sentence delimiter, or both sentence delimiter and one of the other functions at the same time), and of uppercase initial being either a proper name marker or conventionally put after a full stop, or both at the same time. As a by-product, the program produces lists of identified proper names and abbreviations. It further makes use of such lists during the analysis. The best result are, therefore, obtained by running the program twice.

After the delimiting of sentences, each sentence is run through a word class disambiguating system made by Jørg Asmussen. The system was originally made for German as part of the authors thesis; but he now has accomodated it for the needs of The Danish Dictionary. The analyses is based on the likelihoods of different sequences of word classes; these are established during training sessions, where the program asks the human trainer for advice when in doubt. The system's dictionary of "homograph classes" is derived from the Word Bank.

5.2. Corpus analysis

For corpus search and interactive analysis, a tool called Corpus°Bench has been developed by the Danish software house TEXTware A/S according to specifications made jointly by Longman Publishers (UK) and The Danish Dictionary. Concordances can be built in real time according to complex search criteria in the form of Boolean combinations of a keyword (lemma) with neighbouring words and/or text type specifications. The concordance lines can be tagged by the user according to up to eight different, user defined criteria. The lines can be sorted according to any combination of key word, left context, right context, user defined tags, and text type information. Besides the statistically based methods, mentioned above, for collocational analysis are available. Further, frequency information, including frequency distribution over e.g. text types, can be obtained.

References

Atkins, Sue, Jeremy Clear and Nicholas Ostler. 1992. *Corpus Design Criteria.* pp.1–16, LITERARY AND LINGUISTIC COMPUTING, Volume 7, Number 1, Oxford University Press.

Church, Kenneth, William Gale, Patrick Hanks and Donald Hindle. 1991. *Using Statistics in Lexical Analysis.* In Zernik (ed.).

Zernik, Uri (ed.). 1991. *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon.* Erlbaum, New Jersey.

Duncker, Dorthe and O. Norling Christensen. Forthcoming. *Genbrug af ordbogsdata. Den Danske Ordbank.* To appear in the *Proceedings of the 2nd Scandinavian conference on Lexicography, Copenhagen 1993.*

Kiefer, Ferenc et al. (ed). 1992. *Papers in Computational Lexicography. COMPLEX '92.* Linguistics Institute, Hungarian Academy of Sciences, Budapest.

Norling-Christensen, Ole. 1992. *Preparing a Text Corpus - Computational Tools and Methods for Standardizing, Tagging and Structuring Text Data.* In Kiefer (ed.).

TEI P1. 1990: C.M. Sperberg-McQueen and Lou Burnard (ed.s): *ACH - ACL - ALLC. Guidelines for the Encoding and Interchange of Machine-Readable Texts. TEI P1. Draft Version 1.1.* Chicago and Oxford, 1 November 1990.

TEI P2. 1992-: C.M. Sperberg-McQueen and Lou Burnard (ed.s): *ACH - ACL - ALLC. Guidelines for Electronic Text Encoding and Interchange. TEI P2. Part I–VIII.* Chicago, Oxford. In preparation; partly released 1992–93.

TEI26. 1993: *Additional Tag Set for Language Corpora.* In *TEI P2 1992-, part IV,* chapter 26, released 11 March 1993.

Natural language processing in dialogue systems with spoken input

Claus Povlsen
København

Abstract

This paper describes the linguistic analysis done within a research project, the goal of which is the development a spoken language understanding system. The first part outlines the overall system design including a description of the constraints to be dealt with in systems handling spoken input. The second part contains a description of how the domain-specific sublanguage was defined and how domain-knowledge was exploited in the syntactic and semantic analyses in order to make the linguistic description as precise and unambiguous as possible.

1 Introduction

The project *Spoken Language Dialogue Systems*[1] aims to produce two functioning prototypes of spoken language understanding systems which will enable a given user to make flight reservations automatically via the telephone. The task for the Centre for Language Technology in the project is to define and implement the natural language user interface between the speech recognition and the knowledge based components of the system.

Machine-based speech understanding systems differ profoundly from more "traditional" NLP systems which usually take input in a textual format. Processing speech to a symbolic representation involves a huge number of time consuming computations and has thereby set narrow bounds for the domain-specific sublanguage, i.e the user's linguistic interface to the developed system. Its interactive nature is another characteristic feature of the system which has influenced the choice of which type of sentence constructions were included in the grammatical coverage of the sublanguage. Finally as the project is application-oriented (near) real time performance is required, which has been decisive for the definition and delimitation of the lexical and grammatical coverage of the system.

[1]The basic research project is sponsored by the Danish Technical Research Council and is a cooperative effort between the Centre for Language Technology in Copenhagen, the Centre for Cognitive Informatics in Roskilde and the Centre for Speech Technology in Åborg (coordinating partner).

2 The overall system design

The overall architecture of the initial version of the system is modular, with communication between modules being handled by a Dialogue and Control manager, specially designed for graphically defining and executing task-oriented dialogues see Larsen et al. (1993) and Bækgaard et al. (1992).

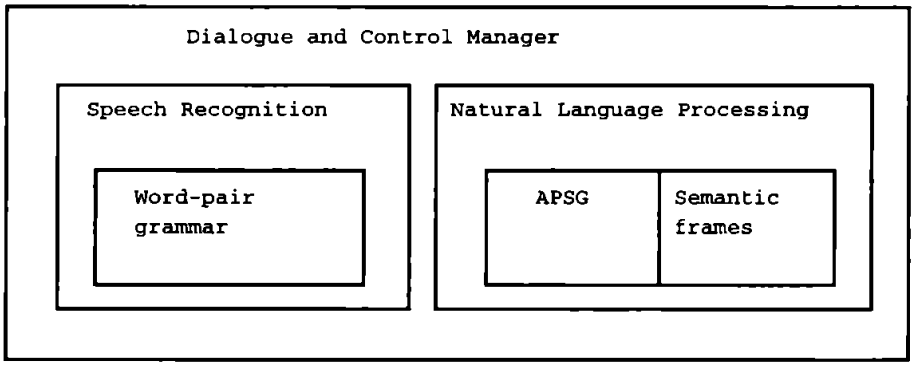


FIG 1: System architecture

The Dialogue and Control Manager receives sentence hypotheses from the speech recognizer, which in turn are sent to the parsing module for analysis. Each sentence hypothesis consists of a list of lexical references that are used by the parsing module for looking up the items in a lexicon. As a result of the NLP, the sentence-semantic information – formally expressed in framelike structures – is returned to the Dialogue and Control Manager for making decisions as to what actions should follow.

In the following, focus will first be on how the word-pair grammar used in speech recognition and the syntactic analysis of the NLP component interact. Thereafter the correlation between the syntactic analysis and the semantic interpretation will be described.

Without access to linguistic knowledge constraining possible combinations of the word forms in the vocabulary, the speech recogniser will in principle regard the number of legal word sequences as being equal to the number of word forms raised to the second power. In practically oriented spoken understanding systems in which real time performance is a decisive design criterion, quality speech recognition is only obtainable if the number of word models under consideration at any point in the acoustic processing is reduced. In order to reduce these reference patterns to be matched, a word-pair grammar, expressing knowledge of legal word-pairs in the sublanguage is defined. The word-pair grammar is automatically generated from the word sequences expressed in the unification based APSG (Augmented Phrase Structure Grammar). This ensures that the grammatical coverage of the APSG is a subset of the coverage of the word-pair grammars. This close relation between the

mentioned grammars has determined a precise and unambiguous formulation of the syntactic rules in the APSG.

In general the syntactic and semantic analyses can be completely separated or integrated in various ways. The graphic representation of the NLP component in the above figure is meant to illustrate that the syntactic and semantic descriptions are distinct, although during processing their applications are interwoven. This will be discussed further below.

3 Definition and delimitation of the domain-specific sublanguage

Experience has shown difficulties implementing large systems with broad lexical and grammatical coverage. In light of this, attention has recently been directed to investigating the possibility of defining subsets of general languages in a principled fashion. Based on analyses of communication within delimited areas, research efforts have concluded that such domains of a general language are characterized by limited vocabularies and use of linguistic phenomena. This corresponds well with the realization that current speech recognition technology must set narrow limits on the linguistic coverage in spoken understanding systems.

In order to define the sublanguage within the application chosen, two different types of data have been collected. A travel agency (DanTransport) gave permission to tape on-site one hour of dialogues between their agents and clients. Furthermore several series of simulated man-machine dialogue experiments (Wizard of Oz-experiments) have been conducted. For a more detailed description see Dybkjær and Dybkjær (1993).

The relative importance of the different kinds of collected data depends on whether the dialogue structure in spoken dialogue systems are strictly system-directed or user-directed. If the dialogue structure is rather unconstrained and therefore comparable to dialogues between humans, recorded human-human data would be a significant source for delimiting the vocabulary. On the other hand, in directed dialogue mode, which constitutes a communication situation different from human-human dialogues, data from simulated human-machine experiments should form the basis for the definition of the domain-specific vocabulary. As in fact the system under development is highly system-directed, the data from the WoZ-experiments have constituted the primary source for defining the domain-specific sublanguage.

The sum of the clients' utterances from the simulated human-machine dialogues have been defined as constituting the domain-specific

sublanguage within the chosen application. Under due consideration to the limited speech recognition capacity, the defined sublanguage has been modified and extended based on acquired knowledge about the domain and about language in general. After adjustment of the vocabulary size, the number of word forms totalled 500.

After having investigated the collected corpus in order to register the linguistic phenomena, the most frequent used syntactic constructions were identified and included in the grammatical coverage. The most dominant phenomenon in the corpus was the presence of a large number of elliptic constructions. As the mode of the dialogue in the simulated system was system-directed, the wizard's (system's) authoritative way of asking questions made it unnecessary and irrelevant for the client to repeat the information already expressed, in which case he just added new information - linguistically expressed as a partial sentence, an ellipsis. A typical example is:

The wizard: *Po hvilket tidspunkt afgor det ønskede fly?*
At what time does the desired plane depart?

The Client: Toogtyve femogfyrre
10:45 PM

In general the implementation of elliptic constructions is implemented by allowing one constituent to be the single element in a sentence rule. In addition domain-specific constituents and word classes have been defined. The rules covering the client's utterance in the above example would thus be as follows:

S :- Hour_p
Hour_p :- Card_p

'Hour_p' is a domain-specific non-terminal covering, among other things, cardinal phrases ('Card_p') expressing *time* and constitutes a single element in a sentence rule. For a more detailed description of the syntactic analysis see Povlsen and Music (1993).

4 Exploitation of domain-specific knowledge in the syntactic analysis

As mentioned above the syntactic grammar in the system serves two purposes. Besides being used for making a structural analysis as a first step towards a semantic interpretation of the utterance, it also forms the basis for the automatically generated word-pair grammars applied for reducing the number of word models during the acoustic signal

processing. The latter function requires strict formulation of the grammar for syntactic analysis in order to make the linguistic knowledge used in speech recognition as precise and constrained as possible.

Besides a reduction in quantity (the number of words and syntactic constructions used), semantic restrictions in sublanguages make it possible to analyse the utterances in a much more specific and precise way. For example on the basis of an analysis of the main verbs with focus on their contextual word patterns domain-specific selectional restrictions and categories can be determined and exploited in the linguistic processing.

4.1 Selectional restrictions

Selectional restrictions express constraints on combinations of lexical units in a given context. By focussing on the argument structure for lexical units, interrelated semantic concepts can be identified. How domain-specific knowledge is expressed in selectional restrictions can be illustrated by looking at following sample utterances from the collected corpus.

jeg vil bestille en billet
I want to order a ticket
jeg vil lave en reservation
I want to make a reservation

If only syntactic restrictions are defined in a sentence rule covering these sample utterance i.e. SUBJECT AUX VERB OBJECT, it would permit acceptance of a large number of meaningless combinations. Consider for instance the following example:

* *en billet vil bestille et barn*
A ticket wants to order a child

If the activation of the syntactic rules is not constrained further, the coverage of the generated word-pair grammars will end up being too broad (loose) and will thereby exceed the limits for the permitted number of active word models, leading to poor speech recognition results. As a first step towards solving this problem, all the domain-specific main verbs and their contextual patterns were examined in concordances. Based on this analysis the domain-specific concepts which link lexical units together were identified. The outcome of the investigation concerning *bestille* and *lave* was that they had the following concept valency frames:

arg1[+HUMAN], arg2[+TICKET [+CONCRETE]]

arg1[+HUMAN], arg2[+TICKET [-CONCRETE]]

The knowledge of domain-specific selectional restrictions is expressed in the lexicon and grammar of the NLP module. The implemented parsing algorithm for the syntactic analysis is an Earley-based, left-right chart parser. This means that in the initial phase of the processing all lexical information is assigned to the input words in the chart and then used by the syntactic grammar as well-formedness constraints in further processing. As mentioned above, the constraint contribute to a lowering of the perplexity in the automatically generated word-pair grammars, thereby improving speech recognition quality.

4.2 Domain-specific categories

In practically oriented NLP systems covering subsets of general languages, definition of word classes and structures specific for delimited subject areas is a method often applied for making the overall process more efficient.

For instance in the domain of flight reservation, it will be more straightforward to define domain-specific categories for phrases such as *klokken toogtyve femogfyrre* (at twenty-two forty-five), *halv ni* (half past eight), which do not fit the ordinary patterns of nominal phrases. By defining non-terminals for these constructions the syntactic component of the system will be activating fewer syntactic rules and thereby reducing the number of wrong word sequences in the coverage of the automatically generated word-pair grammars.

Based on the utterances from the collected corpora, interchangeable elements are assigned with common values in defined attribute value pairs. The cardinals ranging from one to ten is thus coded with: 'minut_fg1=yes' in the lexicon, while all other cardinals are coded with 'minut_fg1=no'. Using this restriction in the grammar rules covering the following phrases:

klokken fem minutter i halv tolv
o'clock five minutes to half two
at one twenty-five

klokken otte minutter over halv fire
o'clock eight minutes past half two
at three thirty-eight

prevents the acceptance of "wrong" word sequences such as:

* *klokken elleve minutter i halv tolv*
o'clock eleven minutes to half twelve
at eleven nineteen

* *klokken seksten minutter over halv fem*
o'clock sixteen minutes past half five
at four forty-six

Dependency among word sequences in the corpora concerning *Hour* is expressed by defining non-terminals. In the sample utterances the word forms within each sequence *fem minutter i halv* and *otte minutter over halv* are dependent on each other in that all the words must be present in order to express a meaningful utterance. In the syntactic grammar these sequences are expressed as follows:

```
half_p =
        [
                {cat=card_p, minut_fg1=yes},
                {cat=n, dalu=minut},
                {cat=p},
                {cat=adj, lex=halv}
        ].
```

By writing a syntactic grammar that is domain-specific, the overall NLP will be more efficient and makes possible an effective matching strategy for elliptic constructions

5 Exploitation of domain knowledge in the semantic interpretation

As mentioned above, there are several linguistic advantages to handling sublanguage instead of general language. Besides a reduction in quantity, semantic restrictions in special languages make it possible to describe sublanguages in a more specific and precise way. Besides reduced polysemy and the possibility of exploiting domain-specific selectional restrictions and defining domain-specific categories, adequate interpretation of an utterance is also simpler, the number of semantic roles in the domain being easier to grasp.

Based on knowledge of known goals within the domain and on a linguistic analysis of the collected corpus, domain-relevant semantic roles have been defined. Based on the above mentioned identification of domain-specific constituents and parts of speech, word classes deemed as essential

for fulfilment of domain-specific goals of the system were then expressed in the semantic representation either as 'frames' or as subordinated 'slots'. Head concepts such as *Reserve* (filled by bestille/reservere) is thus assigned a 'frame', while 'hour' is represented as a 'slot' in this 'frame'.

key: Reserve
slots:
persons
id_number
date{}
hour()
from
to

As mentioned in the description of the overall system design, the syntactic and semantic rules of the NLP-module are separate. However, before processing the two types of descriptions are compiled into the same format. Whenever a new syntactic constituent is found in the input, the parser immediately checks whether any of the semantic rules can be applied. This means that the system searches for semantic interpretations as early as possible in the parse process without waiting for the syntactic analysis to finish.

Error recovery is done on ungrammatical speech recognition results. If the input-utterance *jeg vil bestille en billet til Ålborg* (I would like to order a ticket to Ålborg) is recognised as *jeg vil bestille en Billund til Ålborg* (I would like to order a Billund to Ålborg) the flexible parsing design of the NLP-module makes it possible to interpret the utterance adequately. Despite the fact that the input to the NLP-component is out-of-coverage (no sentence rule exists for this word sequence) access to the semantic information assigned the subconstituents during the processing will be sufficient for making an adequate semantic representation. This is done by implementing a mechanism which collects the results stored in the chart and gather them together under a "dummy" sentence symbol.

6 Conclusion

Satisfactory speech recognition results presuppose access to linguistic knowledge during the acoustic processing. A much debated subject concerning speech understanding systems has been how to integrate speech recognition and NLP in order to simultaneously achieve optimum recognition quality and to generate an adequate syntactic analysis. For a discussion see Brøndsted (1993) and Povlsen and Music (1992). Focus here has been on describing how knowledge of the domain-specific sublanguage has been exploited in the natural language processing in the

initial prototype of the system. This is done partly by application of selectional restrictions and domain-specific categories in order to reduce the generality of the structural analysis and partly by delimitation of the space of interpretation in the semantic analysis.

References

Brøndsted, Tom. 1993. *Integration af akustisk genkendelse og natursprogsprocessering.* Datalingvistik foreningen Årsmøde 1993.

Bækgaard, Anders, A. Roman and P. Wetzel. 1992. *Advanced Dialogue Design -- DDL Tool and ICM.* ESPRIT SUNSTAR 2094, Deliverable IV. 6–2.

Dybkjær, Laila and Hans Dybkjær. 1993. *Wizard of Oz Experiments in the Development of the Dialogue Model for P1. Spoken Language Dialogue Systems* – project report 3.

Larsen, Lars Bo., T. Brøndsted, H. Dybkjær, L. Dybkjær and B. Music. 1993. *Overall Specification and Architecture of P1. Spoken Language Dialogue Systems* – project report 2.

Povlsen, Claus and Bradley Music. 1992. *Natursprogsprocessering i dialogsystemer med talt input.* SKRIFTEN PÅ SKÅRMEN, no. 6. HHÅ.

Povlsen, Claus and Bradley Music. 1993. *Sublanguage definition and specification for P1. Spoken Language Dialogue Systems* – project report 4 (in progress).

Automatisk igenkänning av nominalfraser i löpande text

Björn Rauch
Stockholm

Abstrakt

I uppsatsen redogörs för en samling algoritmer för automatisk nominalfrasmarkering i löpande text. Algoritmerna bygger på varandra och använder den redan utförda analysen. De är därmed enkla och inte tidskrävande. Algoritmerna kan grovt indelas i två grupper: den första gruppen markerar **kärnnominalfraser** eller **minimala nominalfraser**. För svenskan rör det sig i stort sett om bestämningar, som står till vänster om huvudledet (substantivet). Den andra gruppen av algoritmer markerar **maximala nominalfraser**. Här lägger man alltså till prepositionsfraser, infinitivkonstruktioner, m m. Den sista gruppen har inte tagits upp här. Indatan har hämtats ur tidningar, böcker och andra publikationer på svenska. Texterna taggades med ordklasser och morfologiskt markerade, grammatiska kategorier, men för övrigt använder algoritmerna ingen lexikal information. (Även om mindre ordlistor kunde förbättra resultatet avsevärt; t ex en lista över substantiv som bestämmer mängden av någonting och som förekommer i appositioner (i exemplen: *par* och *antal*): *ett par minuter*, *ett antal människor*. Utan semantisk information kan man inte avgöra om det rör sig om en eller två NP.) Indatan innehåller ungefär 12 000 kärnnominalfraser. En vidareutveckling av programmet kan vara att jämföra meningar med liknande struktur (samma finita verb) för att skapa ett valenslexikon (huvudsakligen för verb, men substantiv och adjektiv skulle också kunna vara med).

Allmänna principer

Som man ser i inledningen är uppgiften väldigt komplex. Framför allt om man tänker på att det är en korkad dator som skall utföra arbetet. Därför är det nödvändigt att splittra problemet i små delproblem som lättare kan lösas. Samtidigt kan de olika delarna av programmet ta hänsyn till redan utfört arbete, vilket ytterligare underlättar analysen.

En annan fördel med denna indelning är, att man kan följa principen att inskränka den grammatiska informationen i programmets olika delar, för att se vilka grammatiska kategorier som är nödvändiga respektive onödiga för analysen. Vidare skulle programmet vara tolerant mot 'ogrammatiska' nominalfraser, som:

1 *det stort huset*
2 *den hela frågan*

Tyvärr medför detta också en del problem:

3 *... hörde den blinde statsrådet*

Exempel 3 markerar algoritmen förmodligen som ett verb plus en NP. Men på det viset blir det troligen lättare att upptäcka felaktigt markerade NP (som i 3). Däremot skulle det vara besvärligt att fastställa med en strikt algoritm att 1 och 2 är NP:er.

Det är följaktligen omöjligt att använda enbart frasstrukturregler eller rewrite-rules, utan att reglerna blir mjuka. Medan frasstrukturregler får problem med "nästan"-nominalfraser, säger den andra typen av regler: "kanske är det en nominalfras".

Programmet delades upp i två delar, nämligen regelbasen, som innehåller de lingvistiska reglerna, och algoritmen, som utvärderar indatan med hjälp av regelbasen. Detta åtskiljande förenklar granskningen och förbättrar därmed de lingvistiska reglerna.

Definition av minimal nominalfras

Målet att markera nominalfraser på enbart morfologiska grunder ter sig faktiskt som ett olösbart problem. Men om man begränsar sig till den delmängd som jag kallar **kärnnominalfraser** (nuclear nominal phrase NNP), blir uppgiften vettigare. Begreppet kärnnominalfras kan definieras med utgångspunkt i begreppet nominalfras genom inskränkningar av densammma.

I kategorin nominalfras ingår många komplexa konstruktioner, som gör meningar tvetydiga. Visserligen kunde man klara av ett antal av dessa konstruktioner på grund av ordföljden (alltså syntaxen), men i botten ligger väl huvudsakligen semantiska (och pragmatiska) kriterier, som upplöser dessa tvetydigheter. Det är i första hand två konstruktioner som skall uteslutas med detta argument: prepostionsfraser som (efterställd) bestämmning till nominalfrasen och samordnade nominalfraser. Här följer exempel på dessa (hakparenteser används för att markera nominalfrasgränser; ibland indiceras dem för att förtydliga vilka som matchar varandra):

4a *[Fredrik] tog [bussen] till [Odenplan].*
4b *[Fredrik] tog [[bussen] till [Odenplan]].*
5a *[1 [2 [3 Hans bedömning 3] av [3 krisen 3] 2] och [2 moderaternas förslag 2] 1] skulle komma fram i [artikeln].*
5b *[1 [2 Hans bedömning 2] av [2 [3 krisen 3] och [3 moderaternas förslag 3] 2] 1] skulle komma fram i [artikeln].*

A- och b-versionerna är båda möjliga analyser av strukturen i nominalfraser och båda meningar ändrar betydelsen. Det finns å andra sidan meningar, där kontexten utesluter den ena eller andra tolkningen:

6 *[Sverige] slog [Finland] i [ishockey].*
7 *[Per] tittade på [[figuren] av [trä]].*

För konjunktioner är situationen annorlunda. Här är det svårt att avgöra **vad** som samordnas. Ordföljdskontexten kan vara sådan att konjunktionen står mellan två nominalfraser (som i exempel 5 ovan) men det samordnas inte nominalfraserna, utan exempelvis två satser.

Det finns en tredje konstruktion som utelämnas, nämligen relativa bisatser. Huvudanledningen här är att bisatser 'döljer' en mängd nominalfraser, som därmed går förlorade för en test av analysen. Å andra sidan kan det vara svårt att avgöra var den relativa bisatsen slutar. Om man tittar på hur relativa bisatser används, ser man visserligen att de antingen slutar där meningen tar slut eller vid nästa finita verb (som inte är bisatsens finita verb). Andra möjligheter undviks nästan alltid, fast de förekommer. Anledningen till denna preferens borde vara att det även för en människa är svårt att förstå dessa meningar.

Sammanfattningsvis är det alltså just de bestämningar som i sin tur innehåller nominalfraser som utesluts.

Första algoritmen: Kontextoberoende analys av orden

Här gäller det att skapa utgångspunkten för den följande analysen. Det är därför av nytta att "övergenerera", dvs markera hellre för många kärnnominalfraser än för få. Om alla möjliga kärnnominal-fraser markeras, kan man i nästa steg koncentrera sig på att avgöra om "äkta" nominalfraser genererades. Det verkar däremot vara svårare och mer tidskrävande att hitta nominalfraser i en ordsträng.

Analysen använder ingen syntaktisk information utan genomför en rent morfologisk analys, dvs den granskar varje ord för sig, utan att ta hänsyn till kontexten. Det tycks vara ganska hopplöst med tanke på den komplexa uppgiften att excerpera nominalfraser. Men idén är att man kan säga för olika ordklasser var de kan hamna i en kärnnominalfras eller om de överhuvudtaget kan förekomma i kärnnominalfraser. Prepositioner och verb t ex förekommer aldrig i kärnnominalfrasen. Likaså kan man påstå att där det finns en determinerare (artikel, demonstrativa pronomen) finns även en nominalfras. Vidare kan man säga att nominalfrasen möjligen börjar på vänstra sidan (alltså att det möjligen finns en NP-gräns till vänster om determineraren). Substantiv står som huvudled i en nominalfras och i en kärnnominalfras längst till höger. Reglerna anger precis dessa förhållandena. De anger möjliga NNP-gränser runt orden i meningen. Här följer ett exempel (fler belysande exempel ansluter i nästa avsnitt):

8 *Flickan kysste den snälla pojken.*

Nu granskas varje ord och i enlighet med reglerna sätts parenteser (möjliga NP-gränser) ut (anm: Understrykning visar vilket ord, som genererar parenteserna.):

8' [*Flickan*]] *kysste* [[*den* [snälla [*pojken*]] .

Det ser lite tokigt ut för det finns för många och onödiga paranteser. Men varje ord producerar ju parenteser och "bryr sig inte om" vad de andra orden gör. Nu ansluter steget som sammanfattar de möjliga NP-gränserna till "riktiga" NNP-gränser. Här finns det tre regler:

(1) Följer två vänsterparenteser ('[') på varandra och finns det endast ord emellan (inga högerparenteser!), stryk parentesen till höger.
(2) Följer två högerparenteser (']') på varandra och finns det endast ord emellan (inga vänsterparenteser!), stryk parentesen till höger.
(3) Använd reglerna (1) och (2) successivt, tills det inte längre går.

Resultatet av 8' blir efter upprepade användning av (1) – (3):

8'' [*Flickan*] *kysste* [*den snälla pojken*].

Denna mening markerades alltså alldeles rätt. Den är naturligtvis enkel och motsvarar inte alls vanlig text, som tidningsartiklar, skönlitteratur o dyl. I det följande avsnittet skildras vilka resultat man kan uppnå och vilka problem verkliga texter åstadkommer för algoritmen.

Det fanns vissa problem som programmet inte klarade av. Nu följer en klassificering av de vanligaste misstagen:

- **appositioner**

Programmet antar att varje substantiv är ett kärnled och eftersom den enda information som används i princip är ordklassen, blir det svårt att avgöra om något är apposition eller inte. En möjlig lösning är att sammanfatta två substantiv som direkt följer på varandra. Men för det första klaras inte 10 och dessutom uppstår nya svårigheter med dubbelt transitiva verb (13) (Appositioner diskuteras i avsnitt 6 mer ingående). Exempel:

9 *[ett par] [minuter]*
10 *[ett stort antal] [gulliga katter]*
11 *[Sovjetledaren] [Michail] [Gorbatjov]*
12 *[mannen] [Kalle]*
13 *[Fredrik] gav [Kalle] [boken].*

- **partikelverb, satsadverbial, gradadverb**

Här måste man nämna en känslig punkt i utgångsmaterialet. Ordklassen adverb är för odifferentierad. I kategorin ingår ord som är mycket litet kopplade med varandra såväl syntagmatiskt som paradigmatiskt. Bl a hör till klassen partiklar (*på, uppe*) och gradadverb (*mycket, ganska, lite*) och de kan inte urskiljas från adverb som *också, givetvis, inte* osv. Detta leder till lustiga fel:

14 *[Ändå] tog [man] [det] [lugnt].*
15 *Rör [ner kryddor] och [salt].*
16 *Men [man] rör [ju bara ihop] [en deg], ...*
17 *[Ofta] räcker [det] att strö [ut ungefär] [2 msk mjöl] i [ett tunt lager] ...*

Delvis lyckades programmet att avskilja satsadverbial och partiklar (16), vilket är positivt eftersom har man först isolerat problemet ... Men på köpet får man även att gradadverb, som står först i en nominalfras (17), borttagits. Detta händer emellertid inte ofta. Däremot har satsadverbialen och partiklar mer än femtio procent andel i felen som programmet gjorde.

- **efterställda possessiv, attribut osv**

Ibland förekommer det att ett possessivpronomen eller en adjektivfras som vanligen föregår kärnledet i nomnalfrasen följer efter (i exemplen markeras endast det intressanta fallet):

18 *Säg mig , [flicka lilla] , har du en bra man?*
19 *[Pappa min] , när går vi till Grönan?*

- **nominalfraser med komplexa adjektivfraser**

Med detta menas en konstruktion som huvudsakligen förekommer i kanslispråket och är väldigt markerad i svenskan. Det handlar om en adjektivfras vars bestämmning är en prepositionsfras:

20 *Man föreslog därför [ett för hästsporten gemensamt riksspel].*
21 *[en, i jämförelse med en mer homogen hyresmarknad, högre hyresnivå]*

Dessa nominalfraser markeras på följande sätt:

20 *Man föreslog därför [ett] för [hästsporten] [gemensamt riksspel] .*
21 *[en,] i [jämförelse] med [en mer homogen hyresmarknad,] [högre hyresnivå]*

Andra Algoritmen: Parsning av nominalfraser

Själva kärnan består av en parser för nominalfraser och en "anti-parser" som bedömer om en ordsträng inte är en nominalfras. Parsern undersöker endast ordföljden och inte om orden är rätt böjda (t ex: "... *och detta sista Viggenplanet som* ..."). Till anti-parsern används just den kunskap om strukturen av sådana icke-nominalfraser som diskuteras i det föregående avsnittet. Proceduren för en nominalfras går till så här:

Först körs np-parsern. Den finner nominalfrasen korrekt eller säger att den inte kan avgöra det (tolkar det på det viset):

		parserns svar
22	*den snälla pojken*	är en nominalfras
23	*ut ungefär*	vet ej
24	*ner kryddor*	vet ej

22 godkänns som nominalfras och algoritmen slutar, medan exempel 23 och 24 måste vidare analyseras. Det är anti-parsern som nu får avslöja icke-nominalfraser:

		anti-parserns svar
23	*ut ungefär*	är ej nominalfras
24	*ner kryddor*	vet ej

Det blir över sådana nominalfraser som varken parsern eller anti-parsern definitvt kunde peka ut (23). Nu börjar programmet "anpassa" kärnnominalfrasen genom att ta bort ett ord i taget från den vänstra sidan och kollar igen om den nya strängen är en nominalfras, alltså:

24 *ner kryddor* → **24'** *kryddor*

Nu börjar parsningen om igen och då kommer parsern att godta 24' som nominalfras. Resultatet för 22 - 24 kan man alltså sammanfatta så här:

22 *[den snälla pojken]*
23 *ut ungefär*
24 *ner [kryddor]*

Tredje Algoritmen: Appositioner

Detta tredje steg bör betraktas som ett **försök** att hitta appositioner. Som antytts i avsnitt 4 kan man inte urskilja appositioner på enbart morfologiska grunder. Det är t o m så att meningar är tvetydiga och först på semantisk/pragmatisk nivå upplöses ambiguiteten:

25 *Ett är statsministerns medvetna ljugande inför [konstitutionsutskottet] [1985].*
26 *Ett är statsministerns medvetna ljugande inför [konstitutionsutskottet 1985].*

Det tyder på att en algoritm med dessa hårda restriktioner kommer att göra många fel och frågan är om man överhuvudtaget skall ha med ett sådant steg i analysen eller om man inte skulle inskränka begreppet kärnnominalfras mera.

Det visar sig att appositioner har en egenskap utöver att delarna (nominalfraserna) följer på varandra. Den andra nominalfrasen saknar nämligen determinerare och är indefinit eller nominalfrasen är ett egennamn. Det intressanta är att indefinita nominalfraser utan determinerare inte används så ofta och framför allt inte i kontexten direkt efter en annan nominalfras.

Resultatet av denna parser är blandat. Siffrorna (se avsnitt 7 nedan) visar att fler appositioner hittades än nominalfraser som inte är appositioner. Här följer exempel på felmarkeringar (endast ordsträngar som programmet markerade som apposition visas):

27 *Ikväll sluter en majoritet i [Sveriges riksdag Uganda] till sin bröst.*
28 *Men nu sprids i stället [de forna socialdemokratiska sympatisörerna vind] för våg.*

Detta vittnar om att appositioner inte taggas tillfredställande. Därför kommer en statistik med, och en utan, det tredje steget att presenteras i nästa avsnitt.

7. Resultat

K/F (≈ recall) och K/G (≈ precision) anger procentuellt andelen korrekta jämfört med facit respektive andelen korrekta jämfört med de genererade. Om alla siffror är 100% är resultatet perfekt, om det genereras korrekta men för få NP kommer högerledet att vara 100%. Ett perfekt vänsterled betyder att alla NP har genereras plus lite till

(maximal metod). Normalt bör endast antalet NP:er studeras, men antalet höger- och vänstergränser ("[", "]") kan vara intressanta vid utvecklingen av olika metoder.

Här följer siffrorna för kärnnominalfraser. I statistiken ingår 19 texter på vardera drygt 2000 ord av olika slag såsom lagtexter, romaner, tidningsartiklar och andra. (Exemplen i uppsatsen har – bortsett från exemplen i algoritm-beskrivningarna – hämtats ur dessa texter.) Sammanlagt innehöll materialet 42024 ord i 2517 meningar.

TABELL 1: Resultat efter andra parsern.

	Facit (F)	**Genererade (G)**	**därav Korrekta (K)**	**K/F %**	**K/G %**
NNP	12450	13011	11566	92.9	88.9
"["	12450	13011	12129	97.4	93.2
"]"	12450	13011	12261	98.5	94.2
Ord i NP	20048	19775	19653	98.0	99.4

TABELL 2: Resultat efter tredje parsern (appositioner).

	Facit (F)	**Genererade (G)**	**därav Korrekta (K)**	**K/F %**	**K/G %**
NNP	12450	12408	11719	94.1	94.4
"["	12450	12408	12019	96.5	96.9
"]"	12450	12408	12145	97.6	97.9
Ord i NP	20048	19775	19653	98.0	99.4

8. Slutsats

Syftet med detta taggningsprogram är att demonstrera vad som kan åstadkommas med enkla metoder och med strikta principer. Trots detta är andelen korrekta nominalfraser hög. Det finns en svag punkt som måste nämnas. Kritiken riktar sig mot materialet, nämligen att indatan innehöll den rätta "tolkningen" av orden. Exempel:

genom:	preposition, adverb, substantiv
maskar:	verb, substantiv
väg:	verb, substantiv
bara:	adverb, subjunktion, adjektiv
den, det, ...	determinerare, pronomen

osv

För att avgöra ordklassen måste man analysera hela satsen. Men även här finns möjligheten att köra en liknande enkel analys som vi gjorde för nominalfraser. Den har en väldigt hög träffsäkerhet.

Nu är det endast frågan om vad som kommer härnäst? Som det redan antytts ovan betraktas endast den inre strukturen av en nominalfras. Den direkta konsekvensen är att fortsätta utöka informationsmängden som ställs algoritmen till förfogande. Här blir det bara att granska nominalfrasens kontext. Exempel:

- Efterföljs en nominalfras direkt av prepositionen *av* + nominalfras sammanfattar allt till en enda nominalfras.
- Börjar meningen med en nominalfras, lägg till allt som följer tills det finita verbet kommer.

Formaliserat blir reglerna (NP står inte för en bestämd nominalfras):

- NP + *av* + NP → NP
- satsbörjan + NP + flera ord + finit verb → (satsbörjan +) NP + finit verb

Dessa regler kan (i nästan samma form) översättas till datorspråk. Det bör anmärkas att analysen kommer att generera maximala nominalfrasgränser.

Referenser

Ejerhed, Eva, Gunnel Källgren, Ola Wennstedt and Magnus Åström. 1992. *The Linguistic Annotation System of the Stockholm – Umeå Corpus Project.* Department of General Linguistics, University of Umeå.

Källgren, Gunnel. 1984. *Automatisk excerpering av substantiv ur löpande text. Ett möjligt hjälpmedel vid datoriserad indexering?* Institutet för Rättsinformatik, Juridiska Institutionen, Stockholms universitet.

Källgren, Gunnel. 1992. *Making maximal use of surface criteria in large-scale parsing: the MorP-Parser.* Institutionen för Lingvistik, Stockholms universitet.

Thorell, Olof. 1977. *Svensk Grammatik.* 2:a upplagan, Norstedts, Stockholm.

Interlanguage and Set Theory

Atle Ro
Bergen

Abstract

If one is to exploit the notion of interlanguage in error diagnosis systems, a precise definition of this concept is useful. We will define interlanguage set theoretically, on two different levels. A comparison of a first language and a target language on the X^0-level makes it possible to compare structural similarity between languages, and a comparison on the V_T-level enables an explication of second language acquisition. We also comment on the limitations of set theory applied to interlanguage, and propose some augmentations which are needed in a theory of interlanguage that is to give a satisfactory account of interlanguage data.

0. Introduction

The notion 'interlanguage' alludes to a language "between" two (or more) languages, i.e. a target language (Lt) norm which a student is trying to achieve, and his first language (L1). The interlanguage has characteristics of both of these languages. The nature of the blending, or how "between" is to be interpreted, however, has always been vague in second language acquisition (SLA) literature. In this paper, we will try to make the concept so clear that it can be exploited in a computational system for diagnosing second language errors. In our study, Lt is Norwegian, and L1 is Spanish.

The main features of interlanguages which will be used in the diagnosing system, are overgeneralisation of Lt rule statements and transfer from L1. In the diagnostic system, overgeneralisation will be implemented as constraint relaxation along the lines of Douglas and Dale (1992), and transfer will be implemented by means of an alternative L1 based grammar. Transfer is understood in the sense it is used in SLA research (cf. Odlin (1989)), not in the sense of machine translation (although the planned system bears resemblances with transfer based MT systems).

In the main section of this paper 'interlanguage' is defined set theoretically. In Section two some features which we want in a theory of interlanguage, but which fall outside the set theoretical study in section one, are discussed.

1. Interlanguage – a Set Theoretic Definition

We want to define foreigners' interlanguage or second language in terms of the target language they aspire to master and their first language. Let the first language grammar be G1, and L(G1) the language generated by G1. Let the interlanguage grammar be Gint, and L(Gint) the language generated by the interlanguage grammar, i.e. the interlanguage. Furthermore let the target language grammar be Gt, and L(Gt) the target language.

We then define G1 as:

$$G1 = \langle V_T, V_N, V_{X0}, \{S\}, P \rangle$$

where V_T is lexical entries of L1, V_N = {NP, S, PP, VP, N, ...}, i.e. the set of grammatical categories, and V_{X0} = {N, V, A, P, CONJ, ADV}, i.e. X^0-categories. S is the axiom, and P the grammar rules of G1. We assume that V_T and V_N are disjoint sets. V_{X0} is a proper subset of V_N. Strings over V_{X0} will be called X^0-strings. An X^0-string is e.g. *Det N V N*, whereas a terminal string (a string of terminal symbols) is e.g. *a man eats sushi*. Gt is defined in the same way.

The languages generated by two grammars can now be compared on two levels, the V_{X0}-level and the V_T-level, and 'interlanguage' understood in terms of these two grammars. Both approaches are useful. The former makes it possible to express the degree of structural similarity between languages (with the possibility to explain both positive and negative transfer), and the latter enables one to explicate processes of language acquisition. We will first consider the former approach.

1.1 Comparison on the V_{X0}-level

It is possible that different grammars can generate languages which are equal on one level, but not on another. If the two languages are equal on the V_{X0}-level, i.e. that their sets of V_{X0}-strings are equal, the possibility that the grammars which generate them are different exists, but it is probable that the grammars are quite similar. On the other hand, if the two languages are different on the V_{X0}-level, the rules of the two grammars cannot be equal. As a working hypotheses or plausible assumption we propose that an interlanguage in terms of V_{X0}-categories, is something like what we see in figure 1.1:

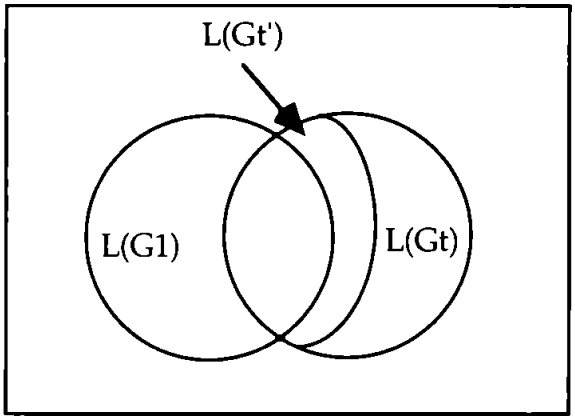


FIG 1.1: Interlanguage w.r.t. V_X0

L(G1) and L(Gt) overlap, and the degree of overlap is determined by the similarity between the two grammars. We make two assumptions about interlanguages:

1) the interlanguage user has a representation of Gt, which we will call Gt', and furthermore, Gt' is not a complete rendering of Gt. This implies that we assume that the full range of possibilities of Gt are not exploited in L(Gint), which means that L(Gt') is a subset of L(Gt).

2) L(Gint) contains strings which are not admitted by Gt, but by G1.

So preliminary we say that an interlanguage L(Gint) is the union of L(G1) and L(Gt') w.r.t V_X0.

1.2 Comparison on the V_T-level

Let us first compare G1 and Gt. Assume that both grammars have the same V_N. Assume further that the axiom is the same, and that the terminal vocabularies are disjoint. Thus the languages are completely different as in figure 1.2.1.

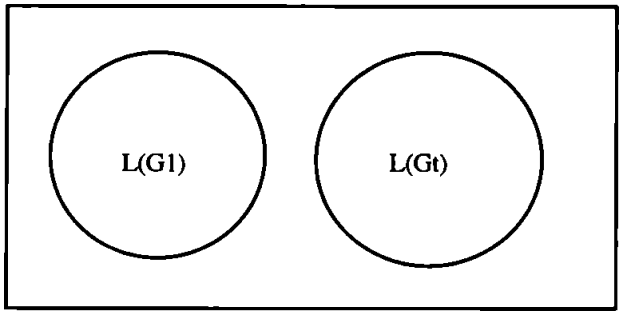


FIG 1.2.1: Comparison of L(G1) and L(Gt)

As soon as L(GI) $\cap$ L(Gt) $\neq \varnothing$ we have an interlanguage w.r.t V_T. This is, however, impossible, because the terminal vocabularies are disjoint. If we, on the other hand, assume that Gint = G1, where V_T corresponds to the empty set, this provides a model of interlanguage at the initial state. We assume that at the outset Gint is very similar to G1, at least w.r.t. production rules, but as it develops, rules from GT are added (acquired). At the outset, the vocabulary of Gint is very small, but increases during the acquisition.

We then define Gint like this:

$$\text{Gint} = \langle V_{T_t}, V_N, V_{X^0}, \{S\}, P \rangle$$

where V_{T_t} is lexical entries of the target language, V_N = {NP, S, PP, VP, N, ...}, i.e. the set of grammatical categories, V_{X^0} = {N, V, A, P, CONJ, ADV}, i.e. X^0-categories, S is the axiom, and P can contain grammar rules of both G1 and Gt.

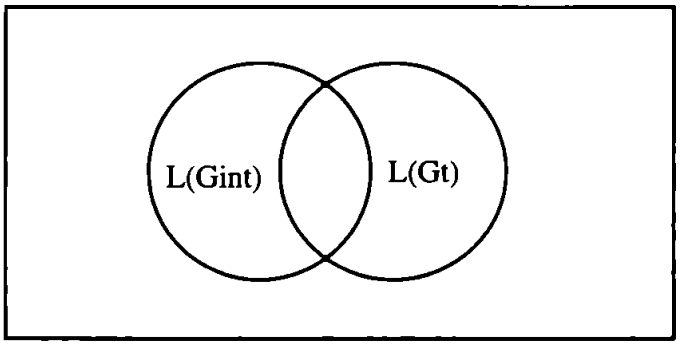


FIG 1.2.2: Interlanguage w.r.t. V_T

The intersection of the two languages is the subset of the interlanguage which is correct (cf. fig. 1.2.2) w.r.t. Gt. As the interlanguage develops, and the similarity between L(Gint) and L(Gt) increases, L(Gt) will be eclipsed to a varying degree by L(Gint).

An important theoretical issue is the following: how can we account for the fact that some rules from G1 are not present in Gint? It is hardly surprising that some rules of Gt are not present in Gint, we can explain this in terms of incomplete language acquisition. But how are some G1 rules excluded from Gint? Do rules from the Gt and G1 exclude each other in Gint, do they exist side by side, or both? The answers to these questions will tell us much about the mechanisms of SLA.

2. Interlanguage competence

Starting with the set theoretic study in section one, we already have a theory of interlanguage competence, albeit a very simple one. In this section we will first elaborate some of the assumptions made in section one, and then go on to discuss some augmentations which will bring the theory closer to the data we want to account for.

We assume that L(Gt') is a subset of L(Gt) (cf. section 1.1). How can we justify such a claim? Imagine that a rule of Gt is not in Gt', nor in G1. This rule licences a special type of strings (e.g. it-cleft sentences). Now there will be no instances of this type of strings in L(Gt'). It is natural that advanced rules of Gt are acquired at a later stage than the more basic ones like S -> NP VP, and this supports our assumption.

We also assume that L(Gint) contains X^0-strings which are admitted by G1 and not Gt. Examples of this kind of erroneous strings are Norwegian pseudo-sentences[1] displaying pro-drop and V2-violations. These errors in Norwegian are admitted by a Spanish G1.

In section 2.2 we claimed that Gint does not contain all the rules of G1. because some interlanguage errors (w.r.t. Gt) that should be accounted for by G1 rules never appear in interlanguage data. Therefore it is an oversimplification to say as we did in section 1.1 that an interlanguage is the union of L(G1) and L(Gt') w.r.t. V_{X0}. As for L(G1), we are dealing with a subset which is diminishing along with the progress of the student.

Now we will introduce some augmentations to the set theoretic account we have made so far. First we will introduce syntactic features in rules and lexical entries. We want to replace the simple rule format of section one with rules that refer to syntactic categories which are feature-bundles or attribute-value matrices (AVM's). We further assume that lexical entries in Gint may be underspecified w.r.t. syntactic features (compared with the corresponding Gt lexical entries), or even have wrong values for features. This augmentation will enable us to account for agreement

[1] By 'pseudo-sentence' is understood an ungrammatical string which is almost a sentence.

errors, non-finite verbs as heads of sentences etc. If we assume that rules refer to feature-bundles, and such rules of Lt are learnt in an incomplete or imprecise fashion, we can imagine that Gint partly is a deprecised and perhaps incomplete version of Gt. And errors like (1) can be accounted for.

(1)	*noen*	*liten*	*prosjekt*
	some-pl	*small-sg*	*project(s)*

This means that we must revise our notion of L(Gt') as a subset of L(Gt) (cf. fig. 1.1). L(Gt') contains strings which are not in L(Gt), like (1).

A theory of interlanguage competence should account for lexical transfer from L1. By lexical transfer we mean that Lt lexical items are assumed to have the same syntactic information associated with them as the corresponding L1 lexical items. With 'corresponding' we mean 'having the same meaning'. The example in (2) illustrates negative lexical transfer from Spanish.

(2)	**Jeg*	*kunne*	*ikke*	*svare*	*til*	*ham.*
	I	*could*	*not*	*answer*	*to*	*him.*

The Norwegian verb *svare* subcategorises for an object NP, while the Spanish verb with the same meaning, *responder*, subcategorises for a PP headed by the preposition *a* (to). If we assume that lexical items of L1 and Lt are linked to each other when they have the same meaning, subcategorisation information from the L1 lexical item can be used in generating the interlanguage string. Thus lexical transfer of the kind illustrated in (2) can be accounted for. To sum up, we assume that the interlanguage competence has access to the L1 lexicon, and lexical items of Gt and G1 are connected as outlined above. L1 word forms are not, however, used as "terminal vocabulary" in interlanguage strings.

3. Conclusion and future work

A set theoretical definition of the concept 'interlanguage' has been given, and a theory of interlanguage competence has been outlined. Future work will exploit the insight from this concept of interlanguage in developing a system for diagnosing ill-formed input based on overgeneralisation of Gt, and negative transfer from L1. This will be done by means of constraint relaxation of Lt rules, and an alternative L1 based grammar.

Acknowledgements

I wish to thank my supervisors Helge Dyvik and Torbjørn Nordgård for fruitful comments while writing the paper. I also wish to thank Patrizia Paggio and Ebbe Spang-Hanssen for pointing out limitations of set theoretical studies of interlanguage after reading an early version of the paper.

References

Douglas, Shona and Robert Dale. 1992. *Towards Robust PATR.* In *Proceedings of the 15th International Conference on Computational Linguistics*, Vol. 2, pp. 468–474, Nantes, France.

Odlin, Terence. 1989. *Language Transfer.* Cambridge University Press.

Morphological Tagging Based Entirely on Bayesian Inference

Christer Samuelsson
Stockholm

Abstract

The influence of various information sources on the ability of a statistical tagger to assign lexical categories to unknown words is investigated. The literal word form is found to be very much more important than other information sources such as the local syntactic context. Different ways of combining information sources are discussed. Methods for improving estimates based on scarce data are proposed and examined experimentally.

1 Introduction

Tagging is the art of assigning a specific label, a tag, to each word in a corpus from a pre-defined set of labels — a (tag) palette. The traditional problem addressed is that of disambiguation, see for example Church (1988). A lexicon states what different tags each word can possibly be assigned to, and for any specific word, this is a small subset of the palette.

Normally, the most likely assignment of tags to the words of a corpus is determined by statistical optimization using dynamic programming techniques, as is well-described in for example DeRose (1988). Some existing taggers, though, make use of hand-coded heuristic rules to guide the assignments, see for example Brill (1992) and Källgren (1991). Until recently, empirical results have indicated that statistical methods are superior to rule-based ones. However, the results reported in Voutilainen *et al* (1992) indicate that this may actully not be the case. However, rule-based approaches suffer from the major disadvantage of being *very* labour intense.

The approach taken here differs somewhat from mainstream tagging endeavours. Our main goal is not disambiguation, but to investigate the influence of various information sources, and ways of combining them, on the ability to assign lexical categories to unknown words. Here, we dispense with heuristic rules and lexical entries altogether, and instead rely entirely on the power of statistics to extract the required information from a pre-tagged corpus during a training phase. This has the advantage of making the tagger completely language independent.

The information sources employed are the literal appearance of the word and the tags assigned to the neighbouring words. The way these sources

of information are combined is novel for tagging applications. Another important innovation is the "successive abstraction" scheme for handling scarce training data by generalizing to successively wider contexts. A final original feature is the fact that the tagger is implemented entirely in SICStus Prolog.

Bayesian inference is used to find the tag assignment T with highest probability $P(T\ M,S)$ given morphology M (word form) and syntactic context S (neighbouring tags). This quantity is calculated from the probabilities $P(T\ /M)$ and $P(T\ /S)$. Before describing how the latter two quantities are estimated, we will address two important issues, namely those of combining them and of estimating the probabilities of events for which there is only a small number of observations.

2 Combining information

Several methods seem currently to be in use for combining information sources. One method is to simply set the probability of the hypothesis H given the combined evidence to the product of the probabilities of the event given each context:

$$P(H \mid M,S) \approx P(H \mid M) \bullet P(H \mid S)$$

Unfortunately, this can lead us far astray from the correct figure. For example, using Bayes' inversion formula, and assuming that these information sources are independent, and conditionally independent given the hypothesis H, i.e.

(1)

$$P(M,S) = P(M) \bullet P(S)$$
$$P(M,S \mid H) = P(M \mid H) \bullet P(S \mid H)$$

will yield us the exact formula

$$P(H \mid M,S) = (P(H \mid M) \bullet P(H \mid S)) / P(H)$$

This means that unlikely hypotheses will be unduly penalized in the above approximation by omitting the denominator $P(H)$. Even if these assumptions are not valid, this line of reasoning shows that intuitively, an extra factor proportional to the probability $P(H)$ is introduced.

Another method is to use a weighted sum of the probabilities:

$$P(H \mid M,S) \approx \lambda_M P(H \mid M) + \lambda_S P(H \mid S)$$

In general, the weight assigned to the morphological probability (λ_M) will be much larger than that assigned to the syntactic probability. The problem with this approach is that these weights are static, and not dependent on the relative predictive power of the two information sources in each particular case.

The approach taken here remedies these shortcomings: We wish to estimate the probability $P(H \mid e_1,\ldots,e_n)$ of the hypothesis H given the evidence e_i: $i = 1,\ldots,n$. We will go by way of the posterior odds $O(H \mid e_1,\ldots,e_n)$ (see for example Pearl (1988), pp. 34–39), which are defined by

$$O(A \mid B) = P(A \mid B) / P(\neg A \mid B) = P(A \mid B) / (1 - P(A \mid B))$$

We will make the independence assumption

(2)
$$O(H \mid e_1,\ldots,e_n) / O(H) \approx (O(H \mid e_1) / O(H)) \bullet \ldots \bullet (O(H \mid e_n) / O(H))$$

In our case the hypothesis H is the tag T and the evidence e_1 and e_2 are the word form M and the syntactic context S. Thus:

$$O(T \mid M,S) \approx (O(T \mid M) \bullet O(T \mid S)) / O(T)$$
$$P(T \mid M,S) = O(T \mid M,S) / (1 + O(T \mid M,S))$$

This formula has several advantages. Firstly, it is exact under the independence assumptions of Eq.(1), with the additional assumption

(3)
$$P(M,S \mid \neg H) = P(M \mid \neg H) \bullet P(S \mid \neg H)$$

which proves that it doesn't introduce an extra factor of $O(H)$. The relationship between equations Eq.(1), Eq.(3) and Eq.(2) is that the two former together imply the latter, but not vice versa. Secondly, using the odds instead of the probabilities has a stabilizing effect when none of the independence assumptions are valid. Thirdly, the impact of each of the two sources of information is allowed to depend dynamically on how much distinctive power they carry, rather than being prescribed beforehand, as is the case when using a weighted sum.

3 Handling scarce data

We now turn to the problem of estimating the statistical parameters for which there is only a small amount of training data.

3.1 Successive abstraction

Assume that we want to estimate the probability $P(E \mid C)$ of the event E given a context C from the number of times N_E it occurs in N trials, but that this data is scarce. Assume further that there is abundant data in a more general context $C' \supset C$ that we want to use to get a better estimate of $P(E \mid C)$.

If there is an obvious linear order $C' = C_m \supset C_{m-1} \supset \ldots \supset C_1 = C$ of the various generalizations C_k of C, we abstract successively to the lowest k for which data suffices. We will refer to this as "linear abstraction". A simple example is estimating the probability $P(T \mid l_n, l_{n-1}, \ldots, l_{n-j})$ of a tag T given the last $j+1$ letters of the word. The estimate will be based on the estimates of $P(T \mid l_n, l_{n-1}, \ldots, l_{n-j})$, $P(T \mid l_n, l_{n-1}, \ldots, l\_\{n-j+1\})$, ... ,$P(T \mid l_n, l_{n-1}, \ldots, l_{n-k})$, where k is the smallest number for which there is a good estimate of $P(T \mid l_n, l_{n-1}, \ldots, l_{n-k})$.

Even if there is no obvious linear order of the various generalizations, they might stem from a small number of sources, each of which has a linear order. An example is generalizing compound nouns using a sortal hierarchy. Here, the possible generalizations of the compound are the compounds of the generalizations of each noun. This can be used to explore the possible generalizations systematically and facilitates pruning using a heuristic quality measure of the estimates. If this measure is simply the total number of observations in the next generalized context, we will call this "greedy abstraction".

3.2 Improving estimates

Several different methods were tried for combining the estimates based on scarce data with estimates from a more general context — in Section 3.2.1 a confidence interval is used and in Section 3.2.2 weighted sums.

3.2.1 Using a confidence interval

To calculate an estimate $\boldsymbol{p}$ of the probability $p = P(E \mid C)$ we will use the fact that the quotient $\xi_n = N_E / N$, where N_E is the number of times E occurs in N trials, is a random variable with a binomial distribution, i.e.

$$\xi_n = N_E / N \sim \text{bin}(p, \sqrt{(p(1-p) / N)})$$

to get a first estimate $\boldsymbol{x}$ of p and a confidence interval $x_1 < p < x_2$ with confidence degree α, where $x_1 < \boldsymbol{x} < x_2$. Given these quantities,

- **If** for a pre-defined threshold θ,
  \label{EqThres}
  $$(x - x_1) / x < \theta \text{ and}$$
  $$(x_2 - x) / x < \theta$$

- **Then** set $p = x$. Here, we are confident ($100 \cdot \alpha$ %) that x is a good ($\pm 100 \cdot \theta$ %) estimate, and are satisfied with this.

- **Else** generalize the context C one step to C'; calculate an estimate $\hat{p}'$ of the probability $P(E \mid C')$ recursively; let $f(x)$ be some suitable function and set $p = f(\hat{p}')$. Examples of such functions are discussed below. Thus, $f(x)$ is used to let the observations of E in context C guide the estimate according to their reliability.

- Re-normalize so that $P(\Omega \mid C) = 1$ for the entire sample space Ω.

For example, using the fact that for large N, ξ_n is approximately normally distributed, that is

$$(\xi_n - p) / \sqrt{(\xi_n(1-\xi_n)/N)} \sim \text{norm}(0,1)$$

we can establish the confidence interval ($\beta = 1 - (1-\alpha)/2$)

$$p = \xi_n \pm t_\beta \sqrt{(\xi_n(1-\xi_n)/N)} \qquad (\alpha)$$

where $P(\eta \leq t_\beta) = \beta$ for a normally distributed random variable η with mean value 0 and standard deviation 1. In other words t_β is the β-fractile of the normal distribution.

Inserted into Eq.(4) this yields

$$(t_\beta \sqrt{(\xi_n(1-\xi_n) / N)}) / \xi_n < \theta$$

or with $\gamma = \theta / t_\beta$

$$N_E / N = \xi_n > 1 / (1 + \gamma^2 N)$$

to determine threshold values for N and N_E.

In a very simple version of the scheme, we could let $f(x)$ be a piecewise linear function such that $f(0) = x_1$, $f(x) = x$ and $f(1) = x_2$. Ideally $f(x)$ should be a continuous, monotonically increasing function, where $f(0) = 0$, $f(x) = x$ and $f(1) = 1$, and where the shape of $f(x)$ would be

continuously parameterized by the degree of certainty in x. A refinement of the simple version to match these criteria (omitting the degenerate cases where it does not hold that $0 < x_1 < x < x_2 < 1$ for the sake of brevity) is letting the function $f(x)$ be piece-wise linear with $f(0) = 0$, $f(t_1) = x_1$, $f(x) = x$, $f(t_2) = x_2$ and $f(1) = 1$, where say $t_1 = (1-\alpha)x$ and $t_2 = \alpha + (1-\alpha)x$.

3.2.2 Using a weighted sum

Another alternative is to use a weighted sum of the estimates:

$$p = \lambda x + \lambda' p'$$

We want λ and λ' to depend on N, the number of observations in the more specific context. We will also require that $\lambda + \lambda' = 1$. Two other desired properties are $N = 0 \Rightarrow p = p'$ and $\lim_{N \to \infty} p = x$.

A very simple strategy is to set p to p' if $N = 0$ and to x otherwise. This means that we abstract only if there are no observations at all in the specific context. We can view this as setting $\lambda(N)$ to a unit step for $N = 1$.

A less naive strategy draws inspiration from the standard deviation. Since the standard deviation behaves asymptotically as $1 / \sqrt{N}$ when N tends to infinity, we want $p - x$ to do likewise. Two different weighted sums meeting these criteria immediately spring to mind:

$$p = (\sqrt{N}\, x + p') / (\sqrt{N} + 1)$$

and

$$p = x + (p' - x) / \sqrt{(N + 1)}$$

The first one simply up-weights the specific estimate with $\sqrt{N}$, the active ingredient of the standard deviation. The second one interpolates linearly between p' and x. The distance from x is proportional to $1 / \sqrt{(N + 1)}$ (The "$+ 1$" is a technicality).

These three methods were pitted against the confidence-interval method in one of the experiments.

4 Information sources

We are now in a position to discuss how to extract morphological and syntactic information and formulate it as $P(H \mid M)$ and $P(H \mid S)$. The basic idea is to approximate these quantities with their relative frequencies. However, when this data is scarce, we will resort to the successive abstraction scheme of Section 3.

The literal ending of the word was inspected as it was suspected to contain crucial clues to the lexical category. For example, in English, any multi-syllable word ending with "-able" is almost certainly an adjective. This is even more accentuated in a language like Swedish, which has a richer inflectional and productive morphology.

To abstract, a letter was substituted with a vowel/consonant marker and abstraction was linear with earlier letters generalized before later. The last 0 – 7 letters of the words were taken into account in the experiments. The number of syllables in the rest of the word was another piece of evidence. The spectrum was zero–one–many and the single abstraction was to any number of syllables. These two generalizations competed in a greedy fashion.

The tags of the neighbouring words in the sentence were recorded. This is the conventional information source, and was believed to be very useful. However, in the experiments this information source proved much less important than the word form. Here, abstraction meant disregarding one of the neighbours at a time (the one furthest away from the word). N-gram refers to inspecting $N-1$ neighbours. Unigram through pentagram statistics were used in the experiments.

5 The experiments

The corpora used both for training and testing were portions of the Teleman corpus, a hand-tagged corpus of almost 80,000 words of miscellaneous Swedish texts (Teleman 1974). Three corpus sizes were tested in the experiments, 800, 7,500 and 65,000 words.

The tag palette used in the experiments is not Teleman's original one of around 250 different tags, but the usual set of lexical categories: Adjectives (adj), nouns, prepositions (prep), verbs, adverbs (adv), determiners (det), pronouns (pron) conjunctions (conj) and number (num), extended with proper names (name), sentence delimiters (eos), the infinitive mark "att " (inf) and characters (char), like "(", "$ " etc.

One observation of the correct hypothesis was removed when calculating its probability to simulate that this observation was not present in the training set. Important to note is that in the bulk mass of these experiments, the tags of the neighbouring words were not assigned by statistical optimization; instead the correct ones were used. This was done to allow gathering enough data to come to grips with the relative importance of the various information sources. However, for a few specific settings of the parameters, a dynamic programming technique was used to estimate the tags of the neighbouring words instead of using the pre-assigned ones.

The successive abstraction scheme employed a somewhat crude version of the confidence-interval method where the normal-distribution approximation was used for all observations except those of zero or all hits, for which the exact values from the binomial distribution were easily obtainable. The confidence level was 95 percent and the tolerance level ± 30 percent.

The task that the tagger carried out was to for each word in the corpus rank the set of tags according to the probability it assigned to them. Section 5.1 tabulates an overview of the results, while Section 5.2 examines one of the table entries in more detail, and Section 5.3 compares it with a tagging experiment were the neighbouring tags were estimated as well.

Section 5.4 compares various versions of the successive abstraction scheme.

5.1 Overview of the results

Table 1 shows an overview of the results given as token percent correct first alternatives, leading to a number of interesting conclusions.

The literal ending of the word is by far the most important information source. The neighbouring tags and the number of syllables are not at all as useful. Each extra letter seems to cost an order of magnitude in training data — the 800 word corpus peaks between 4 and 5 letters, the 7,500 word corpus between 5 and 6, and the 65,000 between 6 and 7 letters. Considering more than two neighbouring tags (i.e. using 4-gram and 5-gram statistics) improves the accuracy only marginally.

TABLE 1 : Token percent correct first alternatives.

Corpus size	Syllable information	Syntactic context	Number of final letters inspected 0	1	2	3	4	5	6	7
		1-gram	27.30	53.72	65.88	75.43	77.30	77.05	76.55	76.18
		2-gram	34.49	58.93	66.38	77.33	80.15	80.02	79.40	78.78
	any	3-gram	46.15	61.17	68.24	77.42	80.52	80.89	80.02	79.16
		4-gram	47.39	61.66	68.11	76.18	80.02	80.40	79.28	78.16
800		5-gram	47.52	62.28	67.87	76.55	79.65	80.27	79.40	78.66
words		1-gram	41.44	62.16	73.08	77.17	78.91	76.80	77.17	76.67
		2-gram	47.89	62.78	73.20	80.40	81.39	80.40	79.53	78.78
	0-1-2^+	3-gram	53.47	65.51	74.19	80.52	81.89	81.51	80.02	79.53
		4-gram	55.96	66.38	74.44	80.15	81.14	81.51	79.78	79.03
		5-gram	56.58	66.87	74.44	80.89	81.39	81.64	80.02	79.28
		1-gram	26.27	51.24	67.10	82.92	86.54	88.04	87.97	87.34
		2-gram	26.08	57.34	69.88	84.41	87.57	88.61	88.55	88.08
	any	3-gram	44.79	63.03	74.67	86.00	88.78	89.70	89.48	89.06
		4-gram	48.10	64.06	75.12	86.16	88.90	89.75	89.57	89.25
7,500		5-gram	48.39	64.33	74.79	86.01	88.90	89.72	89.72	89.40
words		1-gram	39.45	60.77	74.38	84.30	87.05	88.17	87.97	87.41
		2-gram	45.12	64.70	75.70	86.29	87.87	88.82	88.61	88.23
	0-1-2^+	3-gram	54.74	68.78	80.00	87.79	89.24	90.03	89.79	89.31
		4-gram	55.59	69.78	80.29	88.08	89.56	90.07	89.82	89.37
		5-gram	56.21	70.15	80.27	87.99	89.50	90.06	89.90	89.62
		1-gram	25.15	47.71	65.38	83.22	90.52	92.63	93.22	93.17
		2-gram	31.23	53.44	69.02	84.56	91.43	93.11	93.55	93.56
	any	3-gram	46.79	61.72	75.35	87.29	92.49	93.91	94.25	94.21
		4-gram	49.35	63.75	76.39	87.92	92.60	93.98	94.32	94.32
65,000		5-gram	51.22	64.94	76.46	88.16	92.74	94.18	94.46	??.??
words		1-gram	38.72	58.37	75.08	87.65	91.54	92.94	93.33	93.22
		2-gram	45.37	62.71	77.31	88.75	92.20	93.38	93.72	93.63
	0-1-2^+	3-gram	55.22	68.70	80.69	90.23	**93.15**	94.15	94.36	94.31
		4-gram	56.94	70.38	81.98	90.54	93.30	94.24	94.44	94.44
		5-gram	58.31	71.18	82.24	90.73	93.42	94.48	94.61	??.??

A final observation is the notorious "96 percent asymptote" reported from many statistical tagging experiments.

5.2 An expanded table entry

Table 2 shows an expanded entry from the previous table — that in boldface — where the four last letters, the number of syllable preceding those, and two neighbouring tags, were taken into account. The other entries exhibit the same general behavior.

Seeing that nouns and verbs are the most common word types, it is only reasonable that the total average should be close to the figures for these two word classes. Since the corpus is normalized, no distinction is made between capital letters and commons, and the tagger isn't doing too well on spotting names. Also, as one might expect, the tagger is having a bit of trouble telling adjectives from adverbs. A bit more surprising is that the tagger is performing so poorly on conjunctions and numbers, which are generally considered closed word classes, and should not be too difficult

to learn. The explanation to this is to be sought in the way the Teleman corpus is tagged.

TABLE 2 : 65,000 words, 3-gram syntax, 4 letters and syllable information.

Tag	1st	2nd	3rd	4-5th	6-10th	>10th	Observations
adj	85.84	10.48	2.35	1.08	0.25	0.00	4894
noun	95.39	3.43	0.83	0.33	0.03	0.00	16275
prep	98.04	1.46	0.26	0.09	0.14	0.00	7587
verb	91.71	6.18	1.30	0.61	0.20	0.00	10573
char	98.33	0.30	0.00	0.15	0.91	0.30	659
eos	99.89	0.09	0.00	0.02	0.00	0.00	4521
inf	99.47	0.46	0.00	0.00	0.08	0.00	1314
adv	88.51	7.43	2.78	1.12	0.17	0.00	4646
det	99.06	0.69	0.06	0.00	0.19	0.00	1600
pron	94.11	4.20	1.04	0.54	0.10	0.00	7184
conj	84.97	13.68	0.57	0.39	0.39	0.00	3340
num	87.07	4.36	1.15	3.06	4.13	0.23	1307
name	63.68	17.67	5.03	6.87	6.26	0.49	815
Total	93.15	4.89	1.06	0.59	0.30	0.01	64715

It is however note-worthy that the correct word class is among the two highest ranking alternatives over 98 percent of the time.

5.3 A dynamic programming version

In another version of the scheme, where dynamic programming was used to estimate the (two) neighbouring tags, rather than simply inspecting the pre-assigned ones, very similar results were recorded. This fact lends further strength to that claim that morphological information is of much greater importance than the local syntactic context.

A few settings of the various parameters were tested using this scheme, all yielding results conforming to those of Table 3, where the 65,000 word corpus was used, and the four last letters and the number of preceding syllables were employed as morphological information sources. The figure given is again token percent correct first alternatives.

5.4 Varying the successive abstraction scheme

Four different schemes for combining the accurate estimate p' from the general context with the potentially inaccurate estimate x from the specific context were tried out using 3-gram local syntactic information (i.e. two neighbouring tags), and inspecting the four final letters ot the word and the number of preceeding syllables.

TABLE 3 : Comparison between knowing and guessing neighbour tags.

Tag	Known tags	Guessed tags	No tags	Observations
adj	85.84	86.31	83.27	4894
noun	95.39	95.43	92.11	16275
prep	98.04	97.72	98.00	7587
verb	91.71	90.93	89.97	10573
char	98.33	97.57	98.63	659
eos	99.89	97.01	99.87	4521
inf	99.47	98.78	99.85	1314
adv	88.51	87.88	87.77	4646
det	99.06	98.94	98.88	1600
pron	94.11	91.65	95.16	7184
conj	84.97	85.99	79.61	3340
num	87.07	87.83	84.85	1307
name	63.68	65.15	59.63	815
Total	93.15	92.59	91.54	64715

The four strategies were:[1]

1. The confidence interval method as described above.

2. $\boldsymbol{p} = \boldsymbol{p}'$ if $N = 0$,
   $\boldsymbol{p} = \boldsymbol{x}$ if $N > 0$.
   We abstract only if there is no data available at all.

3. $\boldsymbol{p} = (\sqrt{N}\,\boldsymbol{x} + \boldsymbol{p}') / (\sqrt{N} + 1)$.
   The weight of the specific result is simply $\sqrt{N}$ and the sum is normalized.

4. $\boldsymbol{p} = \boldsymbol{x} + (\boldsymbol{p}' - \boldsymbol{x}) / \sqrt{(N + 1)}$.
   The result is on the line between the specific and the general estimate. The distance from the specific eatimate is proportional to $1 / \sqrt{(N + 1)}$.

The results shown in Table 4 reveal that the last two strategies, the weighted-sum methods, are quite superior to the first two, the first one of them being the slightly better. Strategy 1, the confidence-interval method, is only somewhat better than not abstracting at all until forced to, as is done in strategy 2, when both syntactic and morphological information is taken into account.

The explanation for this could be the following: Even though data might be scarce, what is there is there, and those particular observations are more likely to be there as a result of having a higher probability, than by pure chance.

[1] Again N is the total number of observations in the specific context.

With only 800 words, the data can safely be assumed to be scare and the successive abstraction scheme improves the parameter estimates considerably. This is especially true when syntactic and morphological information is combined, and something less coarse grained than a mere ranking of the alternatives is required. Already with 7,500 words, though, the improvements are small and for 65,000 words, where one would expect sufficient data to be available for most estimates, the improvements are marginal. However, at least strategy 3 does not seem to degrade performance.

The best result observed, 95.38 percent, was for the setting of 6 letters, syllable information, 4-gram syntax (three neighbouring tags) and strategy 3 on the 65,000 word corpus.

TABLE 4 : Comparison between different successive abstraction schemes.

Corpus	Strategy	1	2	3	4
800 words	Syntax and morphology	81.89	78.29	86.35	85.86
	Morphology only	78.91	83.62	84.49	84.12
	Syntax only	46.15	48.88	46.03	46.53
7,500 words	Syntax and morphology	88.78	88.74	91.14	90.94
	Morphology only	87.05	90.07	89.83	89.48
	Syntax only	44.79	45.90	45.57	45.82
65,000 words	Syntax and morphology	93.15	93.83	94.06	93.78
	Morphology only	91.54	92.90	92.76	92.46
	Syntax only	46.79	46.80	46.87	46.89

6 Summary and conclusions

A number of interesting results emerged from these experiments. Even though it is not very surprising that the literal appearance of a word is a much more important information source than its local syntactic context for assigning the correct lexical category, it *is* surprising that it is so much more important. The *global* syntactic context, on the other hand, has proved very useful as reported in (Voutilainen *et al* 1992).

The design of the tagger relies heavily on the successive-abstraction scheme. The results are a success for the scheme even though it is a bit disappointing that the simpler weighted-sum method out-performed the more elaborate confidence-interval method. The moral might be phrased "If one wants a point estimate, one shouldn't stare too intensely at confidence intervals".

One tends to consider the Teleman corpus is a bit oddly tagged seeing that the tagger is having difficulties assigning the correct tag to closed class words such as conjunctions, numbers and pronouns.

Finally, the peak performance value of the tagger, 95.38 token percent correct first alternatives, is quite respectable in itself. However, two other approaches to the same task indicate that this result can be improved on. Cutting (1994) attempts the same task by using a lexicon and an untagged corpus to train from, making predictions using only bigram syntactic information in addition to lexical probabilities, and reports 95 percent success rate. This is probably a somewhat more difficult task. Eineborg and Gambäck (1994) report a success rate of 96.3 percent using a neural net with 4-gram statistics and six letter endings. They employ an intermediate abstraction level based on grouping the letters into phonological classes such as fricatives, explosives etc. This could readily be incorporated into the scheme described in this paper and could potentially improve its performance.

Acknowledgements

This work was made possible by the basic research programme at the Swedish Institute of Computer Science (SICS). I would very much like to thank my friend and colleague Jussi Karlgren for inspiring discussions and for pointing me to related work, Gunnel Källgren at Stockholm University for support and encouragement, Krister Lindén at Helsinki University for introducing me to the excellent work by Finnish researchers in this field and Dave Carter and Manny Rayner at SRI International, Cambridge, for comments and valuable suggestions to improvements on draft versions of this paper. I enjoyed very much the friendly competition from Douglass Cutting, Martin Eineborg and Björn Gambäck and the opportunity to discuss and compare our different approaches. I finally thank my other colleagues at SICS for contributing to a very conductive atmosphere to work in.

References

Brill, Eric. 1992. *A Simple Rule-Based Part of Speech Tagger.* pp. 152–155 of PROCS. 3RD ANLP , Trento, Italy.

Church, Kenneth W. 1988. *A Stochastic Parts of Speech and Noun Phrase Parser for Unrestricted Text.* pp. 136–143 of PROCS. 2ND ANLP.

Cutting, Douglass. 1993. *A Practical Part-of-Speech Tagger.* In Eklund (ed.), *Nodalida '93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3–5 June 1993*, Stockholm, Sweden.

DeRose, Steven J. 1988. *Grammatical Category Disambiguation by Statistical Optimization.* pp. 31–39 of COMPUTATIONAL LINGUISTICS, Volume 14, Number 1.

Eklund, Robert, (ed.). 1994. *Nodalida '93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3–5 June 1993*, Stockholm, Sweden.

Eineborg, Martin and Björn Gambäck. 1994. *Tagging Experiments Using Neural Networks.* In Eklund (ed.), *Nodalida '93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3–5 June 1993*, Stockholm, Sweden.

Källgren, Gunnel. 1991. *Parsing without lexicon: the MorP system.* PROCS. of 5TH EACL, Berlin, Germany.

Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Mateo, California.

Teleman, Ulf. 1974. *Manual för grammatisk beskrivning av talad och skriven svenska* (in Swedish), Studentlitteratur, Lund, Sweden.

Voutilainen, Atro, Juha Heikkilä and Arto Anttila. 1992. *Constraint Grammar of English.* Publication Number 21, Deptartment of General Linguistics, University of Helsinki, Finland.

Vad jag i min verksamhet som rättsinformatiker och jurist skulle vilja att datorlingvistiken bidrog med

Peter Seipel
Stockholm

1. En kort presentation av rättsinformatiken

Rättsinformatiken är en gren av rättsvetenskapen, en ganska ny gren, bör tilläggas. Rötterna går tillbaka till åtminstone 1950-talet men då var ännu inte själva benämningen "rättsinformatik" lanserad. Den kom först senare, främst genom den tyskspråkiga terminologin under 1970-talet.[1]

Koncentrerat uttryckt är rättsinformatiken *tvärvetenskaplig*, även om rättsvetenskapen står för det dominerande inslaget. Juridiken är värdämnet, kan man säga. Till de omgivande fält som ger inspiration, idéer och metoder hör systemvetenskap och datalogi, statsvetenskap och organisationsteori, den moderna logiken och – sist men inte minst – lingvistiken, särskilt datorlingvistiken.

Man kan beskriva rättsinformatiken genom en indelning i några huvudområden:

- Informationsbehandling och informationssystem
- Teoribildning om rättsnormer och rättslig styrning
- Teoribildning om analys, utredning och beslutsfattande
- Praktiskt arbete med datorhantering av juridisk information
- Datarätten – ett sidospår i detta sammanhang

För att skapa förutsättningar för en meningsfull genomgång av datorlingvistikens uppgifter i rättsliga sammanhang finner jag det nödvändigt att kort kommentera vart och ett av dessa områden.

Informationsbehandling och informationssystem

Rättsinformatiken sysslar med informationsbehandling inom juridiken, särskilt med inriktning på informationssystem och (modern)

[1] Se närmare Peter Seipel, *Perspectives on a New Legal Discipline*. Stockholm: Liber 1977.

informationsteknologi. Det är knappast någon skam att konstatera att ämnet mer har sitt ursprung i olika praktiska frågor som uppkommit genom datoriseringen än i teoretiska spekulationer. Eller man kanske snarare skall formulera det så: om inte datoriseringen av samhället hade blivit så genomgripande, skulle knappast de teoretiska spekulationerna om rättsligt intressanta aspekter på informationsbehandling ha fått något större genomslag.

Några exempel på de intressen det handlar om:

(a) Rättsväsendets informationssystem

Detta är namnet på ett flertal projekt som drivits ända sedan 1960-talet i den svenska s.k. justitieförvaltningen. Delar av verksamheterna, de som har att göra med information om brott, brottspåföljder m.m., regleras i en särskild förordning.[1] Det långsiktiga målet för verksamheterna – det handlar om flera, parallella projekt – är att skapa samordnade, automatiserade och datorstödda informationssystem inom rättsordningen. En grupp projekt kallas "Systemet för lagstiftningsförfarandet och rättspraxis, LAGRI". Målet för LAGRI är att steg för steg skapa ett väl integrerat informationssystem för texter som utgör rättskällor eller har anknytning till dessa – offentliga utredningar, motioner och propositioner i Riksdagen, författningstexter, referat av domstolarnas avgöranden m.m. En grundtanke är att det handlar om en sammanhängande kedja som går från information om lagstiftningsinitiativ via information om lagstiftningsarbetet till antagna författningstexter och domstols- och myndighetsinformation om hur texterna tillämpas. Man har talat om "lagstiftningscirkeln" och "rättskedjan".

(b) Datorstöd i rättsutbildningen

Området har dragit till sig växande uppmärksamhet under de allra senaste åren.2 En av anledningarna är att texter har grundläggande betydelse i rättsutbildningen: det är fråga om att finna texter, att sammanställa texter, att tolka texter, att själv författa texter. I och med att texter i allt större utsträckning flyttar in i den digitala miljön förändras förutsättningarna och möjligheterna för utbildningen av jurister. Som ett exempel kan de "elektroniska kursböckerna" tjäna. Eleverna får materialet till en kurs i form av notboksdatorer: en del material är förlagrat på hårddisken, annat skall tillföras av studenterna själva genom sökningar i datasamlingar tillgängliga on-line, annat skall presteras genom att bearbeta det förlagrade materialet, genom att utnyttja arbetstillfällen vid seminarier,

---

[1]Förordningen (1970:517) om rättsväsendets informationssystem.

[2]En aktuell översikt finns hos Peter Seipel, *CAI och rättsutbildning*. I: *Festskrift till Jacob W F Sundberg*. Stockholm: Juristförlaget 1993.

kontakter över datanätet med läraren o.s.v.[1] Nya hjälpmedel av detta slag förändrar synen på juridikutbildningens didaktik. Elektroniskt umgänge med rättstexter skapar både nya möjligheter och nya hinder och trösklar.

(c) Elektronisk meddelandeutväxling i handel, administration och transporter

Ett tredje och sista exempel avser s.k. EDI (Electronic Data Interchange). Med viss förenkling går EDI att beskriva som utväxling av meddelanden i starkt standardiserade former mellan datorer för att automatiskt sköta sådant som fakturering, beställningar, skriftväxling med tullmyndigheter, skriftväxling med domstolar m.m. Många av dessa informationsutbyten har juridiskt intresse och kan utgöra själva grunden för en viss rättslig reglering. Juridiken har följaktligen stor betydelse när EDI rutiner byggs upp och teknik och juridik flätas i varandra på sätt som många gånger är minst sagt komplicerade.

Teoribildning om rättsnormer och rättslig styrning

Att rättsinformatiken är praktiskt förankrad i olika datoriseringsverksamheter betyder inte att teoribildningen skulle vara ett svagt eller undanskymt intresse. Tvärtom strävar forskarna inom rättsinformatiken efter att utveckla en djupare förståelse av rättslig informationsbehandling. Detta sker ofta enligt linjer som är välkända inom traditionell rättsteori och rättsfilosofi. Från den synpunkten kan man uppfatta rättsinformatiken som en del av den s.k. allmänna rättsläran (jurisprudensen).

Dessa teoretiska intressen kan man återfinna på olika nivåer. Högt upp är det fråga om sådant som rättsliga styrsystem och den totala rättsordningen betraktad i system- och informationsperspektiv. Tankegångar från kybernetiken är inte främmande. Frågorna gäller sådant som rättsliga styrmedel och deras effektivitet och anpassbarhet. Det kan också vara fråga om att anlägga informationsperspektiv på någon viss rättslig reglering. Lagstiftningen om envars rätt att få tillgång till allmänna handlingar kan t.ex. diskuteras i termer av ett samhälles behov av flexibla styrsystem, vilka medger att problem kan formuleras och hanteras utan förvrängningar och bortträngningar. Ett slutet samhälle är – i systemteoretisk mening – en organism med otillräcklig förmåga att bemästra sin situation och de problem dess omgivning ställer det inför.[2]

---

[1] Se t.ex. Ronald W. Staudt, *An Essay on Electronic Casebooks: My Pursuit of the Paperless Chase*. I: Chicago-Kent Law Review, Vol 68 (1993) No 1, 291.

[2] Jfr James G Miller, *Living Systems*. New York...: McGraw-Hill 1978, särsk. sid. 785–788.

På mellannivåerna finner man intressen i teman som skapandet av rättsliga normer – t.ex. lagstiftningslära i allmänhet och frågor om datoranpassad lagstiftning.[1]

På lägre nivåer trängs "småproblemen" (som inte alltid är så små): Hur ordnar man lämpligen ett system av korshänvisningar mellan författningar? Hur bör texten till ett rättsfallsreferat vara organiserad för att på bästa sätt kunna läsas och förstås?

Teoribildning om analys, utredning, beslutsfattande

Rättsinformatiken anknyter här till en lång tradition inom jurisprudensen. Vad som kännetecknar dess intressen är att automationen av informationsbehandling påfallande ofta reser fundamentala frågor om juristens tänkesätt och argumentationstekniker. För att t.ex. kunna förstå förutsättningarna för utvecklingen och användningen av artificiell intelligens för s.k. expertsystem och andra typer av beslutsstöd inom juridiken är det nödvändigt att ta flera steg tillbaka och på nytt – för det är sannerligen inte första gången i historien – ställa frågor om fundamentala förhållanden. Vad är en "rättsnorm"? Hur går det till att inordna ett konkret fall under en viss rättsregel? Hur hänger informationssökning samman med argumentation och processtaktik? Och så vidare. Man kan tala om dekomposition av rättslig argumentation: element, led, tidsordning, beroenden etc.[2]

Ett tema som ofta dragit uppmärksamheten till sig gäller rättssäkerhet, t.ex. i samband med rättslig informationsökning. Ett argument för dyrbara och omfattande satsningar på datorisering kan vara att rättsäkerheten ökar. Men hur konstaterar man detta? Går det att åstadkomma några prognoser eller kalkyler?[3]

Praktiskt arbete med datorhantering av juridisk information

Den praktiska användningen av informationsteknologin inom juridiken betyder åtskilligt för rättsinformatiken – inte bara för att förklara dess

---

[1]Om lagstiftningslära i allmänhet se Jan Hellner, *Lagstiftning inom förmögenhetsrätten. Praktik, teori, teknik.* Stockholm: Juristförlaget 1990. Om datoranpassad lagstiftning och anknytande ämnen se Cecilia Magnusson-Sjöberg, *Rättsautomation. Särskilt om statsförvaltningens datorisering.* Stockholm: Norstedts Juridik 1993, särsk. sid. 65–69, 181 ff.

[2]Detta angreppssätt finner man hos Peter Wahlgren, *Automation of Legal Reasoning.* Deventer: Kluwer 1993.

[3]Jfr t.ex. *Vissa Rättsdatafrågor. Förslag av samarbetsorganet för rättsväsendets informationsystem (SARI) med anledning av en rapport av 1991 års RÄTTSDATA-grupp.* Ds 1991:75, särskilt sid. 85–87.

framväxt utan också som motivering till dess val av många arbetsuppgifter. Några exempel är klassificering av författningar för lagring i databaser, datorstöd på advokatkontoret och domstolen och utformning av vägvisare i det världsvida, akademiska datanätet Internet. Det kan handla om sådant som är direkt praktiskt användbart men som inte har några långtgående teoretiska ambitioner – således tillämpad forskning eller ren utveckling snarare än grundforskning. Naturligtvis kan lösningen av närliggande, praktiskt angelägna problem många gånger utvecklas vidare mot teoribildning och långsiktig metodutveckling.

Datarätten – ett sidospår i detta sammanhang

Här handlar det om reglering av informationsbehandling i datoriserade sammanhang. En möjlig grovindelning av de områden som ingår i datarätten ser ut på följande vis:[1]

Avtal, upphandling
Telematikmarknaden (IT och telekommunikationer)
Informationsfrihet
Registerlagar
Säkerhet och sårbarhet
Förvaltningsautomation

Till det som kan ge datarätten ett visst intresse i språkvetenskapliga sammanhang hör de ofta förekommande beskrivningsproblemen.[2] Annorlunda uttryckt ger den rättsliga regleringen av elektronisk, digital informationsbehandling och informationsöverföring ofta anledning att på djupet syssla med begreppsbildning och terminologi. Ord som "dokument", "original", "äkta", "skrift", "förvar" o.s.v. blir problematiska i sin nya omgivning. Det är långt ifrån någon trivial uppgift att konstruera "minispråk" som klarar sina många uppgifter i rättstillämpningen: att vara tillräckligt precisa, att inte låsa rättsreglerna vid något visst skede i informationsteknologins utveckling, att vara lätta att förstå och använda för olika medverkande i rättslivet, att kunna användas i en blandad miljö med olika informationsmedier o.s.v.[3]

---

[1]Se närmare t.ex. Peter Seipel, *Juristen och datorn. Introduktion till rättsinformatiken.* 4:e uppl. Stockholm: Norstedts Juridik 1993 och Mads Bryde Andersen, *Lærebog i EDB-ret.* Köpenhamn: Jurist- og Økonomforbundets Forlag 1991.

[2]Dessa beskrivningsproblem kan behandlas från många synvinklar, se t.ex. Peter Seipel, i not 1 anfört arbete, sid. 256–258 och passim. Hos Mads Bryde Andersen spelar beskrivningsproblematiken en huvudroll i monografin *EDB og ansvar. Studier i edberstatningsrettens beskrivelsesproblematik.* Jurist- og Økonomforbundets Forlag 1989.

[3]Den som vill få en god uppfattning om dessa frågor kan läsa Datastraffrättsutredningens över 600 sidor långa betänkande *Information och den nya InformationsTeknologin – straff och processrättsliga frågor m.m.*, SOU 1992:110.

2. Juridiken och juridikens texter

Rättskälleläran

För alla som arbetar med juridikens texter – det gäller såväl jurister som lingvister och andra – är det nödvändigt att vara bekant med vad juristerna brukar kalla rättskälleläran. I korthet handlar denna om att rättstexter har olika formell status och olika vikt för den juridiska argumentationen. Ett uttalande i ett visst lagstiftningsärende av en riksdagsledamot i en motion väger lätt som en fjäder jämfört med vad departementschefen sagt i propositionen. I ländernas rättssystem förekommer varierande rättskälleläror, det finns familjebildningar och det finns gemensamma egenskaper. För svensk del gäller som huvudregel att de egentliga rättskällorna utgörs av författningar med anknytande förarbeten (främst i propositioner och regeringens förordningsmotiv), prejudikat från de högsta domstolarna och doktrinen, d.v.s. den rättsvetenskapliga litteraturen.

Med nära anknytning till rättskälleläran eller – skulle många säga – som en integrerad del av denna har juridiken utvecklat läror och principer om texttolkning. Dessa leder vidare till argumentationstekniker och av skråets accepterade strategier för att hyfsa problemen och bygga under lösningar. Några exempel: snäv tolkning av straffbud; senare tillkomna lagar ges företräde framför äldre; en serie rättsavgöranden kan tolkas "aktivt" för att konstruera en rättsprincip som kanske har endast svagt stöd i varje enskilt avgörande.

2.2. Juridikens språk, normativa funktioner

De speciella sammanhang där det juridiska språket används och utvecklas ger det i många avseenden en särprägel. Man kan diskutera denna särprägel med anknytning till skilda funktioner som rättsspråket skall fullgöra. Det handlar om språkets *dirigerande* funktioner, behandlade inom rättsfilosofin med termer som "performativer" och "fristående imperativer". Det handlar om dess *kommunikativa* funktioner och dess *deskriptiva* funktioner, om dess *konstruktiva* funktioner och om dess *politiska* funktioner. Säkert kan man urskilja ytterligare funktioner – till och med ganska oväntade sådana. Ett vagt läsminne från min tidiga forskartid handlar om rättsspråkets tröstande funktioner – en dom t.ex. skrivs på ett sätt som får den förlorande parten att acceptera sitt nederlag och den lösning av en konflikt som han måste finna sig i.

Dessa olika funktioner, vilka alla går att diskutera inom olika språkvärldar (pedagogiken, medicinen, journalistiken o.s.v.), ger möjlighet att uppmärksamma sådant som rättstexters förmåga att överbringa budskap

till olika adressater – jfr en skatteförfattning i Svensk författningssamling med en text i Riksskatteverkets deklarationsanvisningar för löntagare. Inte minst intressanta är rättsspråkets deskriptiva funktioner. Vad är det egentligen som "beskrivs" i en författningstext? I någon mening en verklighet – faktiska situationer kopplade till önskade handlingsmönster. I en annan mening hypotetiska, önskade tillstånd och positioner – element i rättsliga konstruktioner som bildar en abstrakt verklighet i sig. Därmed kommer man in på rättsspråkets konstruktiva funktioner – att utgöra styrmodeller och fylla funktioner vid rättslig styrning i samhället. En rättslig reglering kan t.ex. analyseras med utgångspunkt i hur pass väl regleringen förmår förmedla information om rättstillämpningen och om det reglerade området till de lagstiftande organen. I kybernetiska termer kan man tala om en återkopplingsfunktion (feed back) hos rättsnormerna. För att välja ett enkelt exempel: först när ett visst beteende kriminaliseras börjar brottstatistik skapas kring det aktuella beteendet. Denna diskussion leder oss snabbt in i avancerade, teoretiska resonemang som får anstå till en annan gång. Låt mig bara runda av med att peka på de politiska funktionerna, som bl.a. har att göra med svårigheterna att få författningstexter genom lagstiftningsmaskineriet. Den slutliga produkten visar ofta tydliga spår av kompromisser, anpassningar, nödvändig tystnad, tolkningsföreträden o.s.v. Spåren kan ge sig till känna också i sådant som systematiken och rubriksättningen i författningstexterna.[1]

2.3. Textanvändning i praktiken, ekonomiska frågor

Juridikens texter, dess primära arbetsmaterial, bildar väldiga volymer. I datortermer talar vi inte sällan om megabytes och gigabytes . Redan på 1970-talet hördes tal om "juridikens informationskris".[2] Det är inte enbart fråga om rättskälletexter i traditionell, svensk mening (jfr ovan). Också sådant som brevväxling, utredningar och räkenskaper kan bilda omfattande material som måste kunna hanteras i samband med förhandlingar, processförberedelser, rättsutredningar m.m.[3] Datorstödda metoder håller mer och mer på att visa sig nyttiga eller till och med oumbärliga i sådana sammanhang.

---

[1]Det är intressant att lägga denna aspekt på t.ex. Britt-Louise Gunnarssons monografi *Lagtexters begriplighet. En språkfunktionell studie av medbestämmandelagen*. Lund: LiberFörlag 1982.

[2]Spiros Simitis, *Informationskrise des Rechts und Datenverarbeitung*. Karlsruhe: C F Müller 1970. Ett senare, mer polemiskt bidrag är Björn Tarras-Wahlberg, *Avreglera mera. Kostnader och effekter av lagar och regler*. Stockholm: SAF:s förlag 1983. Se också Bedre struktur i lovverket. NOU 1992:32. Oslo: Statens Forvaltningstjeneste 1992

[3]Ronald W Staudt, James I Keane, *Litigation Support Systems. An Attorney's Guide. New York...*: Clard, Boardman, Callaghan 1992. Staudt och Kean talar om "megacases" men framhåller att detta inte är den enda situation där rättslig texthantering mår väl av olika former av datorstöd.

Ser vi särskilt på informationssökning kan texternas omfattning och spridning på olika håll ställa till allvarliga problem. En grundläggande anledning finner vi i ""armlängdslagen":[1]

$B = 1/D^2$

Lagen – formulerad halvt på skämt och halvt på allvar – säger att benägenheten att leta fram viss information (B) avtar med kvadraten på avståndet till informationen. Helst arbetar man bara med sådant som man enkelt når vid skrivbordet och bekvämt kan bläddra fram. Mer sällan reser man på sig och vandrar iväg till biblioteket eller konsulterar en text som består av flera band, där jag inte vet i vilket jag skall söka. Avstånden är i själva verket av många slag – språk, ämnesområde, sökmöjligheter o.s.v. Undersökningar av juristernas arbetsvanor m.m. visar att man ofta anser sig ha alltför litet tid till sökning, läsning och analys. Det behövs effektiva arbetsverktyg och datorn utgör ett sådant – rätt utnyttjad, så att den inte lägger ytterligare ett avstånd till alla de som redan finns.

Den juridiska professionen blir allt mer uppmärksam på informationsteknologin som ett stöd i arbetet. I mars i år hölls i U.S.A. den väldiga mässan "TechShow -93". Den är årligen återkommande och ägnas enbart åt datorstöd i juristyrket. Den är stor som en ordinär Älvsjömässa. Seminarierna och föreläsningarna i anslutning till mässan behandlade denna gång teman som "Technology and Total Quality Client Service", "On the Edge of the Internet Breakthrough – Implications for Lawyers" och "How to Design Your Law Office in Cyberspace". De användningar av datorer som dominerar i praktiken torde vara hjälpmedel för avancerad ordbehandling, inklusive sådant som struktureringsverktyg, datorstödd dokumentframställning ("document composition" – där också AI finns med i bilden), dokumenthantering i databaser och, givetvis, informationssökning.

I teknikens släptåg följer också ekonomiska överväganden. Det handlar om att rationalisera juristyrket, att göra små advokatbyråer konkurrensmässiga med stora, att ge möjligheter att expandandera in i nya yrkesområden m.m. Betalningsviljan och betalningsförmågan är ganska stora – om man kan få nyttan och den praktiska användbarheten belagd.

---

[1]Peter Seipel, *Juristen och datorn. Introduktion till rättsinformatiken.* Stockholm: Norstedts Juridik 1993, sid. 121–123.

2.4. Internationaliseringen

Till sist finns det anledning att uppmärksamma juridikens internationalisering som det just nu talas och skrivs mycket om, inte minst med anledning av närmandet till EG.[1] Vad som särskilt intresserar här är den nya "textmiljö" som håller på att växa fram. Grundläggande är naturligtvis att det juridiska arbetsmaterialet ökar ännu mer i omfattning jämfört med tidigare. En mängd nya rättsföreskrifter skall införlivas med den svenska rättsordningen och nya kategorier av texter med utländsk hemort får intresse i den svenska rättstillämpningen. Detta betyder även mer komplicerade rättssituationer, nya behov av att jämföra och beakta parallella rättstexter på olika språk och kollisioner och konkurrens mellan juridiska tolkningsläror. Den starkare internationaliseringen innebär sammanfattningsvis att juristernas umgänge med texter blir rikare på utmaningar, ofta mer komplicerat och med större krav på effektiva metoder för texthanteringen (att återfinna, att strukturera, att jämföra o.s.v.).

3. Den nya, digitala miljön

Den "post-dokumentala" situationen

Den nya, digitala miljö där rättstexter skrivs, registreras, förmedlas, analyseras o.s.v. innebär förändringar. Det finns skäl att anta att dessa förändringar kan visa sig mer grundläggande än vi ännu insett. Några funderingar om detta är på sin plats.[2]

Gränserna förskjuts när det gäller de informationsmängder som organisationer och individer har förmåga och intresse av att kunna hantera. Det handlar om gränser både uppåt och nedåt. En liten juristbyrå kan genom den nya tekniken skaffa sig åtkomst till och möjligheter att söka i förråd av texter som den tidigare av olika skäl måst avstå från. Ett enkelt exempel ger oss Riksdagens allmänt tillgängliga söksystem Rixlex, som lagrar hela texter tillkomna i Riksdagens arbete och som även byggs på med historiskt material. Nu finns således texten till en proposition i ett udda lagstiftningsärende – som den lilla juristbyrån tidigare aldrig skulle ha övervägt att skaffa och ställa på sin egen hylla – omedelbart åtkomlig via datanätet. Exemplet kan lätt utvidgas och ges globala dimensioner. Sett från en annan synpunkt möjliggör digitaliseringen ett umgänge med *små textfragment* som tidigare blivit användbara först för den enskilde

---

[1] *Europagemenskap och rättsvetenskap. Utredning utförd av de juridiska fakulteterna på uppdrag av regeringen.* Uppsala: Iustus Förlag 1992.

[2] De utvecklas närmare i Peter Seipel, *Law Libraries and Information Technology. Notes from a workshop at the Chicago-Kent College of Law*, 6 April 1993. Under publicering i Juridisk Tidskrift.

läsaren. Med andra ord börjar t.ex. biblioteken att arbeta på nivåer under de klassiska – boken, tidskriften, rapporten och rapportserien.De får möjligheter och intresse av att urskilja mindre enheter för klassificering, sammanlänkning och åtkomst.

I denna nya miljö förändras synen på vad som är välstrukturerat och vad som är ostrukturerat. Juridikens traditionella "informationskris" betraktas med nya ögon. En aspekt på detta är att traditionella verktyg för systematisering visar sig otillräckliga: hur förfar man t.ex. med de texter som genereras i en expertdiskussion som förs i form av en datorkonferens? Hur gör man med implicita samband mellan rättstexter, t.ex. att ett visst direktiv från EG-kommissionen med fördel bör läsas i ljuset av vad som sagts i en expertkommitté under GATT? Kan sådant bli ögonblickligt åtkomligt på förfrågan eller måste man lita till den traditionella kommenterande analysen i efterhand?

Mot denna bakgrund har man att diskutera olika verktyg och möjligheter att helt automatiserat eller med olika grader av datorstöd strukturera och hantera texter (minns att det ofta skall ske på megabytesnivå). Traditionella hjälpmedel från biblioteksvärlden, som det universella decimalklassifikationssystemet UDK, måste kompletteras med sådant som hypertexthjälpmedel och Standard Generalized Markup Language, SGML.[1] Arbete av detta slag pågår på många håll. I Sverige har förlaget C E Fritzes nyligen aviserat att man kommer att släppa en "elektronisk lagbok" på kompaktskiva (CD-ROM), där alla gällande författningar i Svensk författningssamling finns lagrade och SGML-märkta för att möjliggöra sökningar och sammanställningar på varierande nivåer. I online söksystemet Rättsbanken hos DAFA Data AB förekommer hypertextfunktioner som enkelt kan förflytta läsaren från t.ex. en rättsfallstext till en författningstext.

En intressant tendens är att flera av de nya verktygen är inriktade på att ordna texter på grundnivån så att det blir möjligt att efter varierande behov strukturera texterna, kommentera dem, koordinera dem o.s.v. Man kan tala om ett växande intresse för "dynamisk ordning" och situationer där den som *konsumerar* en text samtidigt kan vara *skapare* av en text. Läsar- och författarrollerna glider över i varandra. Det är från den

---

[1] *Hypertext* är den generella benämningen på olika metoder som möjliggör för en läsare att skapa sin egen läslogik och lässekvens vid umgänget med texter. Någon kan t.ex. läsa ett EG-direktiv och när läsningen hunnit till ett visst stadgande välja att "hoppa" till ett textavsnitt i den svenska författning som realiserar direktivet i den svenska rättsordningen. *SGML* (som också är en internationell standard, ISO 8879) innebär ett enhetligt sätt att beskriva dokument, vilka i rättsligt sammanhang kan vara t.ex. en författning, ett rättsfallsreferat eller en kommenterande handbok. Det gäller dokumentstruktur, länkar mellan textenheter, tolkningsregler, bearbetningsregler. Se för en utförlig beskrivning Charles E Goldfarb, The SGML Handbook. Oxford: Clarendon Press 1990.

synpunkten värt att nämna ett textbehandlingsprogram benämnt Folio™, vilket i sin senaste version inte förutsätter att någon "huvudtext" eller "originaltext" överhuvudtaget lagras. Programmet arbetar enbart med inverterade filer och kan från dessa skapa och återskapa alla önskade ordningar, inklusive, om så önskas, "originaltexten".

4. Vad bör datorlingvistiken syssla med?

Allmänt

Det har varit min avsikt att presentationen av juridiken i allmänhet och rättsinformatiken i synnerhet implicit skall ha gett anledning till en rad funderingar kring vad datorlingvistiken i dessa sammanhang bör syssla med. Jag vill betona att jag ser det som angeläget forskarna inom datorlingvistiken själva finner uppgifterna intressanta. Egennytta skadar alls inte! Vad jag avslutningsvis skall göra är att närmare se på några konkreta situationer där juridiken bör välkomna medverkan från datorlingvistikens sida. Förhoppningsvis möjliggör den föregående diskussionen en rikare förståelse av denna genomgång.

Skapandet av rättsnormer

Textkontroller av olika slag är angelägna. Det är fråga både om att säkerställa *felfrihet* och att höja texters *kvalitet*. Till och med i officiella författningssamlingar kan man finna felstavningar, förvanskade ord och bortfallna stycken. Olika typer av formella textkontroller som stöd åt vanlig kontrollläsning är angelägna och ger utrymme för idéer från datorlingvistiken. När det gäller kvalitetshöjning vänds blicken mot sådant som terminologikontroll och kontroll av att texter logiskt hänger samman. De tidigaste datorinsatserna inom juridiken gällde sådant som att kartlägga användningen av speciella ord och fraser i författningstexter. Själv minns jag min överraskning när de första datorframställda ordlistorna till svenska förmögenhetsrättsliga lagar (s.k. keyword out of context listor) avslöjade att lagen om avbetalningsköp hade något att säga om adoptivbarn.

Ett ofta diskuterat tema handlar om *regelförenkling* – att skära ned författningsvolymen, att "strömlinjeforma" regleringar, att hålla samman regelverk och göra dem lättillgängliga. Det norska lovstrukturutvalget beskriver uppgiften så:

> "Den overordnede målsetting må være at lover og forskrifter er utformet på en slik måte at budskapet i disse når frem til brukerne... på en mest mulig presis og normativ måte, og med minst mulig omkostninger i form av tid og arbeidsinnsats både for forvaltningen og den enkelte. Språklig klarhet, god meningsmessig sammenheng i reglene og en regelsystematikk det er lett å finne frem i og forholde seg til, er de viktigste kravene som må stilles til regelverket."[1]

Jag finner det uppenbart att man här har att göra med kombinerade rättsliga, administrativa och språkliga överväganden men att det samtidigt inte är någon enkel uppgift att se hur datorlingvistikens arsenal av verktyg kan komma till bästa användning. Idéer och diskussion efterlyses! Tag till exempel denna: att utveckla beskrivningstyper som lämpar sig som "mellannivåer" vid konstruktion av rättsnormer. Det kan handla om gränssnitt som ger en överblick över någon viss reglering, som gör den mer lättillgänglig från någon viss synvinkel (t.ex. den skadelidandes) eller som grafiskt beskriver någon viss regelstruktur. Vilka intressen har datorlingvistiken i sådana uppgifter?

Datoranpassad formulering

Här har man uppmärksammat behov av lättprogrammerad författningstext eller, mer allmänt, utformning av texter så att de lämpar sig för automatiserade miljöer. Det handlar om sådant som termer och begrepp och om rättstexternas struktur och logik.

Som en randanmärkning vill jag nämna att det har skett en svängning i attityderna på detta område. Tidigare talade man gärna om automationsanpassning av den rättsliga regleringen, senare har det blivit minst lika angeläget att betona de krav som olika rättsliga regleringar ställer på informationsteknologin och olika användningar av denna: tekniken skall rätta sig efter lagens krav.

Författarverktyg

Hela batteriet av sådana verktyg behövs.I *juridikundervisningen* ligger tonvikten på språkgranskning. Denna är en tung del av lärarnas arbete – så mycket som nio tiondelar av granskningen av en inlämningsuppgift kan avse relativt enkla språkfel. Vi vill se goda datorprogram för formell textkontroll – gärna med fackspråkliga påbyggnader. I *praktiskt juridiskt arbete* handlar det om verktyg för att hantera struktur i stort hos texter (t.ex. avtal), för att kontrollera texter, för att stödja arbete med

[1]NOU 1992:32 (anförd ovan i not 13), sid. 15.

alternativa formuleringar av texter m.m. Det bör finnas goda möjligheter att röra sig mellan olika nivåer hos texter (jfr traditionella outliners) och olika beskrivningsformer (version 1, 2, 3, n, motpartens perspektiv, kronologiska kritierier, definitioner). Kort och gott ställer det juridiska författandet stora krav på sina utövare, vilket driver på jakten på verktyg för att effektivisera och höja kvaliteten på arbetet.

Spridningen och användningen av rättsnormer

Lexika, termlistor, tesaurusar och liknande både i datorlagrad form och som traditionella publikationer behövs i många sammanhang. Inte minst internationaliseringen och arbetet med flerspråkiga textdatabaser reser nya krav. Datorstödd och automatiserad översättning, terminologikontroll och terminologianpassning hör också hemma i detta sammanhang.

Informationssökningsstöd av olika slag har stort och uppenbart intresse. Vid Stockholms universitet har vi särskilt goda erfarenheter av samverkan med datorlingvistiken på detta område. Under åren som gått har det handlat om bl.a. morfemsegmentering för att förbättra sökordslistor och om "substantivjakt", där substantiv automatiskt excerperats ur författningstexter för att ge kompakta beskrivningar av dessa och underlag för olika fortsatta ansträngningar att hantera texterna. Ett sådant försök handlade om automatiserad kartläggning av samband mellan stadganden i författningar. Senast har professor Benny Brodda intresserat oss för möjligheterna att använda ett antal dokument, t.ex. ett knippe rättsfallsreferat, som startpunkt för en matematiskt baserad metod att hämta fram likartade texter.[1]

I informationssökningssammanhang är också alla möjliga typer av filter angelägna: för rangering av funna texter och textställen, för fokusering, för eliminiering, för att klargöra sammanhang o.s.v. Den stora och växande volymen hos de åtkomliga textmängderna (jfr ovan) gör sållförmågan hos datoriserade metoder till en angelägen egenskap. Något liknande kan sägas om de intressanta möjligheterna hos hypertext och besläktade verktyg. Lyckas man inte kombinera friheten att röra sig i alla tänkbara kunskapsdimensioner (från domslutet om skyddstillsyn till statistik i en psykologisk avhandling om återfallsbrottslingars farlighet o.s.v.) hamnar man mycket snart i hyperkaos. Allt hänger samman.

---

[1] Sammanfattningsvis är det fråga om att invertera den traditionella metoden vid fri textsökning. I stället för att gå från en given fråga (sökordsuppsättning) via en sökfunktion till en uppsättning dokumenttexter som svarar mot frågan, vänder Brodda på problemet och utgår från en given mängd dokument för att *söka den fråga* som utvidgar den ursprungliga dokumentmängden till en (ännu) större mängd relevanta dokument. Se närmare Benny Brodda, *Gimme more o'that. A Potential Function in Document Retrieval Systems?* I: *From Data Protection to Knowledge Machines. The Study of Law and Informatics*. Ed. P Seipel. Computer/Law Series 5. Deventer: Kluwer 1990.

De allt större datorläsbara textmängderna ökar alltså behoven av strukturering. Texter bör därför lagras i en grundform som är så rik som möjligt på information om textens egenskaper – det gäller ett helt egenskapsspektrum sträckande sig från strukturell information (om kapitel, paragraf, moment o.l.) till sådan som har med sakinnehållet att göra (t.ex. om tolkningsförslag och tolkningsalternativ, om begrepp och begreppsfamiljer och om ursprung och källa). Det är för sådana syften som den ovan nämnda SGML-märkningen tilldrar sig rättsligt intresse. Allt vad datorlingvistiken kan åstadkomma för att hjälpa oss med sållning, strukturering och ordnande är välkommet.

"Vanliga uppgifter"

Som en avrundning och avslutning vill jag inte underlåta att beröra några uppgifter som kanske inte är av så specifikt intresse för vare sig datorlingvistiken eller rättsinformatiken, men som alla hör hemma i skärningen mellan språk och informationsteknologi.

För det första: Hur närmar man informationsteknologins fikonspråk till fackspråken på alla de yrkesområden där tekniken skall användas? Sättet att beskriva tekniken och orden som används och inte används är minst sagt problematiska. Det kan handla om djupa språkklyftor som fördröjer förnuftig användning av tekniken och bäddar för dyrbara misstag.

För det andra: Vem ägnar handböckerna, "manualerna" uppmärksamhet? Redan att skriva en bruksanvisning om en cykelpump eller en shunt på en värmepanna kan vara svårt. När det kommer till handböcker om datorprogram och datornät verkar uppgiften nära nog övermänsklig. För några år sedan författades en doktorsavhandling på arkitekthögskolan om svårigheterna att städa i badrum och på toaletter. Det blev omdiskuterad som ett fall, där uppgiften inte ansågs värdig vetenskaplig forskning. Hur är det med manualerna? Är de under språkvetenskapens värdighet?

För det tredje: De stora datorlagrade textsamlingarna med rättstexter ger nu förutsättningar som tidigare inte funnits för studier av "juristsvenska", av ändrat juridiskt språkbruk, av olika delområdens juridiska språkbruk o.s.v. Också här bör finnas uppgifter för lingvistiken och i vissa fall kanske speciellt för datorlingvistiken. Juristernas digitala bord är dukat för envar som önskar ta för sig.

Domain Modeling and Knowledge Structures

Annie Stahél and Helle Wegener
København

Abstract

Natural language communication between the end user and knowledge base requires an interface with access to linguistic knowledge. Further support can be provided by a *domain model*, i.e. a module which, besides domain specific knowledge, contains common world knowledge and rules for the inference of implicit knowledge from the facts explicitly represented in the database. In our paper we present a concrete example of domain modeling. Our domain model is based on associative networks and frames. In our presentation we discuss the criteria applied, i.e. our choice of knowledge primitives and the establishment of knowledge structures by means of a network and the mapping of this network into frames.

1. Introduction

The background for what we want to present here today is the FAGFLADE project, a research project carried out at Department of Computational Linguistics at the Copenhagen Business School. FAGFLADE is short for Danish '**fag**sprogligf**lade**" which means "special purpose language interface".[1]

The aim of this project is to develop and test theories and methods relevant to the construction of text interpreters for texts written in special purpose language. A text interpreter is a program which transfers the information contained in a natural language text into semantic representations which may serve various purposes. It is not our intention to build a complete text interpreter, but we have taken the development of an interpreter to be an ideal goal which defines an overall project which gives rise to a number of interesting subprojects for the investigation of general theories and principles concerning interpreters, e.g. in the domain of syntactics, semantics, lexical and terminological databases and in the domain of knowledge representation.

One of the subprojects under FAGFLADE concentrates on the construction of a natural language interface which can take a natural language question to a knowledge system as input and return appropriate (natural language) answers to the end user.

---

[1]This paper is a slightly modified version of a paper presented at a FAGFLADE seminar in Copenhagen in March 1993.

The core of our knowledge system is a (fictitious) database VIRKBAS, which registers the relevant information about a firm, its employees, products and customers, etc. Apart from explicit information on staff, orders, complaints and the like, the database has a lot of implicit information concerning various relations between the registered entities.

In order to make this information accessible to users of the knowledge system it must be represented in a domain model which allows a representation that combines domain specific knowledge with an appropriate amount of world knowledge.

A prerequisite for the analysis of natural language questions is specific knowledge of the domain plus a certain amount of knowledge of the world referred to by the questions. An important function of the domain model is to serve as a filter that allows an acceptable user question such as 'Who are the colleagues of NN?' to be converted into a query in a formal database query language like SQL. The model must also be able to reject meaningless questions such as 'What is the salary of a TVset?'

2. The database

The specific knowledge of the domain is explicitly present in our (relational) company database VIRKBAS. The tables of the database have been structured on the basis of the Entity/Relationship diagram shown in figure 1.

The Entity/Relationship diagram shows the entity types of the domain. The database registers information about employees, customers, products, complaints etc. Each box in the diagram represents a type of entity. Each entity type is characterized by a number of attributes. Thus the entity employee, for instance, is characterized by attributes like crnumber (civil registration number), name, address and departmental attachment, among others.

The entity types of the diagram are related to each other: an employee, for instance, is employed in a department. Employees have salaries, positions, sell products, etc. Relationships as these are expressed by rhombs in the diagram. For practical reasons, the rhombs have no names in the diagram, as we are not going to focus on these relationships in the present context.

The degree of the various relations between entity types: one-to-one, one-to-many or many-to-many is important, however, since it determines the database structure. One department for instance, can have attached to it a

number of employees, but an employee can be attached to one department only.

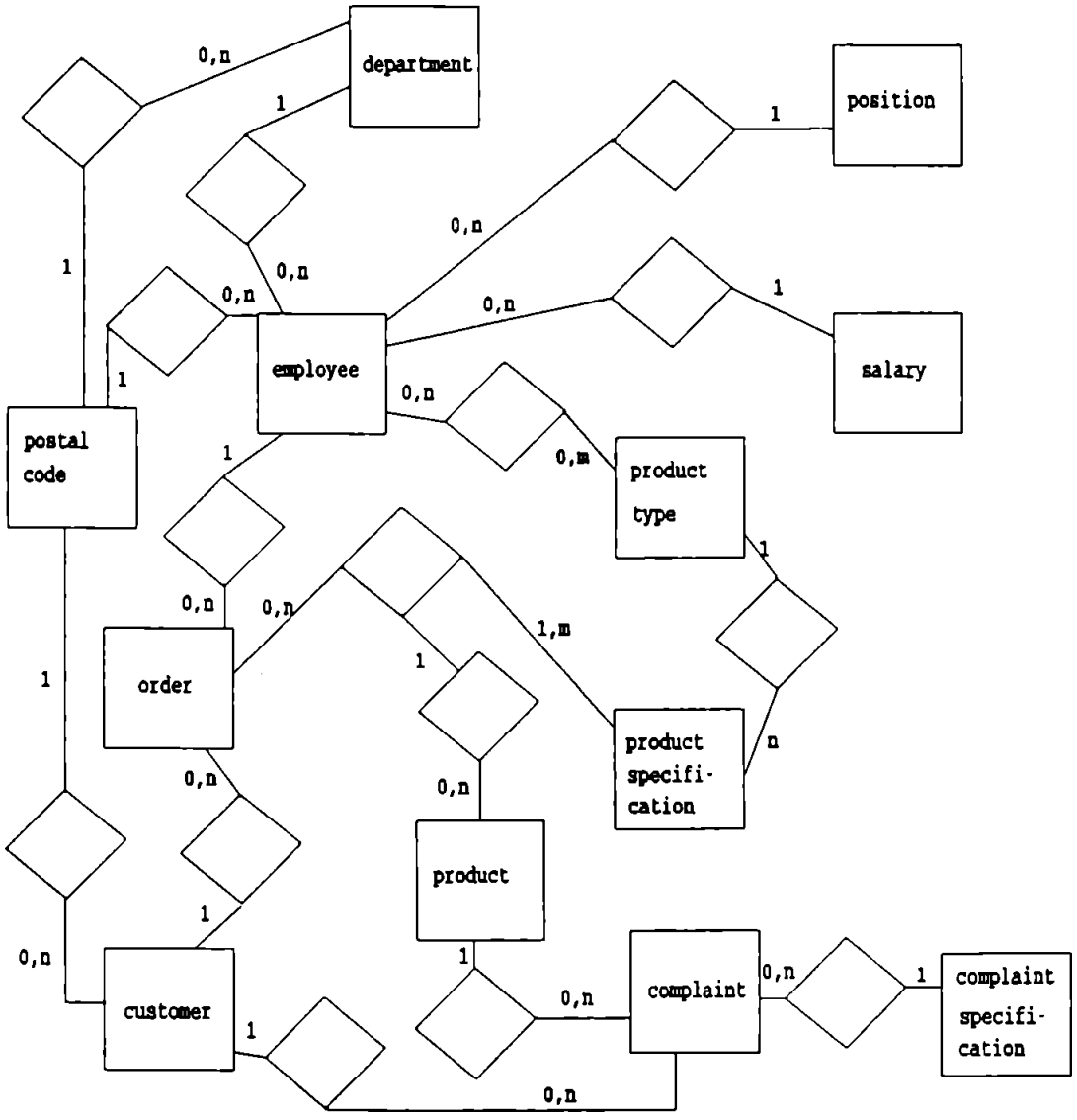


FIG 1: Entity/relationship diagram for the company

Each entity type is mapped into a separate table in the database which defines the properties of the given entity. A one-to-many relation between entity types requires that the entity characterized by the degree "1" is represented in the table of the entity type characterized by the degree "many" by a key. This key makes it possible to access all the information concerning related entities, i.e. related tables.

In the case of a many-to-many relation between 2 entity types, it is normally necessary to create a table to represent this relationship, and this table is then constituted by a key from each of the 2 entities. In the diagram such a relationship exists between "order" and "product specification".

Figure 2 shows the two entity types "department" and "employee" realized as tables in the database. Some of the attributes, which were not included in the diagram, can be seen in the tables where they appear as names of the columns.

dept:

NO	NME	STRT	PCODE	TELNO
1	Sales	Nørrebrogade 12	2200	31859511
2	Administration	Østerbrogade 75	2100	39279140

emp:

NO	FNME	SNME	..	..	..	SX	..	DEPTNO	..	POSTYPNO	WGCODE
1	Signe	Pedersen	..	..	..	f	..	2	..	1	8
5	Hanne	Osteen	..	..	..	f	..	1	..	3	5

FIG 2: Examples of tables of the database

In order to ensure the easiest possible retrieval of information from the database, two views concerning employees and sales activities of the company were created. A view is a virtual table created as a conglomerate of several base tables. Information retrieval from a view is uncomplicated, but virtual tables suffer from certain inadequacies. Updating the base tables via a view is not possible. Furthermore, any restructuring of the base tables would also demand a redefinition of the views. Finally, the meaning of natural language words is defined in terms of semantic predicates related directly to the base tables and not to the views. Consequently, we decided to do away with the views.

3. The semantic net

Apart from the explicit facts represented in the tables of the database, the domain model, as already mentioned, requires a representation of a certain amount of world knowledge and possibly additional expert information concerning the relations between the entities of the domain. The system must have access to the facts that managing directors as well

as area managers are both a kind of managers, that a department is a part of a firm, and that the managing director is the superior of all other types of employees.

Semantic nets have proved to be useful structures for total representations of individual units of information related to each other in such a way that departing from one unit of information it becomes possible to access information available in related parts of the total structure.

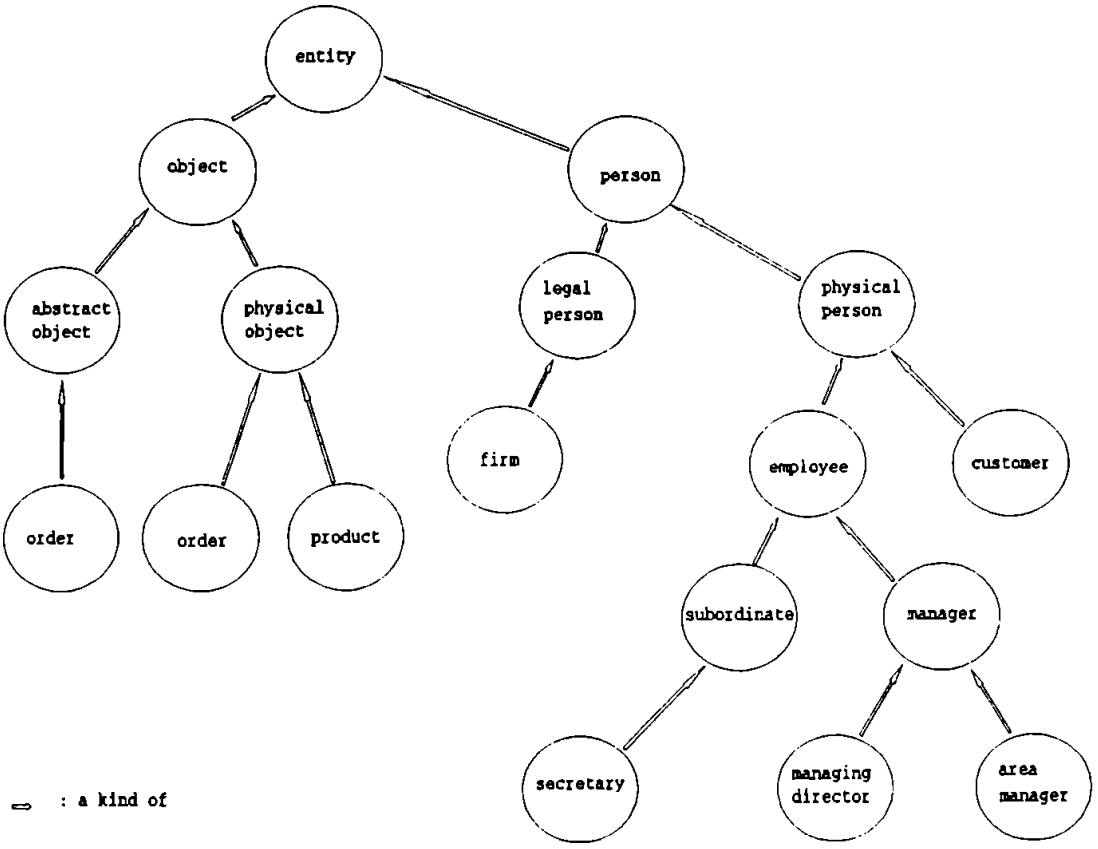


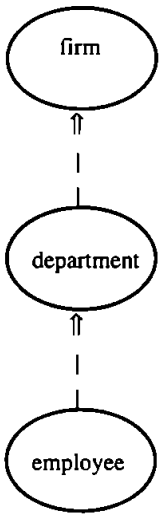
FIG 3: Generic relations

The semantic net consists of nodes representing concepts of the domain (most of which correspond to the entity types of the database), and links between the nodes that represent different types of relations between them. Two types of conceptual relations are established, one of which is the generic relation which can be seen in figure 3. Each daughter node represents a-kind-of the concept represented by the mother node.

The construction of the hierarchy is based upon the presence of characteristic features which distinguish the concepts. Thus PHYSICAL PERSON is distinguished from LEGAL PERSON by the feature crnumber.

The nodes LEGAL PERSON and PHYSICAL PERSON represent concepts which are introduced into the net in order to relate FIRM to PERSON to account for the fact that both firms and persons have legal capacity and share certain relations, as we shall see later.

The other type of conceptual relation is the part-whole relation that produces the hierarchy shown in figure 4. An employee is a part of a department, and a department is part of a firm.



– – ⇒ : a part of

FIG 4: Part–whole relations

It is possible to combine the two types of conceptual relations in one net. The semantic net in figure 5 combines our knowledge about the implicit generic and part-whole relations between concepts.

The advantage of combining different types of links in one net is that this makes it possible to represent role relations which exist between the entities of the domain even if the links between the entities involved are of different conceptual types. Furthermore, all the information on the domain concerning inventory of nodes and the various types of relations between them can be read directly out of the net.

Another part of the implicit information that we want to represent concerns other and more complex types of relations between concepts, i.e. the role relations that exist between the nodes of the net. Figure 6 below shows the representation of the role relations in the combined net.

Examples of role relations are relations such as: "be a superior to" or "be a colleague of", or a 3-place relation such as: "buy from" which holds between the nodes "customer", "product" and "firm". Another example is

the role relation "sell" between the concepts "firm" and "product" as ako "physical object". In this way we make implicit information explicit: in our domain firms can sell only physical objects such as radios and television sets. Consequently, inappropriate questions concerning a sale involving arguments other than firms and physical objects will be rejected.

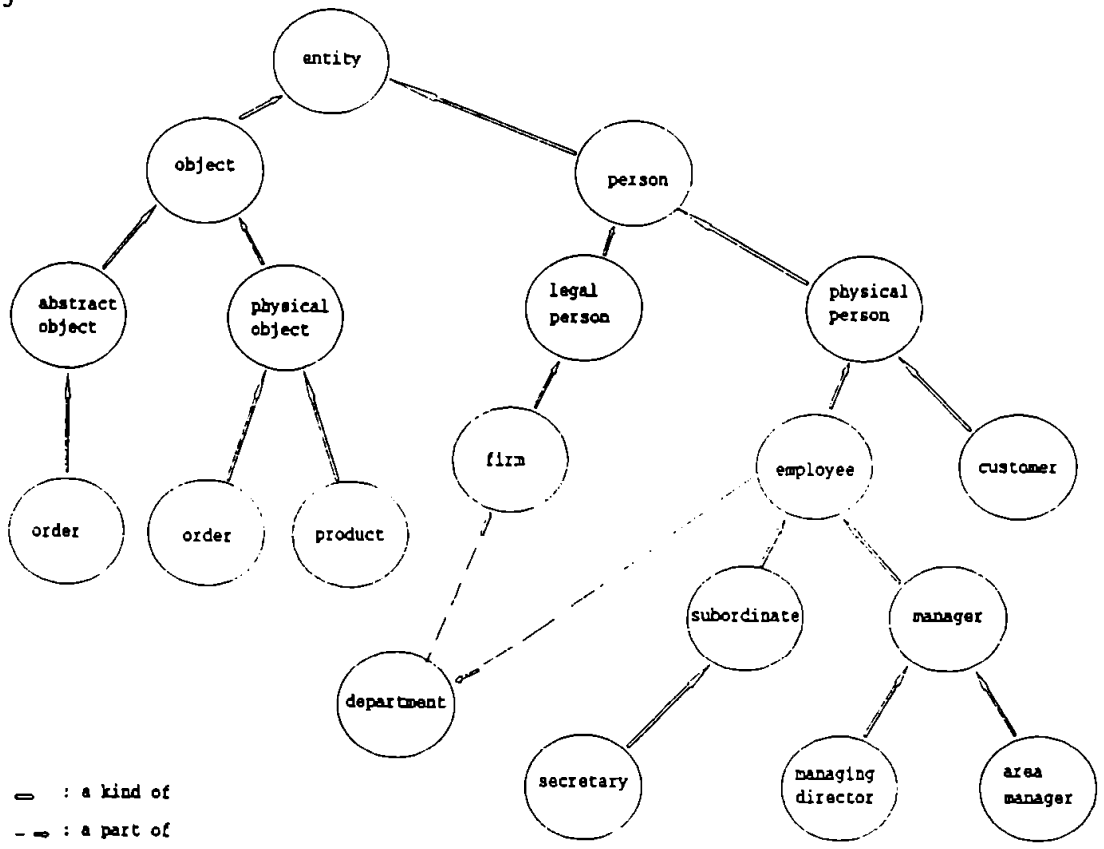


FIG 5: Generic and part-whole relations in one net.

The role relations are shown with double lines between nodes in the net. They are not directed. We consider an explicit marking of the direction redundant as the role relations are defined by the types of the concepts involved combined with the types of thematic roles associated with the concepts in a given relation. The thematic roles will be discussed in the description of the frames below.

The (identification and) choice of role relations is determined by our expectations of the questions that the end users will typically ask about this specific domain: What does the firm sell to customer NN? Who does the firm do business with? Who are the colleagues of NN?

When we introduce the role relations in this combined net the result, however, is a net which contains no less than three different types of links representing 2 types of conceptual relations, the akó and the apo links, and the role relation links. This presents certain problems concerning the

representation of role relations in the net, as the type of conceptual relation between two nodes determines the inheritance of role relations.

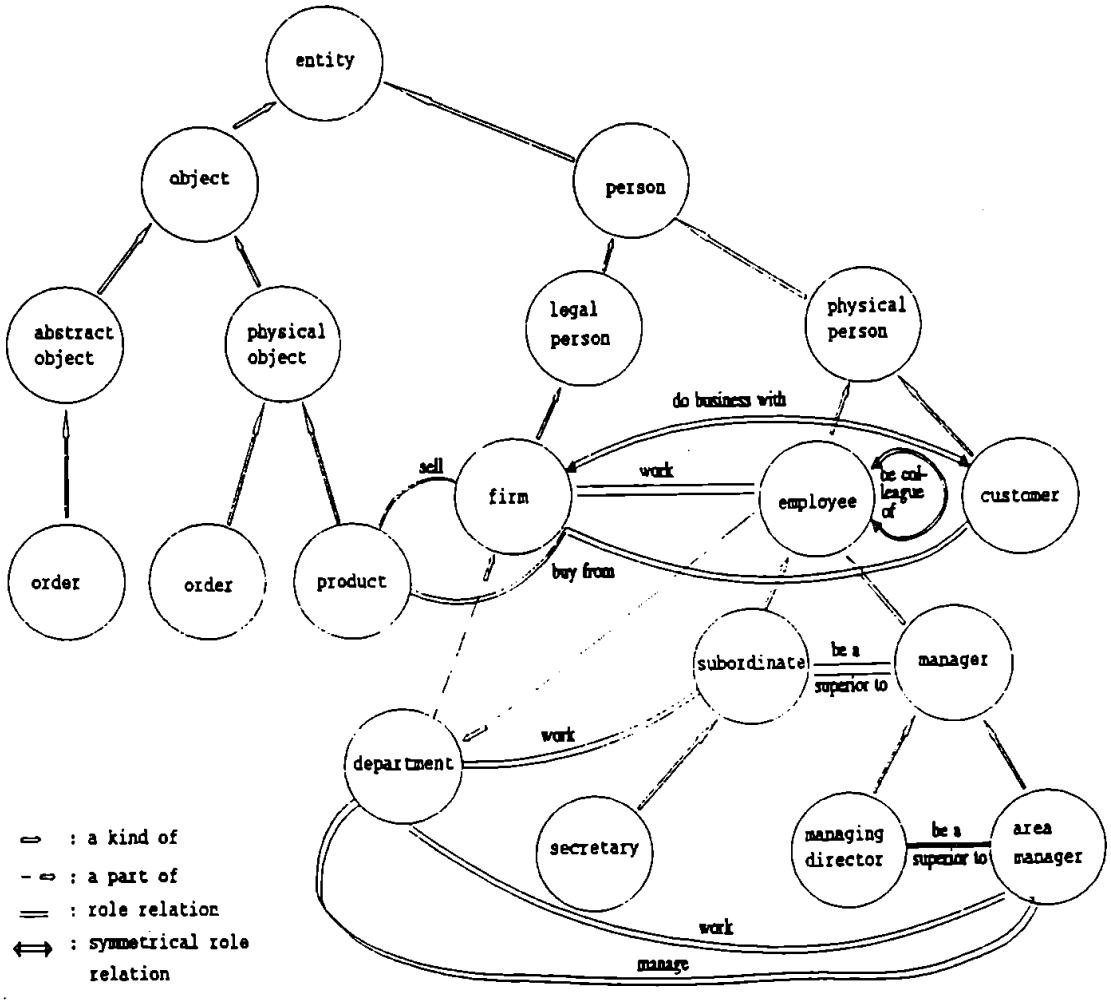


FIG 6: The combined net with role relations.

Let us look at an example: What is represented by establishing the role relation WORK between the nodes FIRM and EMPLOYEE? The role relation WORK is inherited by MANAGER from EMPLOYEE, as MANAGER stands in a generic relationship to this node, i.e. is a kind of EMPLOYEE, as well as by AREA MANAGER who, in his turn, is a kind of MANAGER. Managers as well as area managers work in a firm.

DEPARTMENT, however, which stands in a part-whole relation to FIRM, does not inherit any role relation from the mother node FIRM, since the daughter nodes in a partwhole relation do not inherit the characteristics of the mother node. This very appropriately reflects the fact that a managing director does not work in a department.

On the other hand, in order to represent that an AREA MANAGER works in a DEPARTMENT, this role relation has to be stated explicitly between these two nodes.

4. Frames

The semantic net models the relations between the concepts of the domain. The nodes of the net and the links between them are the knowledge primitives of the model. In order to be able to operate on the knowledge contained in the net and, ultimately in the database, we need a complete description of the units of information constituted by the nodes and the role relations.

For each node and role relation in the net the domain model contains a frame.

4.1 Structure and content of the frames

A frame is a data structure which represents the knowledge attached to each node or role relation, i.e. their definitorial and structural properties. All types of information are represented as feature specifications of the classical slot:filler structure. The feature values may be atomic values, e.g. the name of another frame or a specification of the datatype required for the value in question, or it may be a complex value consisting of another feature:value pair.

The basic structure of a frame is shown in figure 7.

```
[ frame   :   {framename}

  relations : (( conc : ( {type} : {frame name} ) )
                ( role : {frame name} )              )

                  rolestruct: ( { role type} : {frame name} )

  attributes: ( {attribute : datatype} )                 ]
```

FIG 7: The frame structure

The structure of the frame is based on 3 types of information: an identification of the frame, a relational description and attribute specifications.

The **frame name**, which identifies the frame, is the name of a node in the net or the name of a role relation, cf. the following 2 examples:

```
frame : EMPLOYEE
frame : WORK
```

The relational description, **relations**, may contain 2 types of relations, **conc:** the conceptual relations: ako or apo, and **role**, the role relations defined by a name for the specific type of relation, "work", "be colleague of", etc.

Role relations, which appear only in frames that describe concepts, are further defined by **rolestruct**, which contains a specification of the thematic role structure of the relation. The thematic structure is defined by a specification of role type, **role type**, and the name, **frame name**, of the frame that represents the value of a possible filler for the role type slot.

The frame for EMPLOYEE contains the following relational description:

```
relations : ( ( conc : ( ako : PHYSICAL PERSON ),
                       ( apo : DEPARTMENT      ) )
              ( role : ( WORK                  ),
                       ( BE COLLEAGUE OF       ) )
```

An employee is a kind of physical person and a part of a department and, an employee works *somewhere* and is a colleague of *somebody*. Identification of the relevant *somewhere* or *somebody* takes place via the frames for the respective role relations.

The first value specified for **role** above is the name of the frame for the role relation WORK, which contains the following specification of the thematic role structure of the relation:

```
rolestruct: ((  (actor    : EMPLOYEE   )
                (locus    : FIRM    ) ),
            ( (actor      : SUBORDINATE )
              (locus      : DEPARTMENT  ) ),
            ( (actor      : AREA MANAGER )
              (locus      : DEPARTMENT  ) ))
```

The relation WORK implies 2 participants associated with 2 types of thematic roles: actor and locus. In our domain the respective participants of a working relation can be: an employee and a firm, or a subordinate and a department, or an area manager and a department.

The last slot, **attributes**, contains the attributes of a node in the net, specified as one or more attribute names followed by a specification of a **data type**, i.e. the type of the data which can appear as the attribute value in question. Only frames which describe concepts contain attribute slots (so far).The frame for EMPLOYEE contains several attributes:

```
attributes : ( ( Position no  : INTEGER )
               ( Wage code    : INTEGER )
               ( Emp. date    : INTEGER )
               ( Dept. no     : INTEGER ) )
```

The attributes are drawn from tables in the database where they denote properties of the entities. The attributes correspond to the columns of the tables.

We have now described details of the frame structure and content. In the following we present examples of complete frames in order to show how these frames relate to each other in accordance with the semantic net in figure 6 above. The frames in examples (1) to (4) define conceptual relations. Example (5) defines a role relation.

(1)

```
[ frame   :  PERSON
  relations:     ( ( conc : ( ako  : ENTITY  ) )     ]
```

(2)

```
[ frame   :  LEGAL PERSON
  relations: ( ( conc : ( ako : PERSON    ) )

  attributes:             (( Name        :  a STRING      )
                           ( Address     :  s STRING      )
                           ( Telephone   :  an INTEGER    ) )]
```

(3)

```
[ frame   :  PHYSICAL PERSON
  relations : ( ( conc : ( ako : PERSON  ) )

  attributes :          ( ( FirstName  : a STRING     )
                          ( SurName    : a STRING     )
                          ( Address    : a STRING     )
                          ( Telephone  : an INTEGER   )
                          ( CR nr.     : an INTEGER   ) )]
```

(4)

```
[ frame   :  EMPLOYEE
  relations : ( ( conc : ( ako : PHYSICAL PERSON  )
                         ( apo : DEPARTMENT     ) )
                ( role : ( WORK          )
                         ( BE COLLEAGUE OF ) )

  attributes : (( Position code : an INTEGER    )
                ( Wage code     : an INTEGER    )
                ( Emp. date     : an INTEGER    )
                ( Dept. code    : an INTEGER    ) )     ]
```

(5)

```
[ frame   :  {framename}

  relations : ( ( conc : ( ako : STATE         ) )

  rolestruct: (((actor          : EMPLOYEE      )
                (locus          : FIRM          )),
               ((actor          : SUBORDINATE   )
                (locus          : DEPARTMENT    )),
               ((actor          : AREA MANAGER  )
                (locus          : DEPARTMENT    ))   )  ]
```

5. Conclusion

Our domain model represents a fragment of the world. It contains explicit knowledge about what we consider relevant entities and relations in the domain. It also contains implicit knowledge about the domain, i.e. the knowledge which can be defined as knowledge, either about general

logical relations between entities of the domain – or about more complex relations considered to be relevant world knowledge.

The logical basis for the representation of the explicit knowledge of the domain is established in the E/R diagram. The operational representation is established in the tables of the database.

The logical basis for the description of the implicit information is established by a semantic net that represents the entity types or concepts of our domain and different types of relations between them. The operational definition of the implicit information is stated in the frames.

The frames constitute the operational keys of the system. They combine reference to the explicit information in the database tables with the implicit relational knowledge represented in the semantic net.

Bibliography:

Börner, Stefan. 1987. *Datenbankunterstützung für wissensbasierte Systeme.* LILOG REPORT 10, IBM Deutschland GmbH, Stuttgart.

Cruse, D.A. 1986. *Lexical Semantics.* Cambridge University Press.

Date, C. J. 1990. *An Introduction to Database Systems.* 5th edition. AddisonWesley Publishing Company.

Woods, W. A. 1991. *Understanding Subsumption and Taxonomy: A Framework for Progress.* In Sowa (ed.): *Principles of Semantic Networks*, pp. 4595. Kaufmann.

Preferences and Linguistic Choices in the Multra Machine Translation System

Anna Sågvall Hein
Uppsala

Abstract

The work to be presented here concerns the ordering of alternatives in the Multra Machine Translation System. The Multra MT system is a fundamental part of the Multra prototype, modeling a translation work bench with user-controlled mixed mode of mechanical and human translation. The Multra system is based on transfer and unification. It includes three main modules, responsible for analysis, transfer, and generation, respectively. In addition, there is a separate preference module ordering the analysis alternatives before passing them on to the transfer component. Preferences are expressed by means of linguistic rules defined over feature structures. Alternative transfer rules are applied according to specificity; a specific rule takes precedence over a more general one. The specificity principle also governs the application of generation rules. The MT system as a whole, as well as its separate modules, can be tuned to present the best alternative only, or the complete set of alternatives in the preferred order.

1 Introduction

The work to be presented here was carried out in the project *Multilingual Support for Translation and Writing, Multra* (Sågvall Hein 1993a). It concerns the ordering of alternatives in the Multra Machine Translation system. The Multra MT system is a fundamental part of the Multra prototype, modeling a translation work bench with user-controlled mixed mode of mechanical and human translation. In its present version, Multra supports the translation of car maintenance manuals from Swedish to German and English.

The Multra system is based on transfer and unification. It includes three main modules, responsible for analysis (Sågvall Hein 1987 and in preparation), transfer (Beskow 1993a), and generation (Beskow 1993b). In addition, there is a separate preference module ordering the analysis alternatives before passing them on to the transfer component. Preferences are expressed by means of linguistic rules defined over feature structures. Alternative transfer rules are applied according to specificity; a specific rule takes precedence over a more general one. The specificity principle also governs the application of generation rules. The preference rules along with the specificity principle of the transfer and generation processes constitute the Multra preference machinery.

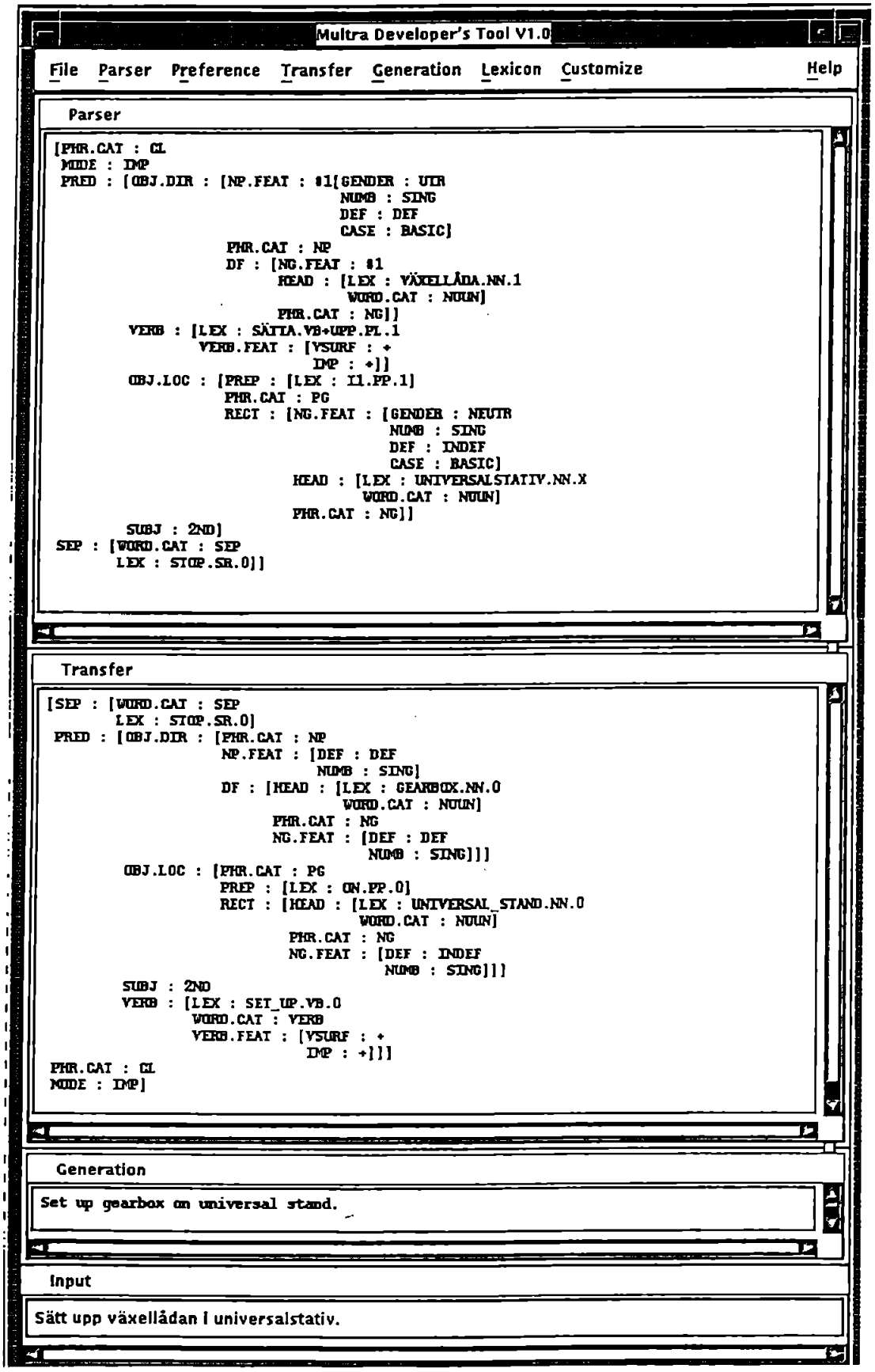


Figure 1: First-best translation of *Sätt upp växellådan i universalstativ.*

The MT system as a whole, as well as its separate modules, can be tuned to present the best alternative only, or the complete set of alternatives in the preferred order. For the design and testing of translation rules, a special environment, Multra Developer's Tool, MDT (Beskow 1992), has been developed, and we will start our presentation of the Multra MT system and preference machinery in this environment.

2 An example

Fig. 1 presents the first-best translation of the Swedish sentence *Sätt upp växellådan i universalstativ.* (see **Input** window) into English, *Set up gearbox on universal stand.* (see **Generation** window). The sentence is analyzed as an imperative clause consisting of a predication (a verb with its complements), and a separator (see **Parser** window). The predication is made up by the phrasal verb *sätta upp [set up]* (lexeme SÄTTA.VB+UPP.PL.1) and its (elliptic) subject, direct object, and locational object. Recursively and in parallel lexical and structural transfer rules apply to analysis structure yielding the English transfer structure displayed in the **Transfer** window.

The target transfer structure is (functionally) isomorphic to the source analysis structure, and the translation process may seem trivial. It does, however, include three kinds of phenomena that call for the preference machinery. They relate to the analysis phase, the transfer phase, and the generation phase, respectively, as will be demonstrated below.

The Swedish noun *universalstativ* (the head of the locational object) doesn't distinguish formally between its singular and plural forms. The intended reading in this example is singular, but a plural reading, eventhough rare, cannot be excluded in this type of contexts. Thus both alternatives have to be accepted but priority be given to the singular form. A preference rule (i) takes care of that.

(i)
```
PREFERENCE OBJ.LOC.SING.PLUR
        <* PRED OBJ.LOC RECT NG.FEAT NUMB> = SING
PRECEDES
        <* PRED OBJ.LOC RECT DF NG.FEAT NUMB> = PLUR
```

As is the case with most prepositions, there are several translations of the Swedish preposition *i*, even though it has been recognized as denoting location in space (not in time). Its default translation into English would be *in*, but when it collocates with *universal stand*, *on* is the correct expression. In other words, the transfer component must account for a default translation, as well as for a translation in context. We introduce the Multra transfer rule format (Beskow 1993a) by presenting the simple lexical rule accounting for the default translation of the preposition (ii).

(ii) LABEL
I1
SOURCE
<* LEX> = I1.PP.1
TARGET
<* LEX> = IN.PP.0
TRANSFER

Lexical transfer rules in Multra define translation relations between lexemes, or basic senses (Allén 1981). The rule in (ii) relates the (Swedish) source lexeme I1.PP.1[1] to the (English) target lexeme IN.PP.0. Analogous to (ii) is the lexical transfer rule UNIVERSALSTATIV presented in (iii).

(iii) LABEL
UNIVERSALSTATIV
SOURCE
<* LEX> = UNIVERSALSTATIV.NN.X
TARGET
<* LEX> = UNIVERSAL_STAND.NN.0
TRANSFER

The translation of *i* in context is handled by a transfer rule covering the preposition along with the noun that it governs (iv). The rule applies to a prepositional group, PG, consisting of the preposition I1.PP.1 and a nominal expression with UNIVERSALSTATIV.NN.X as its head. Further, the whole of the nominal expression governed by the preposition, its rection, is assigned to the variable ?RECT1. Corresponding to the source structure of (iv) the rule defines a target prepositional group introduced by the preposition ON.PP.0. Further a target language attribute, RECT, is defined with the variables ?RECT2 as its value. Finally, ?RECT2 will be bound to ?RECT1 via the TRANSFER relation; recursively and in parallel, transfer rules will be applied to ?RECT1, concluding with the application of the lexical rule UNIVERSALSTATIV (iii).

Both (ii) and (iv) are applicable to our example. However, (iv), or rather its source part, is more specific than that of (ii), and consequently, (iv) will be preferred. (Being more specific means specifying a greater number of identity relations, more specific identity relations, or a greater number of transfer relations, see further Beskow 1993b).

In the transfer process of the example, no shift (cf. Ingo 1990) of function, structure, category, or feature takes place. For instance, the

[1]A lexeme is represented by the basic form of its lemma, followed by a part of speech marker, and a lexeme number. The Swedish lexeme numbers accord with those given in Svensk Ordbok (1986). Lexemes outside the scope of Svensk Ordbok are assigned lexeme number X. If the basic forms of two lemmas coincide, numbers keep them apart, as in our preposition example. As for target lexemes, they are, so far, assigned a zero lexeme number.

Swedish direct a object in the definite form is transfered as such into English.[1] However, in accordance with the English model translation,[2] the direct object will appear in its indefinite form in the resulting translation, viz. *Set up gear box on universal stand.* Thus a shift of definiteness will take place in the generation phase, as will be explained below.

(iv)
```
LABEL
    I_UNIVERSALSTATIV
SOURCE
    <* PHR.CAT> = PG
    <* PREP LEX> = I1.PP.1
    <* RECT HEAD LEX> = UNIVERSALSTATIV.NN.X
    <* RECT> = ?RECT1
TARGET
    <* PHR.CAT> = PG
    <* PREP LEX> = ON.PP.0
    <* RECT> = ?RECT2
TRANSFER
    ?RECT1 <=> ?RECT2
```

The standard rule for generating the predication of an English imperative clause with a direct object and a locational object is presented in (v) below. The rule is formulated in a PATR like style (Beskow 1993a). It comprises three parts, i.e., a label, a sequence of constituents (variables) to be generated, and a number of identity equations, binding the variables to path expressions in the transfer structure and expressing constraints upon this structure.

(v)
```
LABEL PRED3a
X1 --> X2 X3 X4 :
<X1 PRED SUBJ> = 2ND
<X1 PRED VERB> = <X2>
<X1 PRED OBJ.DIR > = <X3>
<X1 PRED OBJ.LOC> = <X4>
```

In (v), X1 refers to an imperative predication of a transfer structure, and the first equation identifies it as such. (The value of the implied SUBJ attribute is set to 2ND in imperative clauses.) The value of the verb attribute will be assigned to X2, the value of the direct object attribute to X3 etc.

In (vi) we present a generation rule that implies a shift of definiteness. It generates a direct object in the indefinite form from a direct object in the

---

[1]Working in a multilingual translation environment, we aim at a transfer component as simple and general as possible, referring the target language specific features to the generation components, see also Sågvall Hein 1993b.

[2]From the English version of our experimental text, a maintenance manual for trucks from Saab-Scania.

definite form, picking up the (unquantified) description field (DF) of the transfered object (Sågvall Hein, in prep.). (vi) being more specific than (v) will be preferred.

If we tune the parser, the transfer component, and the generation component towards all alternatives, six English translations will be generated and presented in the preferred order:

(vi) LABEL PRED3b
X1 --> X2 X3 X4 :
<X1 PRED SUBJ> = 2ND
<X1 PRED VERB> = <X2>
<X1 PRED OBJ.DIR PHR.CAT> = NP
<X1 PRED OBJ.DIR DF > = <X3>
<X1 PRED OBJ.LOC> = <X4>

Set up gear box on universal stand.
Set up the gear box on universal stand.
Set up gear box in universal stand.
Set up the gear box in universal stand.
Set up gear box in universal stands.
Set up the gear box in universal stands.

3 Preferences among source ambiguities

In Multra, the number of analysis alternatives is restricted as far as possible by maximal use of valency information; there is, for instance, no general PP-attachment rule. All PPs, modifying NPs, are attached by valency-rules. For instance *Ta bort luckan för kraftuttagshuset. [Remove the cover of the power take-off housing.]* gets only one analysis, according to which *för kraftuttagshuset [of the power take-off housing]* expresses appurtenance[1] in relation to *luckan [the cover].* Another example: The verb *sätta på* in *Sätt på lyftverktyget 87 792 på växlingsförarhusets plats. [Attach lifting tool 87 792 in position of gear selector housing cover.]* requires a locational object; thus, there will be no interpretation of *på växlingsförarhusets plats [in position of gear selector housing cover]* as a sentence adverbial. There are, however, cases, where the interpretation of a postposed PP as an adverbial cannot be excluded, and in those cases, a preference rule will give priority to the valency bound interpretation.

Number ambiguity is a common phenomenon in the Swedish source text, and even though singular is to be preferred in most cases, there are cases when the plural reading is the intended one. An example of such a case is the headline of a table, see for instance (vii).

[1] According to a suggestion made by Jarmila Panevova.

(vii) Specialverktyg [Special tools]

Fig [Fig]	*Nummer* [No]	*Benämning* [Name]
1	*79 046*	*Dorn* [Drift]
...		

The headline of a table is analyzed as a special kind of sentence fragment, a table name, and priority to the plural reading in such cases is given by a preference rule of the type presented in (i) above. Quite a number of contexts have to be specified in preference rules in order to account for number ambiguity.

Still another type of ambiguity to be handled by preference rules is due to elliptic coordination, see e.g. (viii).

(viii) *Ta bort de fyra skruvarna för locket och kopplingshävarmen.*
[Remove the four bolts of the cover and the clutch lever.]
a) Ta bort de fyra skruvarna för locket och (för) kopplingshävarmen.
b) Ta bort de fyra skruvarna för locket och (ta bort) kopplingshävarmen.

According to a) *lock* and *kopplingshävarm* are coordinated, according to b) the two imperative clauses. a) is to be preferred, and a preference rule may express view. By means of the examples presented above, we hope to have demonstrated that the machinery of preference rules is well apt for ordering structural ambiguities; slightly extended, it can apply to lexical ambiguities as well. The strategy of referring the ordering of source language ambiguities to a separate module contributes to the portability of an MT system; the generality of a standard parser can be maintained, whereas the preference module is tuned to the needs of the individual user and his specific types of text. Defining the preference rules will be an important part of the customization process.

4 Ordering lexical translation alternatives

As an example of a translation ambiguity, we present the set of German equivalents of the Swedish verb *ta bort [remove]* that we found in our experimental text. In all, there are 10 different translations, i.e., *entfernen*, *abnehmen*, *herausnehmen*, *abbauen*, *herausschrauben*, *demontieren*, *ausbauen*, *lösen*, *herausheben*, and *herunternehmen*. The verb is transitive, and, evidently, the distribution of the target language alternatives is determined by its direct object, for instance, *Schrauben herausschrauben; Kupplungsservomechanismus, Kupplungshebel* and *Mutter abbauen*; *Ausrücklager*, *Deckel*, *Dichtung*, *Distanzstück*, *Kupplungshebel*, *O-Ring*, *Dichtring*, *Sicherungsring*, *Traghülse*, *Passscheibe*, *Planeten-getriebebeteil*, *Schaltstangengehäuse*, *Schmierleitung*, *Schraube*, *Sicherungsschraube*, *Traghülse* and *Ölpumpe entfernen*. *abnehmen* takes the same set of objects as *entfernen*, and, in addition to

that, *Kupplungsgehäuse*. These two verbs have the widest use, and hence the most neutral meaning. In all, there are 107 occurrences of *ta bort*, and 57 elliptic uses. *entfernen* covers 90 (58 + 32) cases and *abnehmen* 32 (17 + 15). *entfernen*, being more frequently used than *abnehmen*, will be considered to have the most general meaning, and hence be chosen as the default translation of the verb. Its definition (*entfernen: wegbringen, beseitigen; dafür sorgen dass jmd., etw. nicht mehr da ist*) in Duden (1989) gives further support to this decision. The default translation will be expressed by a simple lexical transfer rule (cf. ii). *abnehmen*, on the other hand, appearing as a more or less absolute synonym of *entfernen*, will be neglected and the remaining translation alternatives be given in context (cf. iv). Due to the specificity criterion, priority will be given to the contextual translations. To sum up, distribution, frequency, and definition provide the general basis for determining default translations in Multra. There is only one default translation for each translation ambiguity, and remaining alternatives are presented to the system by means of phrasal (contextual) transfer rules.

5 Ordering generation alternatives

In **2** we presented the format of the generation rules and the application of the specificity principle to generation by means of an English example, i.e., the generation of a direct object in the indefinite form to be preferred to the definite form. Here we will give one more example of the specificity principle, demonstrating its application to the generation of ellipsis in coordinated clauses in German. The Swedish sentence *Ta bort kopplingsservomekanismen och yttre kopplingshävarmen. [Remove clutch servo mechanism and outer clutch lever.]* is analyzed as a coordinated clause with an elliptic expression of the verb in the second clause, i.e., *Ta bort kopplingsservomekanismen och (ta bort) yttre kopplingshävarmen.* The verb in the first clause is marked '+ surface', and the second one '- surface'. Corresponding to the two possible translations of the verb *ta bort*, a default translation and a translation in context (cf. **4**) four German transfer structures will be presented, based on *abbauen* (the preferred translation) and/or *entfernen*. If the same verb is used in both clauses, an elliptic expression in the first German clause (cf. the Swedish ellipsis in the second clause) must be considered. This can be arranged by means of a generation rule such as the one presented in (x). (x) being more specific than (xi), the default rule for generating coordinated clauses, will be preferred.

(x) %Coordinated clauses; two clauses with a conjunction: same verb

```
LABEL CL.COORD1
X1 ---> X2 X3 X4 X5 X6 :
    <X1 FIRST PRED VERB LEX> = <X1 SECOND PRED VERB LEX>
    <X1 PHR.CAT> = CL
    <X1 FIRST PHR.CAT> = <X1 SECOND PHR.CAT>
    <X1 FIRST MODE> = <X1 SECOND MODE>
    <X1 FIRST PRED SUBJ> = <X1 SECOND PRED SUBJ>
    <X1 FIRST PRED OBJ.DIR DF> = <X2>
    <X2 NG.FEAT CASE> = ACC
    <X1 CONJ> = <X3>
    <X1 SECOND PRED OBJ.DIR DF> = <X4>
    <X4 NG.FEAT CASE> = ACC
    <X1 SECOND PRED VERB> = <X5>
    <X1 SECOND SEP> = <X6>
```

(xi) %Coordinated clauses; two clauses with a conjunction; same or different verbs

```
LABEL CL.COORD2
  X1 ---> X2 X3 X4 :
    <X1 PHR.CAT> = CL
    <X1 FIRST> = <X2>
    <X1 SECOND> = <X4>
    <X1 CONJ> = <X3>
```

If both the transfer and the generation components are tuned for all alternatives, the following translations are generated and presented in the order of appearance below:

Kupplungsservomechanismus und äusseren Kupplungshebel abbauen.
Kupplungsservomechanismus abbauen und äusseren Kupplungshebel abbauen.
Kupplungsservomechanismus entfernen und äusseren Kupplungshebel abbauen.
Kupplungsservomechanismus abbauen und äusseren Kupplungshebel entfernen.
Kupplungsservomechanismus und äusseren Kupplungshebel entfernen.
Kupplungsservomechanismus entfernen und äusseren Kupplungshebel entfernen.

The first alternative corresponds to the model translation. Whether the order between the remaining alternatives is the best one can be discussed. The one presented, however, is the one that is generated when preferences (in terms of rule specificity) are adequatly formulated within each module, but no integration takes place between them. Integrating rule application control between the three modules of the MT system is a major undertaking. It should be motivated only if empirical data supporting a more sophisticated ordering of translation alternatives can be presented. One of the aims of the evaluation of the Multra prototype on site (Saab-Scania AB, Scania Trucks & Buses) is to examine the feasibility of such an effort.

References

Allén, Sture. 1981. *The lemma-lexeme model of the Swedish lexical database.* In B. Rieger (ed.) *Lexical semantics.* Bochum.

Beskow, Björn. 1992. *TransferTool on UNIX: An Introduction.* Dept. of Linguistics. Uppsala University.

Beskow, Björn. 1993a. *Generation in the Multra system.* Dept. of Linguistics. Uppsala University.

Beskow, Björn. 1993b. *Unification Based Transfer.* RUUL 24. Dept. of Linguistics. Uppsala University.

Duden. Deutsches Universalwörterbuch. 1989. Mannheim.

Ingo, Rune. 1990. *Från källspråk till målspråk. Introduktion i översättningsvetenskap [From source language to target language. Introduction to translation theory.]* Studentlitteratur. Lund.

Sågvall Hein, Anna. 1987. *Parsing by means of Uppsala Chart Processor UCP).* In: Bolc, L. (ed.) *Natural Language Parsing Systems.* pp. 203–266. Springer Verlag.

Sågvall Hein, Anna. 1993a, *Multilingual Support for Translation and Writing. MULTRA. Final Research Report. HSFR/NUTEK Language Technology Research Program.* Dept. of Linguistics. Uppsala University.

Sågvall Hein, A., 1993b, *On the Translation of Nominal Expressions in a Unification-Based Multilingual Setting.* In: Hajicova, E. (ed.) *Proceedings of the Conference Functional Description of Language.* Prague.

Sågvall Hein, Anna. In print. *Preference Mechanisms of the Multra Machine Translation System.* In Hall Partee, B. & P. Sgall (eds.) *Meaning and Discourse. Festschrift for Eva Hajicova.* J. Benjamins Publishing Co. Amsterdam/Philadelphia.

Sågvall Hein, Anna. In preparation. *A Computational Grammar of Swedish.*

Svensk Ordbok [A Dictionary of Swedish]. 1986. Stockholm.

Constituency and Semantic Interpretation

Torben Thrane
København

Abstract

The main point of this paper is to present an argument against phrase structure analysis as providing an efficient basis for an automated system that has language *understanding* as its primary goal. There are ususally several constraints on constituency analysis of the X-bar variety, the fundamental rule of which is $X^n \rightarrow Spec X^n\ X^{n-1}\ Comp^n$. Four such rule systems and their constraints are presented, and it is shown that only if one or more semantic constraints are taken into account can the number of potential tree structures be kept at a manageable level and result in 'correct' constituency analyses. But this appeal to semantics, it is argued, is ill advised as a means towards understanding, for it is only meant to secure a uniform *description of sentences*. A more viable appeal would be one according to which structural meaning is exploited for the purposes of constructing and revising models of *situations described by sentences*.

Context and Aim

The study of language is guided by a number of fundamental questions, among them the following:

1 a What constitutes knowledge of a language?
b How does such knowledge develop?
c How is such knowledge put to use?

I will be concerned here with certain aspects of the first and second of these problems (Chomsky 1981,32).

It has always been Chomsky's ultimate aim to answer 1b, and it has always been Chomsky's belief that an answer to 1b presupposes an answer to 1a that can be given in terms of an independent, autonomous description of language structure.

It is my ultimate aim to answer that part of 1c which is concerned with how we *understand* language, and it is my belief that the *purpose* of any investigation determines the format, methods, and principles to be adopted. It is, furthermore, my claim (cf. Thrane 1992a,b; 1993, fc) that computational linguistics in general has accepted Chomsky's belief, no matter what its purpose has been – and that this has prevented serious progress in the study of computational *understanding* of NL.

Chomsky's belief is the foundation of what might be called the *descriptive* paradigm in linguistics – cf. Chomsky (1981,33):

> [W]e can say that a grammar constructed by a linguist is 'descriptively adequate' if it gives a correct account of the system of rules that is mentally represented, that is, if it correctly characterizes the rules and representations of the internally-represented grammar.

What I shall be specifically concerned with here is a central feature of that paradigm, the relationship between constituency rules and semantic interpretation. And although various forms of semantic motivation play a role in the choice of such rules, my conclusion will be that it is the wrong kind of semantic motivation when the purpose of the investigation is language *understanding* rather than language *description*.

PS-rules and constraints

There are two interpretations of any system of PS-rules:

2 a as an autonomous formalization of the knowledge of syntactic structure

b as a set of instructions for tree-building

There are at least four types of constraint on PS-rules:

3 a *assumptions* about the nature of PS-rules
[e.g. that terminals have already been exhaustively classified; that every constituent belongs to a category; that constituency is defined by movability, substitution and deletion; etc.]

b *graph-theoretic restrictions* on the formulation/application of PS-rules
[e.g. they must not lead to crossing branches; single-mother condition, etc.]

c *guidelines* for the formulation/application of PS-rules
[e.g. number of BAR-levels; type of recursiveness, etc.]

d *motivations* for the choice of PS-rules

Only 3d is my concern here, so I'll be a bit more specific about these. Two kinds of motivations for rule systems can be identified:

Data-oriented

4 a A rule system is chosen because it reveals structural differences between sentences S1 and S2 that correlate with *perceived differences* of meaning between S1 and S2.

b A rule system is chosen because it reveals structural properties of a sentence S that will play a role in *determining the meaning of S.*

Theory-oriented

c A rule system R is preferred over another R' because R is more constrained, consistent, and/or general than R'.

Only the data-oriented motivations are my concern here.

To keep the presentation at a manageable level, I shall confine myself to a discussion of PS-rules for the analysis of NP. (5) gives some data that should be handled by such rules. Even though the data are Danish and the rule systems to be discussed are for English, this shouldn't affect the general points being made.

5 a alle de mange andre drenge
all the many other boys
b de mange andre drenge
c mange andre drenge
d andre drenge
e *alle mange andre drenge
f alle andre drenge
g alle drenge
h drenge

The four rule systems to be mentioned are rivals within the Chomsky-tradition, to some extent reflecting its historical development. They are all post X-bar and therefore couched in X-bar terminology, even though one of them is not explicitly presented in such terms by its authors. They all assume a transformational component.

I explicitly mention only those constraints that are unique to the rule-system in question. All of them share such X-bar defining constraints as

- Designated Head
- Introduction of at most one lexical item per rule
- A lexical item introduced by a rule is the Head of the Phrase under analysis
- Allowance for cross-generalization

Four proposed rule systems

System 1 (Jackendoff 1977)

(a) N''' → (N'''|Art''') - N''
(b) N'' → (N'''|Q''') - (A''')* - N' - ...

Constraints

- Uniform Three-Level Hypothesis
- An NP specifier may contain at most one demonstrative, one quantifier, and one numeral. [Jackendoff's (semantic) *Specifier Constraint*]
- Specifiers are not strictly subcategorized for

Problems

- Presupposes both syntactic and semantic subcategorization of specifiers (and lexicon), otherwise ...
- ... it will generate just about *anything*

System 2 (Stuurman 1985; simplified wrt category vs. function distinction)

(a) X' → (Spec) {X|X'} ... [where X = {N|Art|Q} in our context]

Constraints

- Single Projection-Type Hypothesis
- Specifiers are constituents (they have a Head)
- At most one specifier per projection
- Requires a level of 'q-interpretation' (a non-PS, semantic process)

Problems

- Overgeneration: will generate 5e

System 3 (Wexler & Culicover's (1980) rules to generalize Bartsch's (1973) semantic constraints on NPs (inferred – but they assume X-bar theory))

(a) N'''_ (D) N''
(b) N'' _ (Q) N'
(c) N' _ (A) N

Constraints

- Base order generation significant
- Operator – operand organization for semantic interpretation

Problems

- Under-generation: will not generate 5a, and only 5b – f if *andre* is classified as A.

System 4 (Haegemann's (1991) NP-rules and 'Metarules')

(i) (a) N'' → Spec; N'
(b) N'* → N'; XP
(c) N' → N; XP

(ii) (a) X'' → Spec; X'
(b) X'* → X'; YP
(c) X' → X; YP

Constraints

- (i) is just a category-specific instantiation of (ii)
- Requires a representational (semantic) level of Logical Form

Problems

- Undergeneration: will not generate *any* of 5.

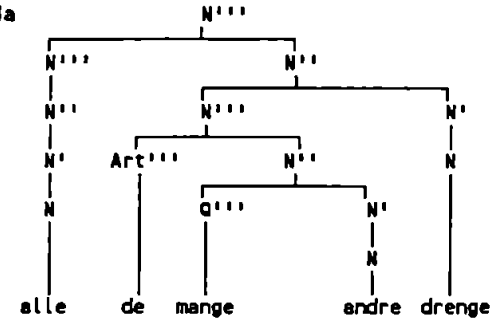
If we look at these four rule systems under interpretation 2a, they are clearly designed to answer questions 1a or b. Jackendoff's and Stuurman's rules are meant to provide partial answers to 1a, while Wexler & Culicover's and Haegeman's are designed to answer 1b. The members of each pair then differ among themselves. Jackendoff's and Wexler & Culicover's rules are data-oriented, whereas Stuurman's and Haegemann's are theory-oriented. There is nothing to choose between them, however, as far as the understanding vs. description dichotomy goes. They are all descriptive.

Computable Representations

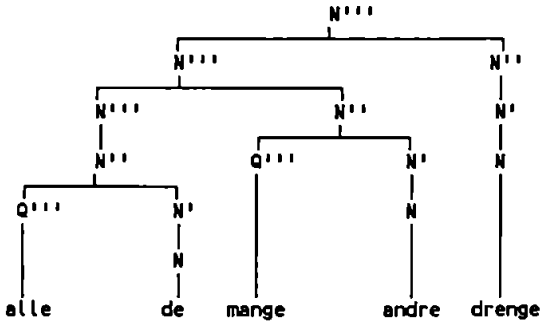
Under interpretation 2b of a rule-system and its associated constraints, a parser is an implementation of a computational process which feeds on information provided by grammar rules and constraints, and then converts one representation – in the form of a NL sentence – into another

representation – in the form of a tree. In this sense, trees are *computable representations*. The following are samples of trees computed from the rule systems and constraints we have been looking at.

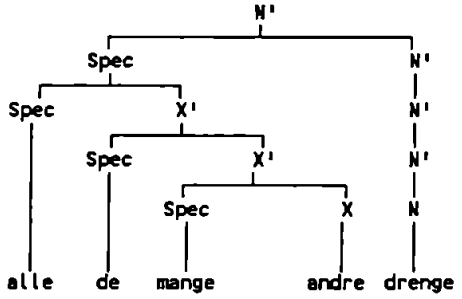
1a



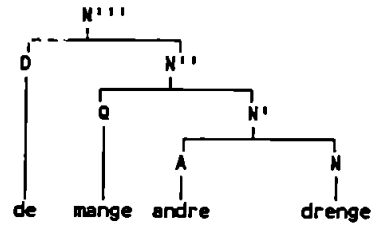
1b



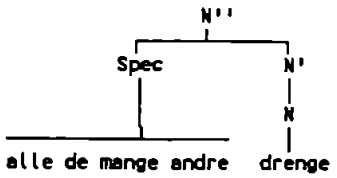
2



3



4



Now, if we horse around with various combinations of constraints and rule-systems, we find there are numerous theoretically possible tree-structures for 5a. If we relax all constraints except that every constituent must have a Head in connection with Jackendoff's rules, we get 312 different structures. If we add one – that *drenge* is the Designated Head – we reduce the number to 79. These statistics are fairly uninteresting. But what is interesting is that *only appeal to some semantically based constraint or motivation will produce the sort of configuration that is seriously considered in works on Phrase Structure.*

Despite this, we cannot assume that semantic motivations by *itself* will lead to the postulation of particular rule-systems. Consider the first semantic motivation (4a) in relation to 6:

6 a drengen købte en is
the boy bought an icecream
b en dreng købte en is
a boy bought an icecream

There is a perceived difference of meaning between 6a and b, which is the same in English as in Danish. None of the rule systems we've looked at would be prepared to propose different syntactic structures for 6a and b. So, a perceived difference in meaning is in itself neither a sufficient nor a necessary condition for proposing different syntactic structures.

Nevertheless, this seems to be precisely what we need to account for language *understanding*: to be able to say that perceived differences in *grammatical* meaning correlate with differences in computable representations – only that these representations are of a different sort from the tree-structures that we have been concerned with so far.

The difference can be explained with reference to the illustration of the relations between language, 'mind' and reality in Figure 1:

There are apparently three computable representations in this diagram:

- the tree is a representation of the syntactic structure of the sentence *it's a box* - assumed to be created on the basis of syntactic knowledge

- the house is a representation of a real house – assumed to be created on the basis of information provided by visual perception

These two are similar in being representations of the phenomena that gave rise to them. They are, in my terms, created on the basis of descriptive information, and they have *inclination of fit* towards a target which is identical to their source. They are *source-inclined.*

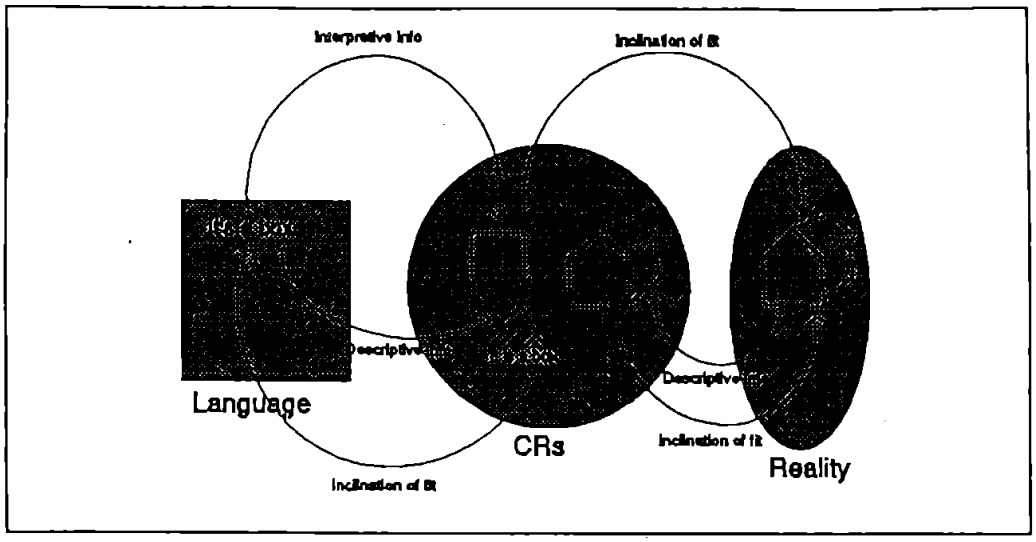


FIG 1 : The effect of descriptive and interpretive information, and *Inclination of Fit*

What is the box a representation of? It is standardly argued, I think, that the box is a representation of the meaning of the sentence *it's a box*. This argument is based on the assumption that lexical items and sentences *contain* meaning, and that this meaning can be independently *represented*. However, nothing so far has proved this assumption either useful or necessary for the purposes of language *understanding*. It is a purely descriptive view. For the purpose of language understanding it is much more fruitful to adopt the view that linguistic items have *semantic effects*. And that semantic effects have consequences for the creation and manipulation of *computable structures*. So,

- the box is not a representation *of* anything, but rather a computable structure with representational *potential*, created on the basis of information made explicit by the meaning of the sentence *it's a box*.

It is different from the other two in not being a *representation* of its source. It is similar to the others in being a structure with *inclination of fit*. I call it *target-inclined*, for it has inclination of fit towards a target which is different from its source. If it *has* a target, then it *becomes* a representation. It is created on the basis of *interpretive* information.

In general, the information that language carries in virtue of meaning is interpretive.

Semantic effect

So, for the purposes of language understanding, linguistic items do not *contain* meaning, they have *semantic effects*. Replacing the notion of semantic content by the notion of semantic effect need not force us to abandon the key principle of (formal) semantics, the principle of compositionality. We can reformulate it as the

Principle of uniformity of semantic effect

> Whatever semantic effect an expression has in one composite expression, it has the same semantic effect in another composite expression.

Pursuit of this principle has some interesting consequences. Firstly, the explanation of specificness and genericness in English, for example, cannot be upheld in its usual form, which in fact assigns two different semantic effects to the articles. Secondly, lexical (or descriptive) meaning is not subject to the principle. The assignment of a certain semantic effect to *bank*, for example, concerns its status as a noun or a verb, not its status as a homonym. The property of having a certain semantic effect is a matter of grammatical, or structural, meaning. It thus makes sense to inquire into, for example, the semantic effects of NP as a structural entity.

Semantic effects of NP

NP contains information that enables us to

• *individuate* entities	semantic effect of D
• *enumerate* entities	semantic effect of Q
• *classify* entities	semantic effect of N
• *assign properties* to entities	semantic effect of A
• *compare* entities	semantic effect of A
• *identify* entities	semantic effect of NP

In accord with Devlin (1991,20f;25), individuation presupposes a basic cognitive capacity to discriminate. Enumeration is a matter of recursive individuation. Classification is a function of individuation and our general cognitive capacity to categorize entities – ie. to realize that two distinct entities may be the 'same' in some respect. Property assignment is a function of individuation and our general cognitive capacity to localize entities – ie. to realize that the same entity may be in different places at different times. Subclassification and comparison are matters of recursive classification and property assignment, respectively. Finally,

identification is a function of either classification or property assignment or both. Figure 2 illustrates these principles.

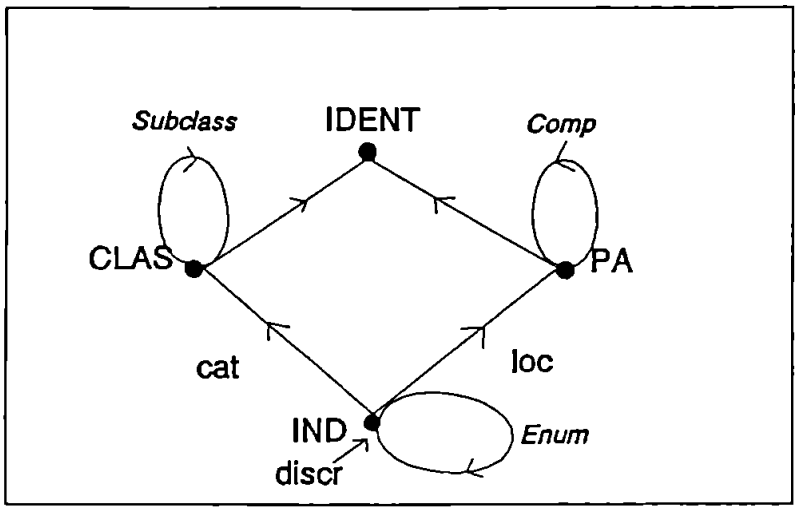


FIG 2 : The semantic effects of NP

This kind of semantic motivation is utterly deplorable for *descriptive* purposes. Yet for *functional* purposes it has two advantages:

- we can give a principled subclassification of (Danish) specifiers
- we can give a general layout of the organization of Danish NP in which the question of hierarchical structure is relegated to secondary importance – perhaps to be accounted for by lexical dependency rules – in deference to the question of linear order, which is far more important for language understanding.

Exhaustive **Selective**

UQ	D	EQ	AltD	A	N
alle	**de**	**mange**	**andre**	**store**	**drenge**
al	disse	få	øvrige		
alt	denne	ene	yderligere		
hele	dette	eneste	næste		
begge	den	to	første		
	det	tre	sidste		
	min-	nogen	anden		
	din-	noget	tredje		
	...	nogle	...		
	's	ingen			
		intet			
	hver	en	**større**		
	enhver	et			
	ethvert				
	hvilken	**største**			
	hvilket				
	hvilke				
	hvaffor				

Notice that some of the otherwise distinct effects are neutralized in some cases. The interrogatives and distributives (*hv-*) neutralize the quantifier - determiner effect. They are just exhaustive, in the sense of instructing the listener to take everything in the universe of discourse which meets the conditions posed by whatever lexical material follows in the NP into account.

Conclusion

I was asked after delivering the present paper what it had to do with computational linguistics. Granted, if the term 'computational linguistics' is reserved for the automatic manipulation of strings in various ways – not a lot. But if it is taken as a term for those varied branches of study that converge on the common goal of "produc[ing] a comprehensive, computational theory of language understanding and production that is well-defined and linguistically motivated" (Allen 1987,2), then – quite a lot. Among the consequences for computational linguistics of the position defended above the following are of especial interest:

- *Rethinking of the nature of 'rules'.* PS-rules may be an efficient and elegant means of capturing the structural properties of sentences. Yet if what we are interested in is not primarily structural properties, but the effect of structural information on computable structures, then they may not be efficient. Perhaps production rules, embellished with instructions for actions, would be a better choice. Cf. Thrane (fc) and Dinsmore (1991).

- *Rejection of correspondence theory as the basis of semantics.* Whether a sentence is true or not is a question of whether the computable structure it gives rise to has inclination of fit towards a factual situation or not. This question is clearly of secondary importance to the primary question of how computable structures are created and maintained in the first place. If the information needed for these procedures emerges from various aspects of NL meaning, then equally clearly these aspects of meaning must take analytic precedence over other semantic matters.

- *Parsing vs. model construction.* Parsing as currently practised is an inherently descriptive endeavour. The product of a successful parse is a set of source-inclined trees that reveal structural properties of NL sentences. However, parsing is a complex procedure which subsumes recognition of input and production of output, and there is nothing to prevent us from writing a parser that will yield a different, target-inclined kind of output structure. Nothing, that is, except the problems of identifying and formalizing the features that constitute the 'situatedness' of natural language. This would entail, among other

things, taking a procedural view of the meaning of specifiers, instead of just recording it and using it for grammaticality checks, as is usually done. Consider in this connection the following remark by Bolter (1984, 125) [my italics]:

> When humans speak to their robots or electronic brains, they do so in something approximating English, often omitting articles and other small words to suggest the computer's preference for reducing language to the bare bones of logic.

This is just utter nonsense in the present context. The implicit belief that 'the bare bones of logic' are embedded in lexical meaning has nothing to recommend it, even under standard assumptions about quantification in natural language and logic. Under present assumptions, withholding from 'our robots and electronic brains' the information provided by 'articles and other small words' is tantamount to preventing them from even beginning to understand what we are talking to them about.

References

Allen, James. 1987. *Natural Language Understanding*. Benjamin, Menlo Park.

Bartsch, Renate. 1973. *The Semantics and Syntax of Number an Numbers*. In Kimball, John P. (ed) *Syntax and Semantics*, Vol. 2. Academic Press, New York. Pp. 51–94.

Bolter, David. 1984. *Turing's Man*. Penguin, Harmondsworth.

Chomsky, Noam. 1970. *Remarks on nominalization*. In *Studies on Semantics in Generative Grammar*. The Hague. 1972.

Chomsky, Noam. 1981. *Principles and Parameters in Syntactic Theory*. In Hornstein & Lightfoot (eds.) *Explanation in Linguistics*. Longman, London. Pp. 32–75.

Devlin, Keith. 1991. *Logic and information*. CUP, Cambridge.

Dinsmore, John. 1991. *Partitioned Representations*. STUDIES IN COGNITIVE SYSTEMS, Vol. 8. Kluwer, Dordrecht.

Haegeman, Liliane. 1991. *Introduction to Government and Binding Theory*. Blackwell, Oxford.

Jackendoff, Ray S. 1977. *X-Bar Syntax*. MIT, Cambr. Mass.

Stuurman, Frits. 1985. *Phrase Structure Theory in Generative Grammar*. Foris, Dordrecht.

Thrane, Torben. 1992a. *The fallacy of descriptivism*. In Hansen, S.L. & F. Sørensen (eds.) *Issues in Semantic Representation*. Samfundslitteratur, Copenhagen.

Thrane, Torben. 1992b. *Dynamic Text Comprehension*. In Jansen, Steen et al. (eds.) *Computational Approaches to Text Understanding*. Museum Tusculanum, Copenhagen. Pp.173–90.

Thrane, Torben. 1993. *The Computations of Text Comprehension*. In Ahrenberg, Lars (ed.) *Papers from the 3rd NOTEX Conference on Text Comprehension in Man and Machines*. Linköping 1993. Pp. 173–82.

Thrane, Torben. forthcoming. *NP-structure and computation*. To appear in Herslund, M. (ed.) COPENHAGEN STUDIES IN LANGUAGE, Vol. 16. Samfundslitteratur, Copenhagen.

Wexler, K. and P. Culicover. 1980. *Formal Principles of Language Acquisition*. MIT, Cambr. Mass.

Machine Translation Strategies: A Comparison of F-Structure Transfer and Semantically Based Interlingua

Martha Thunes
Bergen

Abstract

Two machine translation (MT) systems which respectively utilize the transfer and interlingua strategies will be presented and compared, emphasizing design principles. Feature structures and unification-based grammar are common denominators for the two MT systems; in particular, both make use of Lexical-Functional Grammar (LFG). In the transfer system, Machine Translation Toolkit, developed by Executive Communication Systems, of Provo, Utah, transfer is based on LFG f-structure representations. In the interlingua system, PONS, constructed by Helge Dyvik, Department of Linguistics and Phonetics, University of Bergen, situation schemata representing the semantics of the source language text are employed as interlingua descriptions.

Introduction

The background for this paper is a study of these two MT systems where they are tested on English-to-Norwegian translation of technical text. The aim of the project is to find out to what extent the two different strategies, which have been employed in the systems, are able to maintain translational equivalence when put to the task of translating the same set of sentences. Since both applications are development environments for machine translation, and not ready made systems, the investigation will focus on potential for improvement and extendability, given the principles on which system design is based.

The notion of 'translational equivalence' denotes the relation that holds between source and target language expressions which are accepted as valid translations of each other. Translational equivalence is not an equivalence relation in formal terms: it is often the case that when translating between two given languages, translating a particular target expression back into the source language does not yield the original source expression as the optimal result.

The main difference between the strategies of transfer and interlingua can be described as follows: In transfer-based MT systems the translation process typically consists of three steps: analysis, transfer and generation. Analysis produces a source language dependent representation of input text. During transfer this is transformed into a target language dependent

representation which is the basis for target text generation. In principle, language pair specific information is employed only during transfer. In an interlingua system source sentence analysis yields a representation of the input string which is, ideally, language neutral, or at least neutral between source and target language. Because it is language neutral it is referred to as an 'interlingua' representation. Target text generation can be based directly on the interlingua representation.

The MT systems presented here both draw on the framework of **Lexical Functional Grammar**, cf. Bresnan (1982). This is a generative, non-transformational, unification-based grammar formalism. Linguistic expressions are assigned two levels of syntactic representation (see fig. 1): constituent structure, or c-structure, describes hierarchical and linear ordering of syntactic constituents. C-structures are derived by phrase structure rules. In addition to c-structure, there is a functional structure, or f-structure, where grammatical functions are represented. Nodes in a c-structure are annotated with functional equations. Functional equations, together with functional information associated with lexical entries, relate c- and f-structure to each other. The relation between c- and f-structure is one of co-description rather than derivation: Partial descriptions of an f-structure become associated with c-structure nodes. The f-structure is not derived by performing operations on the c-structure.

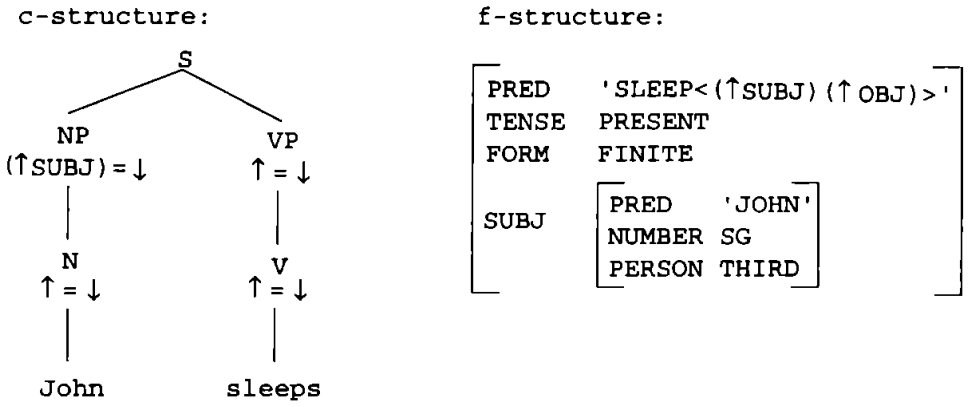


FIG 1 : A basic LFG representation of the sentence *John sleeps.*

A transfer system

Machine Translation Toolkit is a transfer-based MT system. Its grammars are designed in accordance with the LFG formalism. Lexical entries and grammar rules are coded as feature structures, or directed acyclic graphs (dags). A feature structure is a set of pairs of attributes and values. The f-structure representation of *John sleeps* in fig. 1 is an example of a feature structure. A linguistic representation language, LECS, has been developed

for the purpose of coding Toolkit language descriptions as feature structures. The structures that are built during the translation process are also represented as dags, coded in LECS. Information contained in the linguistic data base of the Toolkit system is mainly declarative, but there are also procedural elements in the linguistic descriptions. Firstly, monolingual lexical entries contain calls to structure-building operations that are employed during analysis and generation. Secondly, the bilingual transfer component consists of transfer entries, which contain translations as well as transfer rules. Transfer rules specify procedures, or dag-modifying functions, for transforming source sentence representations into corresponding target sentence representations. (1) is a sample transfer entry, written in LECS. In (1) the transfer rule named STD-TEN-P calls a function that substitutes the source language value of the attribute PFORM ('preposition, word form') with the value specified for PFORM in the corresponding target lexical entry.

(1) bilingual transfer entry mapping English *from* onto Norwegian *fra*:

```
en_from :: [ WORT  {  [ TECH  # GENERAL #
                FORM  "fra" ]  }
        \ STD-TEN-P ]
```

In the Toolkit system the analysis stage of the translation process outputs an f-structure representation of the source sentence, as illustrated in FIG. 2.

```
CAT     SBAR
FS      PRED        # SUBJ #
        FORM        burn
        VOICE       PASSIVE
        TENSE       PRES
        SUBJ        FORM        gas
                    PERSON      THIRD
                    NUMBER      SG
                    DEFINITE    PLUS
                    SPFORM      the
        ADJUNCTS    C 1         FORM        flare
                                PCASE       # SORC #
                                PFORM       from
                                QUANTIFIER  C 1   FORM      a
                                                  NUMBER    SG
                                                  DEFINITE  MINUS
```

FIG 2 : Toolkit, simplified source f-structure: *The gas is burned from a flare.*

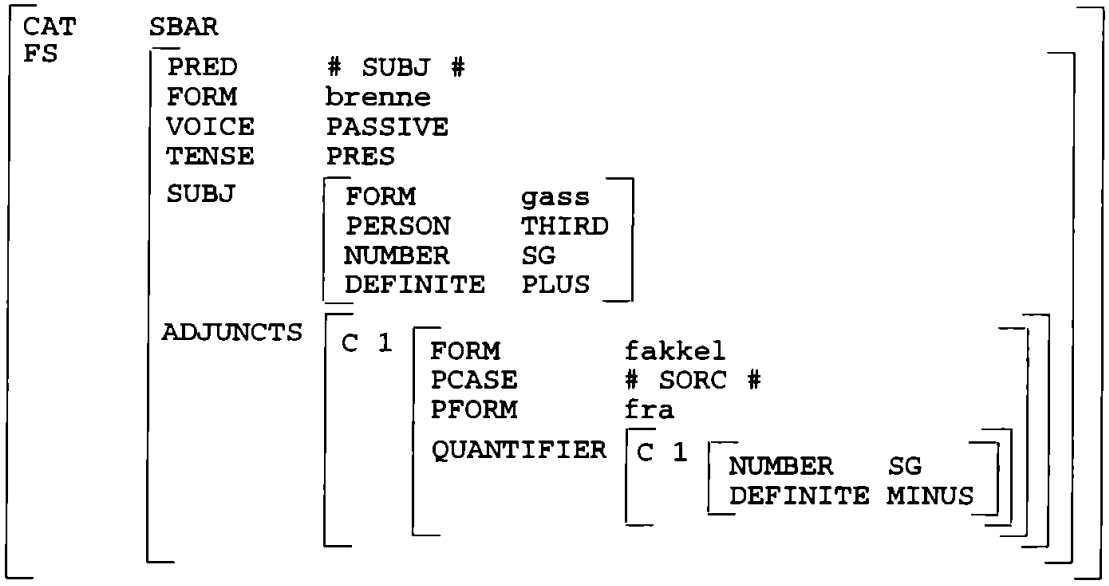


FIG 3 : Toolkit, simplified transfer dag:
The gas is burned from a flare. –> Gassen brennes fra en fakkel.

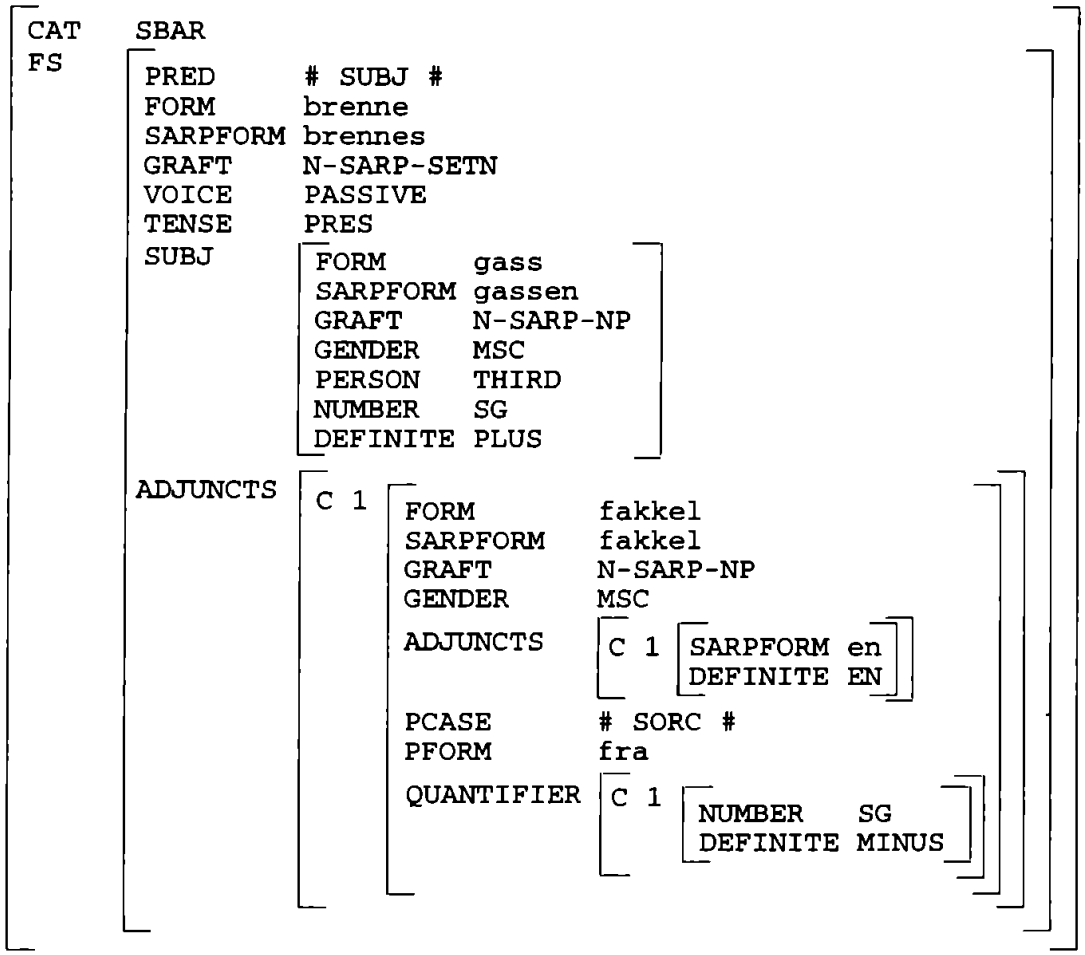


FIG 4 : Toolkit, simplified f-structure representation of target sentence:
Gassen brennes fra en fakkel.

In this particular source dag base forms of the words in the input sentence are given as values of the attributes FORM, SPFORM and PFORM. These values are pointers to a set of transfer entries (en_burn, en_flare, en_gas, en_from, en_a, en_the) which are processed during transfer. As a result transfer rules are executed, modifying the source dag into a transfer dag (fig. 3). In the transfer dag transfer rules have substituted English word forms with corresponding Norwegian forms. Also, transfer rules have deleted certain attribute-value pairs containing source language information which should not be carried over to generation. The target word forms in the transfer dag point to target lexical entries (nW_brenne, nW_gass, nW_fakkel, nW_fra). The information contained in these entries is added to the transfer dag, creating a target f-structure (fig. 4). The target dag contains inflected word forms which have been computed by applying morphological rules referred to in the target lexical entries. Lexical entries also point to syntactic rules, which build constituents. Syntactic constituent order is determined by functional ordering rules, which project grammatical functions onto syntactic constituents. Such rules are introduced either by monolingual lexical entries or transfer entries, and they apply only during generation. In the target dag they are referred to by the values of the attribute GRAFT.

An interlingua system

The PONS system is an experimental interlingua system for automatic translation of unrestricted text. 'PONS' is in Norwegian an acronym for "Partiell Oversettelse mellom Nærstående Språk" (Partial Translation between Closely Related Languages).

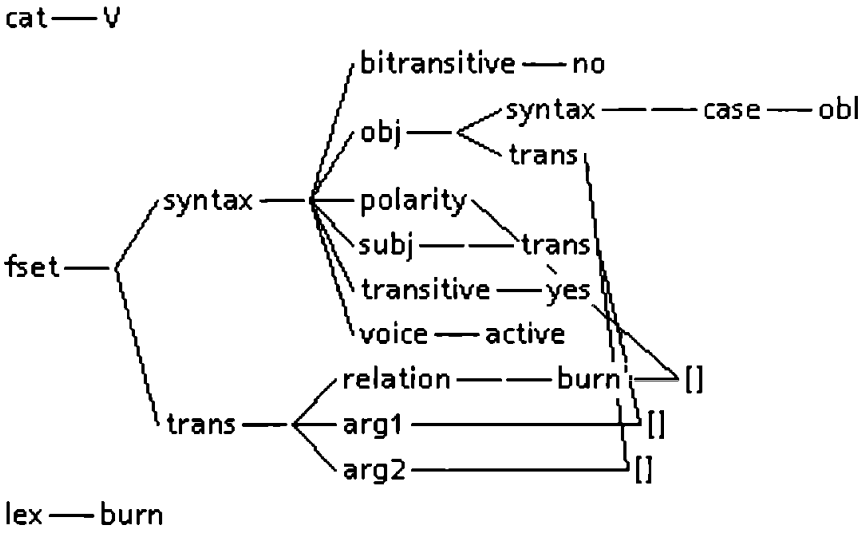


FIG 5 : PONS, simplified feature structure representing the word stem *burn*.

Translation is based on semantic analysis; however, a central principle is to exploit structural similarities between languages in cases where information about the syntactic structure of the source sentence can be used directly in target sentence generation. As a consequence of this, the PONS system has three different modes of operation: they vary with respect to the level of analysis at which translation is done. Interlingual translation is carried out only in the mode where translation is based on semantic representation. Linguistic descriptions in PONS are implemented in an extended version of D-PATR (Karttunen 1986). All grammatical and semantic information is coded as feature structures or directed graphs. The feature structure in fig. 5 is a graph representation of a sample lexical entry. All linguistic information in PONS is declarative; there are no procedures contained in the data base.

Before starting the translation process, different kinds of pointers are established between rules and word stems in source and target grammars. This is done automatically by a routine built into the system. The pointers describe a set of correspondences between representations of linguistic units in the two languages. These correspondences are exploited in cases where structural similarities between source and target language allow translation to be based on syntactic representation. The input sentence must be parsed before mode of translation can be chosen. Parsing yields one or more constituent trees. Attached to the topmost node in the tree is a feature structure representing the whole sentence; an example is given in fig. 6. Substructures of this structure are associated with individual nodes in the parse tree. A feature structure in PONS has essentially two components: *syntax* contains syntactic information, whereas *trans* is a semantic representation. Links between syntactic functions and semantic roles are expressed by giving shared values to specific attributes of the two substructures. E.g., *trans* of the syntactic subject is unified with *arg2* of the semantic relation *burn'*.

The parse tree also contains pointers to corresponding rules and word stems in the target grammar. The complexity of translation is automatically determined by the kinds of pointers that are contained in the parse tree. **Mode 1** performs word-for-word translation. It is necessary that the source and target stems express the same semantic relations and that the target pointers at each node show that source and target sentences are identical in syntactic structure. During translation terminal nodes in the parse tree are substituted with corresponding target word stems (fig. 7a). Inflected word forms must be found which are compatible with the feature structures associated with terminal nodes. However, if there are any word order differences between source and target expression, mode 1 will be insufficient, and **mode 2** may be employed. Mode 2 exploits correspondences between syntactic rules in source and target grammar. Differences in constituent order are allowed, but it is required that there is direct corres-

pondence between sense-carrying words (such as noun, verb, adjective) in source and target string. Fig. 7b) illustrates mode 2: During translation that subpart of the parse tree which represents the rule NP –> POSS N' is substituted with a subtree representing the target rule NP –> N' POSS.

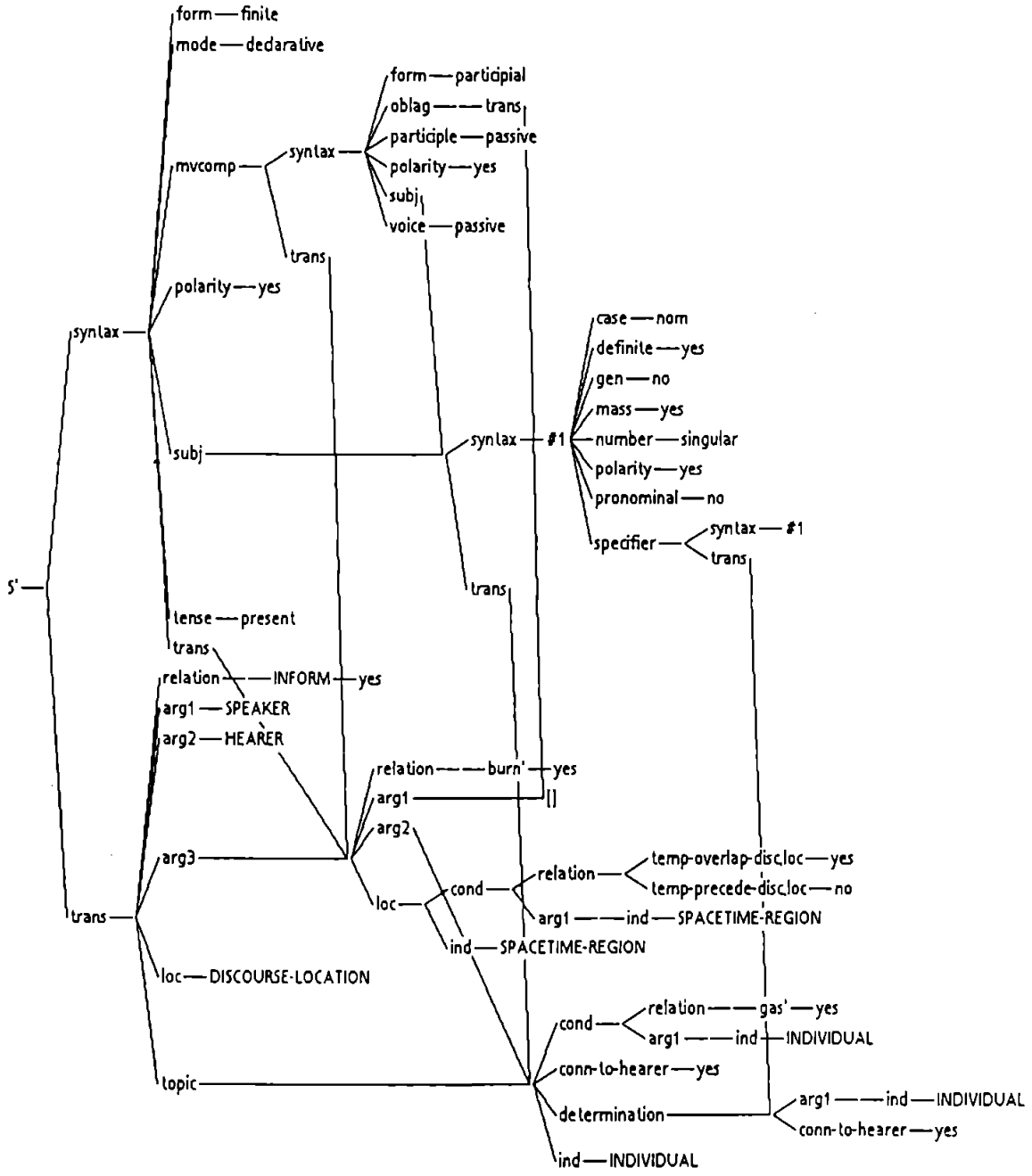


FIG 6 : PONS, simplified feature structure representing *The gas is burned.*

Next, terminal nodes are substituted with target word stems, and inflected word forms are found. **Mode 3** is used in all instances where 1 and 2 are insufficent. In mode 3 interlingual translation is carried out: the semantic representation of the source text functions as an interlingua expression. This representation is contained in the *trans*-part of the feature structure.

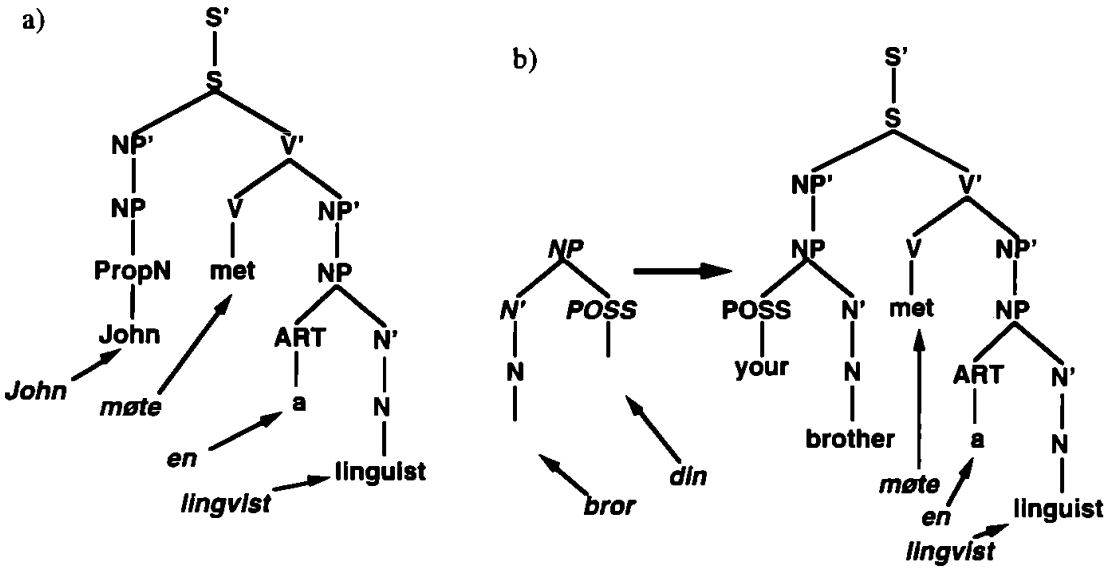


FIG 7 : a) PONS, mode 1: word-for-word correspondence:
John met a linguist. –> John møtte en lingvist.
b) PONS, mode 2: rule-to-rule correspondence:
Your brother met a linguist. –> Broren din møtte en lingvist.

The *trans*-structure is a situation schema: the notion of a situation schema has its origin in Situation Semantics (Barwise and Perry 1983, Fenstad et al. 1987) where situation schemata are used to represent the semantic relations contained in linguistic expressions. A situation schema consists of a set of attributes and values, where attributes designate types of roles in a fact and values refer to role fillers. A situation schema representing a sentence contains not only the propositional content, or the described situation, of that sentence. It contains also grammaticalized information about the utterance situation. To achieve translational equivalence the situation schema must include the information that is necessary to construct a target sentence that will express the same propositional content and have the same pragmatic function as the source sentence. To generate a target sentence from a situation schema the system must extract from the target grammar word stems and rules which express the semantic relations contained in the situation schema. Next, the full feature structures associated with these rules and stems are unified into the situation schema, extending this to a feature structure containing both syntactic and semantic information. To determine word order the syntactic rules of the target grammar are processed to build the constituent trees which are compatible with the feature structure.

The systems compared

In situation schemata in PONS linguistic meaning expressed by the source sentence is coded in attribute-value pairs neutral between source and target language. Translation via situation schema is based on the idea that two expressions from two different languages are translational equivalents if they are represented by the same situation schema. The situation schemata in PONS are declarative descriptions stating which expressions of source and target language that, at least according to the system, are translational equivalents. A situation schema is a representation neutral between analysis and synthesis, and also neutral with regard to direction of translation. Thus, the relation that holds between source and target expression is bidirectional and declarative.

Since PONS is a purely declarative system, the same syntactic rules in a grammar may be used for analysis as for generation. This is due to the fact that both analysis and generation are related to the same kind of representation, namely the feature structure where syntactic and semantic properties are interrelated, but contained in separate modules.

In PONS no language pair specific information is used in interlingual mode. Neither is any language specific information about how semantic relations are linked to syntactic functions contained in the situation schema. Accordingly, generation in mode 3 requires a fair amount of syntactic processing. To avoid inefficiency, grammars must be written with care, so that the generation algorithm does not build a number of trees representing different rules but identical strings.

As opposed to the situation schema in PONS, the transfer dag in Toolkit is language pair specific and dependent on the direction of translation. It follows from this that the transfer dag is not neutral between analysis and generation and may only be used for the purpose of generation. To generate a string from a transfer dag and to analyse a string to produce an f-structure cannot be reversible operations when execution of transfer rules transforms the source dag. The relation between source and target expression is unidirectional and irreversible.

As a consequence of the transfer strategy and the somewhat procedural character of the system, Toolkit needs separate rules for analysis and generation. Functional ordering rules specify how syntactic functions contained in the transfer dag are projected onto constituents of the target sentence. Moreover, a particular transfer entry specifies in what way semantic roles are linked to syntactic functions in the target language. Considerations of efficiency lies behind the use of separate rules for generation. Both transfer and generation rules are designed to keep the amount of work done during generation at a minimum. A result of this is

that it is not necessary to build parse trees during generation. It should, however, be mentioned that a subset of the rules found in the Toolkit system are in fact neutral between analysis and generation. But analysis rules as well as generation rules employ structure-building operations and are therefore of a procedural kind. It is a question whether it is easy enough to keep track of effects that result from applying and modifying the different kinds of rules in the Toolkit system. This pertains to analysis, transfer and generation rules.

Acknowledgements

The author is indebted to Helge Dyvik, Torbjørn Nordgård, Magnar Brekke and Roald Skarsten for their valuable help, information, criticism and encouragement during the preparation of this paper.

References

Barwise, J. and J. Perry. 1983. *Situations and Attitudes.* The MIT Press, Cambridge, Massachusetts.

Bresnan, J. (ed.). 1982. *The Mental Representation of Grammatical Relations.* The MIT Press, Cambridge, Massachusetts.

Dyvik, H. 1990. *The PONS Project: Features of a Translation System.* SKRIFTSERIE Nr. 39 SERIE B, University of Bergen, Department of Linguistics and Phonetics.

Executive Communication Systems, Inc. 1985. *Machine Translation Toolkit. Utilities User's Manual.* Provo, Utah.

Executive Communication Systems, Inc. 1985. *Machine Translation Toolkit. LECS User's Manual.* Provo, Utah.

Fenstad, J.E., P.-K. Halvorsen, T. Langholm and J. van Benthem. 1987. *Situations, Language and Logic.* Reidel, Dordrecht.

Karttunen, L. 1986. *D-PATR – A Development Environment for Unification-Based Grammars.* REPORT No. CSLI-86-61, Center for the Study of Language and Information, Stanford University.

A Noun Phrase Parser of English

Atro Voutilainen
Helsinki

Abstract

An accurate rule-based noun phrase parser of English is described. Special attention is given to the linguistic description. A report on a performance test concludes the paper.

1. Introduction

1.1 Motivation.

A noun phrase parser is useful for several purposes, e.g. for index term generation in an information retrieval application; for the extraction of collocational knowledge from large corpora for the development of computational tools for language analysis; for providing a shallow but accurately analysed input for a more ambitious parsing system; for the discovery of translation units, and so on. Actually, the present noun phrase parser is already used in a noun phrase extractor called *NPtool* (Voutilainen 1993).

1.2. Constraint Grammar.

The present system is based on the Constraint Grammar framework originally proposed by Karlsson (1990). A few characteristics of this framework are in order.

- The linguistic representation is based on surface-oriented morphosyntactic tags that can encode dependency-oriented functional relations between words.

- Parsing is reductionistic. All conventional analyses are provided as alternatives to each word by a context-free lookup mechanism, typically a morphological analyser. The parser itself seeks to discard all and only the contextually illegitimate alternative readings. What 'survives' is the parse.

- The system is modular and sequential. For instance, a grammar for the resolution of morphological (or part-of-speech) ambiguities is applied, before a syntactic module is used to introduce and then resolve syntactic ambiguities.

- The parsing description is based on linguistic generalisations rather than probabilities. The hand-written rules, or **constraints** are validated against representative corpora to ensure their factuality. Also heuristic constraints can be used for resolving remaining ambiguities.

- Morphological analysis is based on two-level descriptions (Koskenniemi 1983). Large lexicons and informative morphosyntactic descriptions are used to represent the core vocabulary of the language. Words not recognised by the morphological analyser are processed with a very reliable heuristic analyser.

- Parsing is carried out with linear-precedence constraints that discard morphological or syntactic readings in illegitimate contexts. Typically, a constraint expresses a partial generalisation about the language.

1.3. System architecture

A typical analyser in this framework also thepresent one employs the following sequentially applied components:

1. Preprocessing
2. Morphological analysis
3. Morphological heuristics
4. Morphological disambiguation
   - 4a. Grammar-based constraints
   - 4b. Heuristic constraints
5. Lookup of alternative syntactic tags
6. Syntactic disambiguation
   - 6a. Grammar-based constraints
   - 6b. Heuristic constraints

Descriptions pertaining to modules 1–4 are directly adopted from the ENGCG description, written by Voutilainen, Heikkil and Anttila, and documented in Voutilainen, Heikkil and Anttila (1992), Karlsson, Voutilainen, Heikkil and Anttila (Eds.) (forthcoming). Here, only the barest characteristics of modules1–4 in effect, a part-of-speech tagger are mentioned. The reader is referred to Karlsson et al. (forthcoming) for further details and justifications.

- The preprocessor recognises sentence boundaries, idioms and compounds. The ENGTWOL morphological analyser employs a 56,000-entry lexicon and a morphosyntactic description based on Quirk et al. (1985). Some 93–98 % of all word-form tokens in running text become recognised. 'Morphological heuristics' is a rule-based module that assigns ENGTWOL-style analyses to those words not recognised by ENGTWOL itself. About 99.5 % of these heuristic

predictions are correct. Ambiguity in English is a nontrivial problem: on an average, the ENGTWOL analyser furnishes two alternative morphological readings for each word.

- The morphological disambiguator applies a grammar with a set of 1,100 'grammar-based' and another set of 200 heuristic constraints. After the combined application of these 1,300 constraints, 96–98 % of all word form tokens in the text are morphologically unambiguous, while at least 99.6 % of all word-form tokens retain the correct morphological reading. These figures apply to standard non-fiction English. The accuracy may decrease somewhat if the text is colloquial, fiction, dialectal or otherwise non-standard. – To my knowledge, this precision/recall ratio is by far the best in the field.

2. Parsing scheme

The ENGCG description also contains a syntactic grammar based on a parsing scheme of some 30 function tags. The somewhat unoptimal recall and precision of the syntactic description on the one hand, and the observation that the parsing scheme was unnecessarily delicate for some of the applications mentioned above, on the other, motivate a more ascetic parsing scheme. I have designed as new syntactic parsing scheme with only seven function tags that capitalise on the opposition between noun phrases and other categories on the one hand, and between heads and modifiers, on the other. Next, the tags are presented.

- *@V* represents auxiliary and main verbs as well as the infinitive marker to in both finite and non-finite constructions. For instance:

  She should/@V know/@V what to/@V do/@V

- *@NH* represents nominal heads, especially nouns, pronouns, numerals, abbreviations and *-ing*-forms. Note that of adjectival categories, only those with the morphological feature *<Nominal>*, e.g. *English*, are granted the *@NH* status: all other adjectives (and *-ed*-forms) are regarded as too unconventional nominal heads to be granted this status in thepresent description. An example:

  The English/@NH may like the unconventional

- *@>N* represents determiners and premodifiers of nominals (the angle-bracket '>' indicates the direction in which the head is to be found). The head is the following nominal with the tag *@NH*, or a premodifier in between. For instance, consider the analysis of *fat* in *fat butcher's wife*:

fat/@>N butcher's/@>N wife/@NH

The annotation accounts for both of the following bracketings:

[[fat butcher's] wife]
[[fat [butcher's wife]

Our tag notation leaves implicit certain structurally unresolvable distinctions in order to maximise on the accuracy of the parser. For instance, on structural criteria it is impossible to decide whether the butcher or his wife is fat in this case. To avoid the introduction of certain other types of semantic or higher-level distinctions, the tag @>N represents not only what are conventionally described as determiners and premodifiers: also non-final parts of compounds as well as titles are furnished with this tag, e.g. *Mr./@>N Jones* and *Big/@>N Board.*

- *@N<* represents prepositional phrases that unambiguously postmodify a preceding nominal head. Such unambiguously postmodifying constructions are typically of two types: (i) in the absence of certain verbs like 'accuse', postnominal *of*-phrases and (ii) preverbal NP-PP sequences, e.g.

  The man in/@<N the moon had a glass of/@N< ale.

  Structure-based resolution of the attachment ambiguities of prepositionalphrases that are preceded by a verb and immediately by a noun phrase is often very difficult or impossible (Quirk et al. 1985). To maximise on the informativeness of the syntactic analysis, the present description capitalises on the unambiguously resolvable 'easy' cases without paying the penalty of introducing systematic unresolvable ambiguity in the hardcases. It is, however, still quite easy to identify the inherently ambiguous cases, if necessary: they are prepositional phrases tagged as *@AH*, and they are preceded by a nominal head.

Currently the description does not account for other types of postmodifier, e.g. postmodifying adjectives, numerals, other nominals, or clausal constructions. Clausal constructions are ignored because their accurate treatment presupposes effective control of clause-level information (or clause boundaries), which is hard to employ in the present description. Besides, postmodifying clauses would probably be marginal for some applications, at least for index term generation.

- @AH represents adjectival heads, adverbials of various kinds, adverbs (also intensifiers), and also those of the prepositional phrases that cannot be dependably analysed either as an adverbial or as a postmodifier. For example:

  There/@AH have always/@AH been extremely/@AH many people around/@AH.

  Note in passing that ed-forms occurring after the primary verbs 'be' and 'have' are generally analysed as main verbs rather than as *@AH*'s, to which status they could in principle be ranked as potential (adjectival) subject complements. A uniform analysis one way or the other (*@V* vs. *@AH*) is not harmful here because neither category qualifies as a nounphrase in the present application. Besides, the ambiguity due to the subject complement and main verb reading in this type of configuration tends to be unresolvable on structural, and often even on any other, criteria, so the present uniform analysis saves us from some (structurally) unmotivated ambiguity.

- *@CC* and *@CS* are familiar from the ENGCG description: the former represents co-ordinating conjunctions, and the latter represents subordinating conjunctions. For example:

  Either/@CC you or/@CC I will go if/@CC necessary.

Finally, a short sample output of the parser is in order:

```
("<*the>"
   ("the" <*> <Def> DET CENTRAL ART SG/PL (@>N)))
("<inlet>"
   ("inlet" N NOM SG (@>N @NH)))
("<and>"
   ("and" CC (@CC)))
("<exhaust>"
   ("exhaust" N NOM SG (@>N)))
("<manifolds>"
   ("manifold" N NOM PL (@NH)))
("<are>"
   ("be" <SV> <SVC/N> <SVC/A> V PRES -SG1,3 VFIN (@V)))
("<mounted>"
   ("mount" <SVO> <SV> <P/on> PCP2 (@V)))
("<on>"
   ("on" PREP (@AH)))
("<opposite>"
   ("opposite" <Nominal> A ABS (@>N)))
("<sides>"
```

```
   ("side" N NOM PL (@NH)))
("<of>"
   ("of" PREP (@N<)))
("<the>"
   ("the" <Def> DET CENTRAL ART SG/PL (@>N)))
("<cylinder>"
   ("cylinder" N NOM SG (@>N)))
("<head>"
   ("head" N NOM SG/PL (@NH)))
("<$.>")
```

Here *inlet* remains ambiguous due to the modifier and head functions because of a coordination ambiguity.

3. About the parsing grammar

The syntactic grammar contains some 120 syntactic constraints, some 50 of which are heuristic. Like the morphological disambiguation constraints, these constraints are essentially negative partial linear-precedence definitions of the syntactic categories. The present grammar is a partial expression of four general grammar statements:

1. *Part of speech determines the order of determiners and modifiers.*
2. *Only likes coordinate.*
3. *A determiner or a modifier has a head.*
4. *An auxiliary is followed by a main verb.*

We will give only one illustration of how these general statements can be expressed as constraints. A partial paraphrase of the statement *Part of speech determines the order of determiners and modifiers*: 'A premodifying noun occurs closest to its head'. In other words, premodifiers from other parts of speech do not immediately follow a premodifying noun. Therefore, a noun in the nominative immediately followed by an adjective is not a premodifier.Thus a constraint would discard the @>N tag of *Harry* in the following sample sentence, where Harry is directly followed by an unambiguous adjective:

```
("<*is>"
   ("be" <SVC/N> <SVC/A> V PRES SG3 (@V)))
("<*harry>"
   ("harry" <Proper> N NOM SG (@NH @>N)))
("<foolish>"
   ("foolish" A ABS (@AH)))
("<$?>")
```

We require that the noun in question is a nominative because premodifying nouns in the genitive can occur also before adjectival premodifiers; witness *Harry's* in *Harry's foolish self.*

Regarding the heuristic elements in the grammar, the main strategy is to prefer the premodifier function over head function. The underlying heuristicis that a noun phrase is not directly followed by another unless there is an explicit noun phrase edge – e.g. a determiner or a genitive in between.

4. A test run

The parser was tested against a text collection new to the system. In all, 3,600 words from newspapers, detective stories, technical abstracts and book reviews were analysed. Some of the texts contained characteristics from spoken language and fiction, so the corpus can be considered a somewhat hard test bench for the system.

Of all words, 93.5 % became syntactically unambiguous, and 99.15 % of all words retained the most appropriate syntactic reading, i.e. 31 contextually appropriate readings were discarded. (A little over 97 % of all words became **morphologically** unambiguous; also heuristic constraints were used.) Of these 31 errors, 18 were due to the syntactic constraints; 11 were due to disambiguation constraints, and 2 were due to the ENGTWOL lexicon. Some observations about the misanalyses are in order.

- Errors tend to co-occur. In the following sentence fragments, four contextually legitimate infinitives were discarded by the morphological disambiguator (the misanalysed word is indicated with a slash, followed by the discarded feature.

  ..either to enhance (boost/INF or increase/INF) or to suppress (dampen/INF or decrease/INF) other nodes' activation.

  One of the constraints discards an infinitive if to the left, there is another unambiguous infinitive, and in between, there is neither a coordinating conjunction nor another infinitive marker (e.g. *to* or a modal auxiliary). Parenthetical expressions of this kind were ignored in the grammar, so both *boost* and *dampen* lost their infinitive readings, retaining some other verb readings. The infinitive readings of *increase* and *decrease* were lost as a domino effect: a constraint about coordination forbade a sequence consisting of a non-infinitive verb coordinating conjunction infinitive.

- Generally, a pronoun does not take a determiner or a premodifier. Heuristic constraints capitalise on this, resulting in the following misanalyses:

  ..that is, the same/DET ones should underlie..
  ..are general/@>N ones.

  The determiner reading of *same* as well as the premodifier reading of *general* is discarded. These errors are actually quite easy to correct: *one* is an untypical pronoun in that it quite often takes a determiner or a premodifier. Correcting the relevant constraints presupposes the addition of another context condition that in effect functions as a brake: whenever the pronoun happens to be a form of *one*, a preceding determiner or premodifier reading is left intact.

- In the following cases, the morphological disambiguator lost two noun readings:

  Peanut-butter tan/N.
  Expensive gold watch/N.

  Non-clausal utterances that are not marked as such (e.g. with a heading code) are known to be problematic for the present description, based on the assumption that an utterance ending with a fullstop or a question mark or an exclamation mark is a sentence with at least one finite verb. In the above cases, the finite verb readings of *tan* and *watch* were selected because no other finite verb candidates were available in the 'sentence'.

- Above, it was mentioned that some heuristic syntactic constraints prefer the premodifier function over the head function. A couple of misanalyses resulted:

  ..the relationship/@NH Ashdown had confessed..
  During the same campaign/@NH Tory politicians told..

- Multi-word adjectives turned out to be the most fatal single error source for the syntactic constraints:

  ..might not be language/@>N specific.
  ..error/@>N prone..
  A Cell/@>N Organized Raster Display for Line Drawings <ENDTITLE> " Attribute/ Based File Organization..

  There is a constraint that discards the premodifier function tag of a noun if the following word is an adjective (or a non-finite *ed*-form).

This generalisation misses adjectives consisting of a noun-adjective sequence, e.g. *language specific*. This leak in the grammar can be mended to some extent at least by imposing lexical context-conditions that licence a premodifying noun in front of certain adjectives or non-finite *ed*-forms such as *specific* or *based*, both of which seem to be quite productive in the formation of multi-word adjectives. A representative collection of these adjectives can be extracted from large ENGCG-tagged corpora relatively easily.

Overall, it seems to me that relatively few of the misanalyses are elementary from the point of view of higher-level syntactic generalisations; in terms of lexical knowledge, these errors can often be quite easily anticipated. For instance, a better version of the grammar may still reject premodifying nouns in general in case the following word is an adjective but a limited class of known exceptions, such as *specific*, can be accounted for by imposing further lexical context conditions. The more accurate the present description becomes, the more lexico-grammatically oriented it is likely to be.

These observations seem to bear on a more general question about how lexical information can be employed in structural analysis, such as part-of-speech disambiguation. One view held in the literature has, roughly speaking, been to identify using structural information with grammar-based methods, and using lexical information (as lexical preferences) with statistical methods (see e.g. Church 1992; Church and Mercer 1993). Our observation is that information about lexis certainly is a useful addition to more general structural information, and, more importantly, lexical information can also be employed in a grammar-based system, such as the present reductionistic one. Furthermore, the superior recall/precision ratio of the present system suggests that a rule (or knowledge) based use of lexical information, in conjunction with more general structural information, may be preferable over using lexical information in the form of probabilities.

5. Technical information

The ENGTWOL morphological analyser uses the two-level program by Kimmo Koskenniemi and Lingsoft, Inc. The latest version of the Constraint Grammar parser was written by Pasi Tapanainen. Also several Unix utilities are used in the present prototype. On a Sun SPARCstation 10/30, the whole system from preprocessing through syntax analyses some 400 words per second. Some optimisation efforts would be worthwhile; at present, much of the processing time is taken by very simple operations that have not been implemented effectively. The hardest problem of parsing with a large grammar has already been

addressed quite satisfactorily: disambiguation and syntactic analysis together can be carried out at a speed of more than 1,000 words per second.

The system will become available. Contact the author for further details, e.g. by email to Atro.Voutilainen@Helsinki.FI.

References

Church, K. 1992. *Current Practice in Part of Speech Tagging and Suggestions for the Future.* In Simmons (ed.) 1992. *Sbornik praci: In Honor of Henry Kucera.* Michigan Slavic Studies.

Church, K. and R. Mercer. 1993. *Introduction to the Special Issue on Computational Linguistics Using Large Corpora.* COMPUTATIONAL LINGUISTICS, Vol. 19, Number 1.

Karlsson, F. 1990. *Constraint Grammar as a framework for parsing running text.* In Karlgren, H. (Ed.) COLING–90. *Papers presented to the 13th International Conference on Computational Linguistics*, Vol. 3. Helsinki, Finland.

Karlsson, F., A. Voutilainen, J. Heikkil and A. Anttila (eds.). (Forthcoming). *Constraint Grammar: a Language-Independent System for Parsing Unrestricted Text.* Mouton deGruyter.

Koskenniemi, K. 1983. *Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production.* Publication No. 11, Department of General Linguistics, University of Helsinki.

Quirk, Randolph, S. Greenbaum, G. Leech and J. Svartvik. 1985. *A Comprehensive Grammar of the English Language.* Longman, London.

Voutilainen, A. 1993. *NPtool, a detector of English noun phrases.* In *Proceedings of the Workshop on Very Large Corpora, June 22, 1993.* Ohio State University, Ohio, USA.

Voutilainen, A., J. Heikkil and A. Anttila. 1992. *Constraint grammar of English. A Performance-Oriented Introduction.* Publications No. 21, Department of General Linguistics, University of Helsinki.

Vad jag i min verksamhet som språkvårdare skulle vilja att datorlingvistiken bidrog med

Margareta Westman
Stockholm

Ska det här vara ett utbyte så måste jag börja med att förklara vad vi språkvårdare håller på med. Inte minst eftersom många människor, inklusive lingvister, ofta har en lite skev uppfattning om vad språkvårdare egentligen gör.

En av våra uppgifter är att tala om för folk hur de ska tala eller skriva, och det gör vi direkt, till exempel per telefon, eller indirekt genom handledningar, ordböcker, språkspalter och radioprogram. Däremot gör vi det aldrig utan att vara ombedda, tillfrågade. Utöver ren rådgivning verkar vi också för att hålla i gång en allmän diskussion om språket, för att höja medvetenheten.

Man kan undra hur det kommer sig att folk över huvud taget frågar hur något ska heta eller vad ett ord betyder. Alla som kan språket kan ju språket – det är ett axiom, tror jag, i språkvetenskapen.

Ändå är det ju så att väldigt mycket av vårt språkkunnande är inlärt på ett mer systematiskt sätt än det rent spontana samtalspråket är. Riktigt fullärd blir man väl heller aldrig ens i sitt eget språk. Detta hänger i sin tur ihop med att språket inte är statiskt. Levande språk förändras ständigt.

Den språkform vi språkvårdare yttrar oss om är standardspråket, alltså det språk vi alla har lärt oss som en övernorm som används för att skriva och också tala i mera offentliga eller stora sammanhang.

Hur kan vi då upplysa folk om hur det ligger till? Vi måste grunda våra råd på bruket, vi kan inte förlita oss enbart på fåtöljlingvistisk intuition. Vi måste faktiskt samtidigt både misstro vår intuition – och utnyttja den.

För att kunna ge bra råd behöver vi kunskap på tre nivåer av språket. Vi behöver känna till

1) bruket, hur folk gör, dvs. faktiskt talar och skriver,

2) uppfattningen, hur folk tror att de gör, dvs. de allmänna åsikterna om hurdant språkbruket är,

3) idealet, hur folk anser att man bör göra, dvs. människors idealbild av språket.

De här tre olika aspekterna är viktiga alla tre. De ingår alla på olika sätt i språkkunnandet och i många fall måste vi undersöka alla nivåerna.

Först något om bruket

Den självklara lösningen vore att undersöka förhållandena i en textkorpus och räkna. Så enkelt är det dessvärre inte. Man kan konstatera frekvenser av det ena eller det andra. Men hur är det vid närmare skärskådande? Är de observerade skillnaderna helt fritt varierande eller finns det subtila betydelseskillnader i olika textkontexter? Eller skillnader i olika sociala kontexter? Frågan är alltså: Vad är egentligen **samma** fall?

Problemet kan illustreras med ett exempel. Jag tar upp uttrycket *vare sig – eller* som är en av de vanliga käpphästarna för språkriktighetsivrare, samtidigt som det är principiellt intressant eftersom det gäller frågan om förhållandet mellan negation och nekande innebörd.

Det här uttrycket används ofta utan något *inte*, trots att en negation av hävd har ansetts nödvändig för att den riktiga innebörden ska förmedlas.

"Äsch", säger ni, "ett enskilt uttryck, det är ingen konst att fånga upp via någorlunda stora korpusar. Det fixar vi lätt."

Och visst kan man få ledning av fynd i stora korpusar, men fynden måste också analyseras. I det här fallet t ex, tycks det finnas en skillnad mellan konstruktioner då *vare sig – eller* föregås av ett finit verb som kan ta en negation intill sig och konstruktioner där *vare sig – eller* i sin helhet är infogat i en inledande nominalfras:

Språket är inte så lätt att förstå, vare sig i teori eller i praxis.

respektive

(Inte?) Vare sig i teori eller praxis är språket lätt att förstå.

I det första fallet kan inte negationen slopas; däremot händer det ofta i fall av den andra typen.

När vi så med hjälp av många textexempel tycker oss ha funnit en möjlig distinktion behöver den prövas. För att värdera exempel som de här behöver vi tillgång till bedömningar av andra än oss själva. Med det menar jag inte lingvisters tyckanden, utan mer systematiskt samlade åsikter. Och därmed är vi inne på nivå 2 och 3, uppfattningarna och idealen.

Det här första exemplet är ändå rätt enkelt, det är ju inte så svårt att söka ett enskilt uttryck. Det finns knepigare fall där bruket tycks hålla på att förskjutas. Vi kan ta frågan om *-a* eller *-e* på adjektivet i bestämd form, alltså, sådant som:

den amerikanske/amerikanska författaren Susan Faludi
den ryske/ryska källan meddelar
universitetets nye/nya datasnille

Traditionell språkvård avvisar kategoriskt *e*-form när nominalfrasen syftar på en kvinna, när huvudordet inte är ett personord eller när huvudordet är ett substantiv i neutrum.

För att få fram ordentligt med exempel på sådant här behövs satslösta, dvs parsade, eller på något sätt preparerade korpusar. Även så är det rätt besvärligt att analysera de olika fallen, för det är många faktorer som tycks samspela: referensen, huvudordets betydelse och morfologi, adjektivets betydelsetyp, hela frasens genretillhörighet och dessutom gamla regionala språkskillnader.

Därtill eller kanske på grund av allt detta tror jag att det finns genuina åsiktsskillnader om hur man får eller bör göra, vilket vi också behöver kartlägga.

Rent syntaktiska problem är naturligtvis inte lättare att konstatera bruket av. Jag har t.ex. svårt att föreställa mig hur jag skulle kunna få fram relevant material över hur subjektsregeln tillämpas vid infinitivuttryck genom att göra datorsökningar.

En sak är helt dock klar: Vi behöver kunna utnyttja stora textkorpusar av skilda slag, från olika genrer och tider. Stora, väldefinierade och verkligen spridda över genrer och tid på så vis att man kan få jämförbarhet. Spridning över genrer är inte minst viktigt för vårt stora projekt med en konstruktionsordbok. I den ska vi visa vilka typer av bestämningar som olika ord måste ha respektive kan ta och vilka fraser, mer eller mindre fasta, de kan ingå i.

Jag undrar också om man inte borde försöka få med det man kan kalla vuxnas folkliga skrivande, alltså sådant som klubbtidningar (kan gälla

husdjur – sport – frimärken), annonsblad, föreningsprotokoll. På så vis skulle vi kunna få inblick i hur texter ser ut när ingen central kontroll varit verksam.

Detta om bruket.

Om folks uppfattningar

Det nästa vi behöver är alltså kunskap om folks uppfattningar och ideal. Hur samlar och registrerar man data om vad folk anser att de gör? Detta är faktiskt viktigt. Mycket inlärning av språk vilar på det, inklusive vår egen inlärning av skriftspråket och främmande språk.

På Språknämnden hade vi en gång en idé om att upprätta en panel av språkkunnigt folk eller folk som talar och skriver i offentliga sammanhang och registrera deras uppfattningar (jfr inledningen i *American Heritage Dictionary*). De här personerna skulle få sig underställda olika språkproblem som är på tapeten, både gamla inkörda och nya, och få bedöma vad som går an och vad som inte går an. Det gäller alltså någon sorts regelbundet återkommande språkmentometer som kanske kunde skötas via datorer. Så kunde man med jämna mellanrum få en översikt där åsiktstrycket på olika punkter registrerats. Helst skulle man förstås vilja stämma av dessa personers egen praxis mot deras tro om sin praxis och även deras uttryckta ideal.

En annan källa till kunskap om uppfattningar är litteraturen om språk. Här kommer våra planer på en databas som kunskapsbank in. I den ska läggas in för det första referenser till språkvetenskaplig litteratur, handböcker och språkspalter. Där ska också in vår egen excerpering av nya ord och uttryck, konstruktioner osv. Banken ska tjäna oss på två sätt, dels i den dagliga rådgivningen, dels i vår grundforskning som behövs som underlag för utgivning av ordböcker och handböcker.

Om och när vi någonsin får råd med den utrustning som behövs kommer vi också att behöva bistånd av datalingvister för att lägga upp det hela på det sätt som är klyftigast för våra behov, och för andras, ty ytterligare ett syfte med basen är att göra våra samlingar tillgängliga för andra forskare.

Ett behov här är bättre scanningsteknik, dels för att föra över våra kort i basen, dels för att föra in nya excerpter.

Vårt riktigt stora problem just nu är emellertid åtkomligheten, eller den bristande åtkomligheten hos existerande textbaser. Vi vet ännu inte om vi – utan att behöva betala väldiga årliga avgifter som vi inte har råd med –

kan få komma in på universitetsdatanätverken. Hittills har de varit slutna för oss. Numera kommer allt fler korpusar på CD, och det skulle kunna vara en lösning. En vanlig invändning mot den lösningen är att man då inte är inne i de senaste uppdateringarna. Det är emellertid en klen invändning om alternativet är att man inte är inne alls.

Om nya ord

Jag vill gärna också ta upp ytterligare en forskningsdel av vår verksamhet. Språknämnden ska enligt sina stadgar följa svenska språkets utveckling i tal och skrift – som den anspråklösa uppgiften är formulerad.

Vi registrerar alltså nyheter i språket, bland annat nya ord. Det är ju på det området som saker händer snabbast och tydligast.

Nya ord tillkommer på olika sätt, som gör det olika svårt att komma åt dem genom någon sorts automatisk excerpering. Det kan gälla:

1. Helt nya ordformer som *deska* (arbeta med desktop), *krockkudde* (i bilar), *roligan* (om snäll, lugn fotbollsentusiast), *tjugolapp* (om tjugokronorssedeln),

2. Nya fraser

   a) med nya ord i som *varken bu eller bä* (varken det ena eller det andra),

   b) utan nya ord i som *gröna räkenskaper* (om bokföring där man räknar in miljökostnader), *fiska danskt* (som går så till att man sätter sig en bit upp från stranden med matsäcken och låter den yngste i laget vakta metspöna och ropa om det händer något)

3. Gamla ord i ny betydelse som *tjuga* (om tjugokronorssedeln), *golv* (om undre gräns),

4. Nya bildliga användningar som *logga ut* (för 'dö'), *ta fram* (för 'utarbeta')

Den första gruppen borde vara lättast att datorfinna men den blir väl överbelastad av alla tillfälliga sammansättningar – som i och för sig kan vara intressanta. De övriga typerna måste vara mycket besvärliga att söka automatiskt. Eller?

Men – än en gång – vi behöver verkligen korpusar och konkordanser – tillgängliga och lättanvända.

Till slut en fundering

Om vi skulle få tillgång till perfekta korpusar med spridning över tider och genrer, finns det då en risk att vi alla kommer att medverka till att låsa språkutvecklingen vid status quo? Normeringskraften hos t.ex. ordböcker är ju stark. Riskerar vi rundgång på ett sätt som aldrig hittills har inträffat?

Jag tror att risken eller chansen är liten – men det är verkligen viktigt att normering och råd grundar sig på bredast och djupast möjliga insikter i språklivet och inte på diverse idiosynkratiska föreställningar. Och för detta behöver vi hjälp av datorlingvisterna.

From English to PFO: A Formal Semantic Parser

Jordan Zlatev[1]
Stockholm

Abstract

Pagin and Westerståhl (1993) present a formalism called PFO (**P**redicate logic with **F**lexibly binding **O**perators) which is said to be well-suited for formalizing the semantics of natural languages. Among other things, PFO permits a compositional formalization of "donkey sentences" of the type *If a farmer owns a donkey he beats it.*

In this paper we present a formal procedure and its computer implementation (written in PROLOG) that translates from a limited fragment of English to PFO, i.e. a *formal semantic parser.* The translation is done in two steps: first a DCG grammar delivers a parse tree for the sentence; then a number of translation rules that operate on (sub)trees apply to the analysed sentence in all possible orders which may give rise to different "interpretations". For example the sentence *Every man does not love a woman* receives 6 different formalizations corresponding to the 6 possible orders of applying the universal quantification rule, the existence quantification rule and the negation rule.

Other ambiguities which the parser accounts for are those between anaphoric and deictic interpretations of pronouns: for the sentence in the first paragraph the parser will provide a formalization in which the variable for *he* is co-indexed with that for *farmer* (the "anaphoric" interpretation) and a formalization with a new variable (the "deictic" one).

1. Introduction

PFO, which stands for **P**redicate logic with **F**lexibly binding **O**perators, is a logical formalism developed by Peter Pagin and Dag Westerståhl (cf. Pagin & Westerståhl 1993, hence P & W). Its novelty consists in the fact that it permits a compositional formalization of certain problematic natural language constructions not by extending the semantics of first-order predicate logic (PL) as in e.g. Discourse Representation Theory (DRT, Kamp 1981), but by changing its syntax.

Section 2 reviews the motivation for developing PFO, presents it in brief, compares it to PL and shows how the first, but not the second allows for a compositional formalization of "donkey sentences". This section is closely based on P & W, sections 1–4.

---

[1]The research reported in this paper was done while participating in the project *Logic with Flexibly-Binding Operators* at the Department of Philosophy, Stockholm University during the 92-93 academic year. The rule system presented in section 3 was established after numerous discussions with Peter Pagin and Dag Westerståhl.

However, Pagin and Westerståhl do not present a formal procedure for translating from sentences in a natural language such as English into PFO. It has been the task of the work reported in this paper to describe such a procedure for a limited fragment of English (comparable to the fragment presented in the classical "PTQ" paper of Richard Montague (1974) though without intensional contexts). **Section 3** will thus present a formalization of the translation from English to PFO for a number of basic linguistic constructions.

Since it is to be entirely formal, this procedure should be equally well performable by a computer program and the programming language PROLOG makes it quite straightforward to express the translation rules as computer code. Implementing the translation procedure as a computer program was a convenient way to check for the consistency of the rules, their ordering, interaction etc. Its purpose has been one of a "debugging device". It is both the translation procedure from section 3 and its implementation, which we briefly present in **section 4**, that we refer to as a "formal semantic parser".[1]

Finally, **section 5** will briefly point out some engineering and theoretical conclusions that derive from the project of implementing a translation procedure English-to-PFO.

2. A brief presentation of the PFO formalism

Through PFO, P & W challenge "... the view that certain natural language constructions with anaphoric pronouns cannot be *compositionally formalized* in predicate logic, at least not in any reasonable way". [p.189, my italics]

The principle of compositionality stating that "the meaning of a complex expression is a function of the meaning of its parts" is both vague and controversial and something more will be said about it in section 5. But the notion of a "compositional formalization" is quite straightforward: if X is a constituent of Y in NL (natural language) and X is formalized as X_{FL} and Y as Y_{FL} in FL (formal language), then X_{FL} is to be a constituent of Y_{FL} in FL.

The "natural language constructions" that do not seem to fulfil this requirement include the so-called "donkey sentences", brought to the attention of the linguistic community first by Geach (1962). Consider (1),

---

[1]Strictly speaking, as sections 3 and 4 make clear, both the formalization procedure and the implementation consist of a *syntactic parser*, which delivers a phrase structure tree, and a *translator* that in a number of consecutive steps transforms the parse tree into a PFO formula. By "formal semantic parser" we mean both parts.

which is *not* a donkey sentence, and its compositional formalization in PL, (1_{PL}).

(1) If Bill owns a car he is rich
(1_{PL}) $\exists y(car(y) \wedge owns(b,y)) \rightarrow rich(b)$

But (2), which is a donkey sentence, constitutes a problem. (2*), which is derived by analogy to (1_{PL}) is not a sentence (a well-formed formula) in PL: *y* in *drives(b,y)* is not bound. (2?), which is derived by extending the scope of $\exists$, does not have the right meaning. The "right" formalization is, of course, (2_{PL}) but it is not compositional: it does not have as constituent *$\exists$y(car(y) $\wedge$ owns(b,y))* , which is the formalization of *Bill owns a car* .

(2) If Bill owns a car he drives it
(2*) $\exists y(car(y) \wedge owns(b,y)) \rightarrow drives(b,y)$
(2?) $\exists y((car(y) \wedge owns(b,y)) \rightarrow drives(b,y))$
(2_{PL}) $\forall y((car(y) \wedge owns(b,y)) \rightarrow drives(b,y))$

PFO differs from PL in the following three respects:

(a) The *variable-binding operators* of PFO are *binary* rather than *unary*. [X,Y] expresses universal quantification and (X,Y) expresses existential quantification. Furthermore PFO fuses variable-binding and sentential operators, so that [X,Y] also expresses *material implication* between Y and Y and (X,Y) expresses *conjunction*. (3) and (4) would therefore formalize the following way in PFO and PL respectively.

(3) A man sleeps
(3_{PL}) $\exists x(man(x) \wedge sleeps(x))$
(3_{PFO}) (man x, x sleeps)

(4) Every man sleeps
(4_{PL}) $\forall x(man(x) \rightarrow sleeps(x))$
(4_{PFO}) [man x, x sleeps]

The PFO formalizations are both simpler and, in a sense, closer to natural language in providing a "subject" and "predicate" part, and not having to complement with conjunction and implication operators that have no correlate in the sentences.

(b) Variable-binding is *unselective* (PFO), rather than *selective* (PL) which means that all variables common to two immediate subformulas get bound, without any need for explicit indication. So e.g. [Px, (Qy, Rxy)] corresponds to $\forall x(Px \rightarrow \exists y(Qy \wedge Rxy))$.

(c) Finally, quantification priority is from the *outside in*, rather than from the *inside out*, so that e.g. [Px, (Qx, Rxy)]—notice the slight difference from (b) above—will correspond to $\forall x(Px \rightarrow (Qx \wedge Rxy))$.

There is much more to be said about PFO: P & W present a formal specification of its syntax and semantics, show how to perform natural deduction with it and compare it with "dynamic" logics such as DRT. Here I will end this brief presentation by returning to the donkey sentence (2) and show how in PFO it gets formalized analogously to (1), i.e. compositionally.

First both (1) and (2) get translated into an intermediary stage, which is the result of formalizing the *if-(then)* construction.

($1_{PFO'}$) [b owns a car, he is rich]
($2_{PFO'}$) [b owns a car, he drives it]

Then the first subformula in both is transformed according to the formalization rule for indefinite phrases in object position (cf. 3.2) and the pronoun *he* is substituted with the same constant as that for *Bill*.

($1_{PFO''}$) [(car y, b owns y), b is rich]
($2_{PFO''}$) [(car y, b owns y), b drives it]

And finally a pronoun interpretation rule applies to ($2_{PFO''}$) producing

($2_{PFO'''}$) [(car y, b owns y), b drives y]

The last contains as constituents the PFO formalizations of the constituents of (2), and indeed looks very similar to ($1_{PFO''}$) while getting a different kind of interpretation due to the different way of doing variable-binding in PFO. Now to the main subject of this paper: the exact rules and derivational procedure for e.g. arriving from (2) to ($2_{PFO'''}$), i.e. from English to PFO.

3. Formalizing the translation from English to PFO

It turned out convenient to divide the formalization of the translation procedure English-to-PFO into two stages: (a) a syntactic analysis of the English sentence and (b) a translation of the parse tree produced by (a) into a PFO formula. The main advantage of this modular design is that the translation rules of stage (b) can be "structure-dependent"[1]: i.e. their application can depend on non-terminal as well as on terminal symbols.

[1]Cf. Chomsky (1975) for an argument for the necessity of "structure-dependent" rules.

3.1 Syntactic analysis

The grammar used for parsing the English sentences is a context-free phrase structure grammar with the small generalization allowed by adding the morphosyntactic *features*[1] CASE (with values `nom` and `acc`) for pro-nouns and FIN(iteness) (with values `fin, inf`) for verbs. These serve as constraints on the phrase structure rules, disallowing sentences such as:

**Him* loves Mary. *John loves *he*.
*Pedro *own* a donkey. *Pedro does not *owns* a horse.

A third feature, GEN(der) (with values `fem, masc, neutr`) is marked in the lexicon for nouns and pronouns. It does not play a role in the syntactic analysis, but it does in the translation rules that deal with pronoun interpretation (cf. next subsection).

The only peculiarity of the grammar worth mentioning is the use of two noun phrase subcategories with corresponding symbols in the grammar NPs and NPq. The latter includes noun phrases that have *every* or *no* as determiners (such as *every man* or *no woman that sleeps*) while the first includes pronouns, proper names and noun phrases with determiners *a* and *the*. The reason for this is semantic: NPq:s involve rules of universal quantification for their formalization and the translation rules described in the next section require this distinction in order to avoid producing incorrect formalizations for sentences that involve disjunction. The following rules from the grammar see to it that if at least **one** of the noun phrases in a disjunction is an NPq, the whole disjoint noun phrase is an NPq.

NPs -> NPs or NPs
NPq -> NPs or NPq | NPq or NPs | NPq or NPq

The grammar in its entirety is given in *Appendix A*.

3.2 Translation rules

Once an English sentence is analysed with the help of the grammar, it is available to the translation rules. This is how the first rule used in the translation procedure looks like:

(R1) <<every <CN>cn>npq <VP>vp>s ⇒
[<<CN>cn <x>np>s, <<x>np <VP>vp>s]

[1] As in *unification-based grammars* (cf. Shieber 1986).

All rules have this common form: The left-hand side of the translation symbol, ⇒, is a *structural description.* The right-hand side is a *PFO formula.* The brackets "<" and ">" mark phrase structure (in order to avoid confusion with the PFO operators), with an index on the right specifying the syntactic category. Symbols in capital letters stand for *phrase-structure variables*, i.e. any part of the phrase-structure tree that has the category specified by its index. (As can be seen, the variable symbols and their indices coincide, so to simplify the notation we will abbreviate $<W>_W$ as W in the following). The structural description part always contains reference to some "logical word" such *every, if, or* etc. or to a pronoun, while the PFO formula has *PFO-variables* of syntactic category NP; the significance of this will be seen in a moment. (The marking of a PFO-variable with $<...>_{np}$ will also be omitted for abbreviation.)

Notice also that the structural description requires that the input to a translation rule be of syntactic category S, which is also the category of the two subformulas on the right-hand side. Rules can operate on the subformulas produced by other rules. They can apply in all possible orders and when we have reached a PFO formula on which no other rule can apply, we have a *PFO formalization* of the initial English sentence.

One (negative) consequence of the fact that rules apply on whole sentences is that rules that refer to a noun phrase in their structural description need to come in pairs: one for when this noun phrase is "subject" as in (R1) and one when it is "object", such as (R2).

(R2) $<NP <Vtr <every\ CN>_{npq} vp>_s \Rightarrow [<CN\ x>_s, <NP <Vtr\ x>vp>s]$

These are the rules of *universal quantification.* The rules of *existence quantification,* (R3) and (R4), introduce one more complication: U and W are *anonymous phrase-structure variables*, they can be instantiated by any part of the phrase-structure that otherwise fulfils the structural description.

(R3) $<<U <a\ CN>_{nps}\ W>_{np}\ VP>_s \Rightarrow (<CN\ x>_s, <<U\ x\ W>_{np}\ VP>_s)$
(R4) $<NP <U <a\ CN>_{nps}\ W>_{vp}>_s \Rightarrow (<CN\ x>_s, <NP <U\ x\ W>vp>s)$

The purpose of these variables is to allow the existential quantifier of an (indefinitly) embedded indefinite noun phrase to have a wider scope than a linearly preceding universal quantifier. If (R1) applies to (5) first (after the sentence is syntactically analysed) it will produce the PFO formula ($5_{PFO'}$).

(5) Every man who owns a donkey sleeps
($5_{PFO'}$) $[<<man\ who <owns <a\ donkey>_{nps}>vp>_{cn}\ x>_s, <x <sleeps>vp>s]$

Now (R5), which provides a conjunctive interpretation of relative clauses can apply to the left subformula to produce ($5_{PFO''}$).

(R5) <<N Comp VP>$_{CN}$ x>$_S$ => (<N x>$_S$, <x VP>$_S$)
($5_{PFO''}$) [(<man x>s, <*x* <*owns* <*a donkey*>*nps*>*vp*>*s*), <x <sleeps>$_{VP}$>$_S$]

Finally (R4) can apply, with U instantiated as *owns* and W as nil, to the italicized subformula — remember that x is of category NP! — to yield ($5_{PFO'''}$) which is equivalent to (5_{PL}).

($5_{PFO'''}$) [(<man x>$_S$, (<donkey y>$_S$, <x <owns y>$_{VP}$>$_S$)), <x <sleeps>$_{VP}$>$_S$]
(5_{PL}) $\forall x \exists y((man(x) \wedge donkey(y) \wedge owns(x,y)) \rightarrow sleeps(x))$

However, (5) has another interpretation, which would correspond to the PL sentence obtained by exchanging the places of quantifiers. The corresponding PFO formalization can be obtained by starting with (R3) with U = *every man who owns* and then (R1) and (R5):

R3: (<donkey x>$_S$, <<every man who owns x >$_{npq}$ <sleeps>$_{VP}$>$_S$)
R1: (<donkey x>$_S$, [<<man who owns x >$_{CN}$ y>$_S$, <y <sleeps>$_{VP}$>$_S$])
R5: (<donkey x>$_S$, [(<man y>$_S$, <y owns x >$_S$) , <y <sleeps>$_{VP}$>$_S$])

Similarly, by applying (R1) + (R4) + (R10) ((R10) is one of the two rules for negation) in the six possible orders, six different formalizations of e.g. *Every man does not love a woman* will be derived, corresponding to the six different possible orderings of the quantifiers $\forall$, $\exists$ and the negation operator $\neg$ in PL.

(R10) <NP <Aux not W>$_{VP}$>$_S$ $\Rightarrow$ [<NP <Aux W>vp>s, $\perp$]

The list of rules for the fragment includes rules for *definite noun phrases* and *disjunctions*, which are somewhat more complex, but introduce nothing essentially new. The rules for translating pronouns to variables, e.g. the rule needed to transform ($2_{PFO''}$) to ($2_{PFO'''}$) above, however, differ more. Their task is to produce a formalization which corresponds to an *anaphoric interpretation* (a variable which is co-indexed with the variable of a possible antecedent) whenever it is syntactically and semantically possible and/or a *deictic interpretation* (a new variable). (6) and (7) demonstrate cases when syntactic respectively semantic constraints do not permit an anaphoric interpretation of the final pronoun.

(6) If Pedro owns a donkey, he beats her
(7) If Pedro owns every donkey, he beats it

The first constraint is enforced through the GEN feature, mentioned in section 3.1. The subject pronoun rule is (R17).

(R17) <<Pron:*GEN*>$_{nps}$> VP>$_{s}$ ⇒ <*x* VP>$_{s}$
IFF <<N:*GEN*>$_{cn}$ *x*>$_{s}$
<*y* VP>$_{s}$

The "IFF <structure>" statement serves as a constraint on whether the variable *x* can be used: it is possible only if the specified structure exists as a subformula somewhere in the current PFO formula (i.e. the one that the structural description is a part of as well). In the case of (R17) this means that the current PFO formula should have a noun with the same value for the GEN feature and the same PFO-variable *x*. This condition will not be fulfilled for (6) so the only part of the rule applicable will be the part that introduces *y*, a new variable.

The semantic constraint necessary is somewhat more complex. The anaphoric PFO formalization of (7) is (7_{PFO}) which will indeed be produced by the translation rules.

(7_{PFO}) *[[donkey x, p owns y], p beats x]

This formalization can be disallowed through a constraint such as the one discussed by Pagin & Westerståhl stating in effect that a variable that is quantified within [X,Y] is not to be used outside [X,Y]. This, however, has been more difficult to express procedurally than one can imagine. It is not as simple as to say that PFO-variables introduced by universal quantification rules such as (R1), (R2) and (R10) are not "reusable". Example (8) has an anaphoric interpretation, ($8_{PFO'}$), despite of that.

(8) Every man loves a woman that pleases him
($8_{PFO'}$) ([man x, ((woman y, y pleases x), x loves y]

But neither can a pronoun always co-refer with a noun that is within the same sentence; a different order of applying the translation rules (e.g. (R4) + (R5) + (R1)) will produce ($8_{PFO''}$) in which *him* cannot be anaphoric.

($8_{PFO''}$) ((woman y, y pleases him), [man x, x loves y])

What seems to be necessary is a mechanism that "remembers" when an universal quantification has been introduced in a PFO-formula and allows co-referece with the universally quantified variable *only among subformulas* of that formula.

4. Computer implementation

As mentioned in the introduction, the purpose of the computer implementation in PROLOG has been mainly one of a debugging device, and therefore the implementation is quite crude. Here we will only present the basics of the notation and "trace" the derivation of the classical donkey sentence *If a man owns a donkey then he beats it.*

The syntactic analysis is performed by a standard *Definite Clause Grammar* (DCG) (cf. Pereira and Warren 1980) which straightforwardly uses the rules in *Appendix A* and PROLOG's built in top-down interpreter with unification to produce a syntactic tree (with nouns and pronouns marked for their GEN feature), in the form of a PROLOG *list.*

```
[s,if,[s,[nps,[[dets,a],[cn,[n,farmer,masc]]]],
     [vp,[vtr,owns],[nps,[[dets,a],[cn,[n,donkey,neutr]]]]]],
   then,
   [s,[nps,[pron,he,masc]],[vp,[vtr,beats],[nps,[pron,it,neutr]]]]]
```

The implementation of the translation rules to apply on this structure also consists of an *input list* and *output list,* which specify the structural description and PFO formula respectively. The following is e.g. the implementation of (R1).

```
r1(SD,PFO_formula) :-
   SD = [s,[npq,[[detq, every],CN]], VP],
   PFO_formula = [all, [s,CN,[np,X]], [s,[np,X],VP]].
```

The only difference from (R1) is that non-terminal symbols are specified in the first position of their respective (sub)list and that PROLOG's square brackets which specify the boundaries of a list are used both to mark phrase structure and, together with the "modifiers" `all` and `exist`, PFO operators. When a translation rule such as as (R3) and (R4) has "anonymous phrase structure variables" this is dealt with in the following way. A four-place predicate, `mem`, looks recursively for a certain constituent within a tree, then, having found it, substitutes it with a formalization and returns the new tree:

```
mem(<Constituent>, <Tree>, <Formalization>, <New_tree>)
```

With its help the following is a faithful implementation of (R4).

```
r4(In,Out) :-
   In = [s,NP1, [vp|Rest]],
   mem([nps,[[dets,a],CN]],Rest,[np, X],NewRest),
   Out = [exist, [s,CN,[np,X]], [s,NP1,[vp|NewRest]]].
```

Let us now trace the gradual transformation of the analysed sentence into a PFO formalization. First the *If-(then)* rule (R16) applies to produce:

```
[all,
       [s,[nps,[[dets,a],[cn,[n,farmer,masc]]]],
            [vp,[vtr,owns],[nps,[[dets,a],n,[n,donkey,neutr]]]]]],
       [s,[nps,[pron,he,masc]],[vp,[vtr,beats],
                 [nps,[pron,it,neutr]]]]]
```

Then the existence quantification rule for subject-NP's (R3) applies to the first subformula of the above:

```
[all,
 [exist,[s,[cn,[n,farmer,masc]],[np,x1]],
       [s,[np,x1],[vp,[vtr,owns],[nps,[[dets,a],
                                       [cn,[n,donkey,neutr]]]]]]],
 [s,[nps,[pron,he,masc]],[vp,[vtr,beats],[nps,[pron,it,neutr]]]]]
```

Then the existence quantification rule for object-NP's (R4) applies to the second subformula of the first subformula of the above to produce:

```
[all,
 [exist,[s,[cn,[n,farmer,masc]],[np,x1]],
            [exist,[s,[cn,[n,donkey,neutr]],[np,x3]],
                      [s,[np,x1],[vp,[vtr,owns],[np,x3]]]]],
 [s,[nps,[pron,he,masc]],[vp,[vtr,beats],[nps,[pron,it,neutr]]]]]
```

The subject pronoun interpretation rule (R17) applies to the italicized subformula, finds a possible anaphor, `[cn,[n,farmer,masc]]`, with the right GEN feature and substitutes *`[nps,[pron,he,masc]]`* with the corresponding variable.

```
[all,
 [exist,[s,[cn,[n,farmer,masc]],[np,x1]],
            [exist,[s,[cn,[n,donkey,neutr]],[np,x3]],
                 [s,[np,x1],[vp,[vtr,owns],[np,x3]]]]],
 [s,[np,x1],[vp,[vtr,beats],[nps,[pron,it,neutr]]]]]
```

Finally we come to the last pronoun, which according to an object pronoun rule (R18) can be substituted with an "old" variable, `[np,x3]`, to yield an anaphoric interpretation, or with a "new" variable, `[np,x5]`, to yield a deictic interpretation.

Apart from some cosmetic details added here for perspicuity, this derivation illustrates the performance of the parser (which also yields a large number of equivalent formalizations).

5. PFO and natural language processing

The project of formalizing and implementing the translation procedure English-to-PFO has lended some support to the claim that PFO is well-suited for formalizing natural language semantics. The rules required for carrying out the formalization procedure are quite simple, yet efficient. The "toy implementation" showed that the translation rules do not involve unpredictable interactions. There is no need for any restrictions on the

order of application independent of the structural description, unlike in "classical" transformation grammar. So from the perspective of (applied) natural language processing PFO may prove to be an attractive formalism because (a) due to its compositional nature it *minimizes ambiguity*, e.g. there is no need for different treatments of *a car* in (1) and (2) and (b) does this without extensively extending first-order predicate logic, i.e. in a relatively *constrained formalism.*

However, the particular kind of compositionality that characterizes PFO, compositionality *on the sentence level*, also showed a few drawbacks. The necessity of having "subject"-"object" pairs of rules was cumbersome in itself, but the possible positions of a noun phrase in a sentence is far greater than that. The formalism must therefore be extended to below-sentence compositionality before it can be truly useful for linguistic description. On the other side, the compositional treatment of "donkey anaphora" in a formalism with "a single, uniform notion of semantic content" (P & W, p. 120) seemed to make it harder to specify the semantic constraint on binding. P & W *do* make a clear specification, but they do it *declaratively*, while the lack of any intermediate structures such as the DRS's of DRT make it necessary for the formalization procedure itself to embody this constraint. As pointed out at the end of 3.2. what seems to be called for is a "short term memory" that keeps track of which rule has applied where in the PFO-formula. But this seems to go against the "single, uniform notion of semantic content".

Finally, it should be reminded once again that "semantic compositionality" is not an unproblematic notion. In one sense—that simple expressions combine to produce complex expressions—it seems to be all-encompassing and thus vacuous. In the other, formal, sense defined in section 2 as a relation between a natural and a formal language it may be too strong a constraint. Modification (e.g. *fake gun*), polysemy, intensional contexts and many other natural language phenomena seem not to be easily coerced into it. If PFO can be extended to deal with some of these other phenomena this would present an even greater challenge.

References

Chomsky, N. 1975. *Reflections on Language*. Maurice Temple Smith Ltd.

Geach, P. Th. 1962. *Reference and Generality*. Ithaca, N.Y.

Kamp, H. 1981. *A theory of truth and semantic representation*. In Groenendijk, Janssen and Stokhof (Eds.), *Formal Models in the Study of Language*. Amsterdam.

Montague, R. .1974. *The Proper Treatment of Quantification in Ordinary English*. In *Formal Philosophy: Selected Papers of Richard Montague*, Thomason, R. (Ed.) New Haven, Conn.

Pagin, P. and D. Westerståhl. 1993. *Predicate Logic with Flexibly Binding Operators and Natural Language Semantics*, in JOURNAL OF LANGUAGE, LOGIC AND INFORMATION Vol 2: pp. 89–129, Kluwer.

Pereira, F. and D. Warren. 1980. *Definite Clause Grammars for Language Analysis*, ARTIFICIAL INTELLIGENCE, 13.

Shieber, S. .1986. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Notes No.4, Stanford, CA.

Appendix A

The context-free grammar with morphosyntactic features for syntactically parsing the fragment of English sentences. Features are marked within square brackets, with **&** signifying conjunction, **=/** "is different from" and **|** disjunction.

```
S -> S or S
S -> if S then S
S -> NP[HEAD = pron & CASE = nom] VP
S -> NP[HEAD =/ pron] VP

NP -> NPs | NPq

NPs -> Dets, CN | PN | Pron
NPq -> Detq CN
NPs -> NPs or NPs
NPq -> NPs or NPq | NPq or NPs | NPq or NPq

CN -> N | N Comp VP

VP -> Vitr
VP -> Aux Neg Vitr[FIN = inf]
VP -> Vtr NP[HEAD = pron & CASE = acc]
VP -> Vtr NP[HEAD =/= pron]
VP -> Aux Neg Vtr[FIN = inf] NP[HEAD = pron & CASE = acc]
VP -> Aux Neg Vtr[FIN = inf] NP[HEAD =/= pron]
VP -> Cop Adj
VP -> Cop Neg Adj

PN -> bill | pedro
N -> farmer[GEN = masc] | donkey[GEN = neutr] | woman[GEN = fem]

Vitr -> sleeps[FIN = fin] | sleep[FIN = inf]
Vtr -> owns[FIN = fin] | loves[FIN = fin] | beats[FIN = fin]
Vtr -> own[FIN = inf] | love[FIN = inf] | beat[FIN = inf]

Dets -> a | the
Detq -> every | no

Pron -> it[GEN = neutr]
Pron -> he[GEN = masc & CASE = nom] | she[GEN = fem & CASE = nom]
Pron -> her[GEN = fem & CASE = acc] | him[GEN = masc & CASE = acc]

Aux -> does
Neg -> not
Comp -> who | that
Cop -> is
Adj -> tired | rich
```

List of participants

Lars Ahrenberg
Datavetenskap
Linköpings universitet
581 83 Linköping
Sweden
lah@ida.liu.se

Gunilla Allstig Lamos
Institutionen för franska & italienska
S:t Eriksgatan 92 4tr
113 62 Stockholm
Sweden

Jan Anward
Institutionen för lingvistik
Stockholms universitet
106 91 Stockholm
Sweden

Hans Arndt
Institut for Lingvistik
Aarhus Universitet
DK-8000 Aarhus C
Denmark
LINHA@hum.aau.dk

Björn Beskow
Institutionen för lingvistik
Uppsala Universitet
Box 513
751 20 Uppsala
Sweden
beskow@ling.uu.se

Klaus von Bremen
Apelvägen 6
181 62 Lidingö
Sweden

Ivan Bretan
Swedish Institute of Computer Science
Box 1263
164 28 Kista
Sweden
ivan@sics.se

Benny Brodda
Institutionen för lingvistik
Avdelningen för datorlingvistik
Stockholms universitet
106 91 Stockholm
Sweden
brodda@ling.su.se

Peter Bursell
Modus Språkteknologi
Box 15055
104 65 Stockholm
Sweden
bursell@ling.su.se

Mats Carlvik
Institutionen för lingvistik
Avdelningen för datorlingvistik
Stockholms universitet
106 91 Stockholm
Sweden

Ole Norling-Christensen
Den Danske Ordbog
Københavns Universitet
Njalsgade 80
DK-2300 København
Denmark

Douglass R. Cutting
Apple Advanced Technology Group
1 Infinity Loop
Cupertino, CA
USA
cutting@apple.com

Ingrid Edmar
Institutionen för franska och italienska
Stockholms universitet
106 91 Stockholm
Sweden

Martin Eineborg
Swedish Institute of Computer Science
Box 1263
164 28 Kista
Sweden
eineborg@sics.se

Robert Eklund
Institutionen för lingvistik
Avdelningen för datorlingvistik
Stockholms universitet
106 91 Stockholm
Sweden
robert@ling.su.se

Gunnar Eriksson
Institutionen för lingvistik
Avdelningen för datorlingvistik
Stockholms universitet
106 91 Stockholm
Sweden
gunnar@ling.su.se

Ruth Feil
Handelshojskolen i Aarhus,
Datalingvistik
Fuglesangs Alle 4
DK-8210 Aarhus V
Denmark
hhadja@uts.uni-c.dk

Björn Gambäck
Swedish Institute of Computer Science
Box 1263
164 28 Kista
Sweden
gam@sics.se

Joakim Gustafson
Institutionen för talöverföring och
musikakustik
KTH
100 44 Stockholm
Sweden
joakim_g@speech.kth.se

Steffen Leo Hansen
Institut for Datalingvistik
Handelshöjskolen i København
Dalgas Have 15
DK-2000 Fredriksberg
Denmark
slh/id@cbs.dk

Tellervo Hyttinen
Statsrådets kansli / språktjänsten
Aleksandersgatan 3 d
SF-00170 Helsingfors
Finland

Peter Ingels
Datavetenskap
Linköpings universitet
581 83 Linköping
Sweden
petin@ida.liu.se

Per Anker Jensen
Institut for Datalingvistik
Handelshöjskolen i København
Dalgas Have 15
DK-2000 Fredriksberg
Denmark

Jussi Karlgren
Swedish Institute of Computer Science
Box 1263
164 28 Kista
Sweden
jussi@sics.se

Sabine Kirchmeier-Andersen
Institut for Sprog og Kommunikation
Odense Universitet
Campusvej 55
DK-5230 Odense
Denmark
ska@dou.dk

Gunnel Källgren
Institutionen för lingvistik
Avdelningen för datorlingvistik
Stockholms universitet
106 91 Stockholm
Sweden
gunnel@ling.su.se

Arne Larsson
Nokia Telecommunications
Transmission Systems, Customer
Services
P.O. Box 12
SF-02611 Espoo
Finland
larsson@ntc02.tele.nokia.fi

Janne Lindberg
Institutionen för lingvistik
Avdelningen för datorlingvistik
Stockholms universitet
106 91 Stockholm
Sweden
beb@ling.su.se

Anders Lindström
Infovox
Box 2069
171 02 Solna
Sweden
Anders.Lindstrom@infovox.se

Mats Ljungqvist
Infovox
Box 2069
171 02 Solna
Sweden
Mats.Ljungqvist@infovox.se

Helge Niska
Tolk- och översättarinstitutet
Stockholms universitet
106 91 Stockholm
Sweden
hniska@seumdc51.bitnet

Bodil Nistrup Madsen
Institut for Datalingvistik
Handelshöjskolen i København
Dalgas Have 15
DK-2000 Fredriksberg
Denmark

Joakim Nivre
Institutionen för lingvistik
Göteborgs universitet
412 98 Göteborg
Sweden
joakim@ling.gu.se

Torbjørn Nordgård
Institutt for fonetikk og lingvistikk
Universitetet i Bergen
Sydnesplass 9
N-5007 Bergen
Norway
nordgaard@hf.uib.no

Ole Norling-Christensen
Den Danske Ordbog
Köbenhavns Universitet
Njalsgade 80
DK-2300 København
Denmark

Claus Povlsen
Center for Sprogteknologi
Njalsgade 80
DK-2300 København
Denmark
claus@cst.ku.dk

Ove Rasmussen
Institut for jysk sprog- og
kulturforskning
Aarhus universitet
Niels Juelsgade 84
DK-8200 Århus N
Denmark

Björn Rauch
Institutionen för lingvistik,
Stockholms universitet
106 91 Stockholm
Sweden

Atle Ro
Institutt for fonetikk og lingvistikk
Universitetet i Bergen
Sydnesplass 9
N-5007 Bergen
Norway
Ro@hf.uib.no

Christer Samuelsson
Swedish Institute of Computer Science
Box 1263
164 28 Kista
Sweden
christer@sics.se

Peter Seipel
Institutet för rättsinformatik
Juridiska institutionen
106 91 Stockholm
Sweden

Ebbe Spang-Hanssen
IAAS, Københavns Universitet
Njalsgade 80
DK-2300 København
Denmark
esh@cphling.dk

Annie Stahél
Institut for Datalingvistik
Handelshöjskolen i København
Dalgas Have 15
DK-2000 Fredriksberg
Denmark

Carin Svensson
Institutionen för lingvistik
Avdelningen för datorlingvistik
Stockholms universitet
106 91 Stockholm
Sweden

Jan Svanlund
Svenska språknämnden
Lundagatan 42, uppg. 5
117 27 Stockholm
Sweden

Anna Sågvall Hein
Institutionen för lingvistik
Uppsala Universitet
Box 513
751 20 Uppsala
Sweden
Anna.Sagvall_Hein@ling.uu.se

Finn Sörensen
Institut for Datalingvistik
Handelshöjskolen i København
Dalgas Have 15
DK-2000 Fredriksberg
Denmark

Torben Thrane
Institut for Humanistisk Informatik
Edb-center
Københavns Universitet
Njalsgade 80
DK-2300 København
Denmark

Martha Thunes
Institutt for fonetikk og lingvistikk
Universitetet i Bergen
Sydnesplass 9
N-5007 Bergen
Norway
Thunes@hf.uib.no

Carl Vikner
Institut for Datalingvistik
Handelshøjskolen i København
Dalgas Have 15
DK-2000 Fredriksberg
Denmark

Atro Voutilainen
Institutionen för allmän språkvetenskap
PB 4
SF-00014 Helsingfors universitet
Finland
avoutila@ling.Helsinki.FI

Helle Wegener
Institut for Datalingvistik
Handelshöjskolen i København
Dalgas Have 15
DK-2000 Fredriksberg
Denmark

Kjell Westerberg
Tekniska Nomenklaturcentralen
Västra vägen 9 C
171 46 Solna
Sweden

Margareta Westman
Svenska språknämnden
Lundagatan 42, uppg. 5
117 27 Stockholm
Sweden

Eva Wikholm
Institutionen för lingvistik
Uppsala Universitet
Box 513
751 20 Uppsala
Sweden
eva.wikholm@ling.uu.se

Jordan Zlatev
Institutionen för lingvistik
Avdelningen för datorlingvistik
Stockholms universitet
106 91 Stockholm
Sweden
jordan@ling.su.se

Conference Program

PROGRAM

TORSDAG 3/6

11.00-13.00,
Södra huset, hus E, plan 3 Registrering

13.00, E 10 Välkomstanförande, allmän information

Session A

Ordförande: Steffen Leo Hansen

13.15-13.40, E 10
Anna Sågvall Hein
Preferences and Linguistic Choices in the Multra Machine Translation System

Session B

Ordförande: Ivan Bretan

13.15-13.40, E 387
Martin Eineborg, Björn Gambäck
Tagging Experiments Using Neural Networks

13.45-14.45, E 10
Peter Seipel
Vad jag i min verksamhet som rättsinformatiker och jurist skulle vilja att datorlingvistiken bidrog med (Föredrag av inbjuden talare med diskussion.)

14.45-15.15
KAFFE

Ordförande: Anna Sågvall-Hein

15.15-15.40, E 319
Björn Beskow
Machine Translation in the Multra System: System Architecture and Control

15.45-16.10, E 319
Christer Samuelsson
A Morphological Tagger Based Entirely on Bayesian Inference

16.15-16.40, E 319
Torbjørn Nordgård
On GB-Parsing and DRS-Construction

Ordförande: Lars Ahrenberg

15.15-15.40, E 387
Arne Larsson, Magnus Merkel
Semiotics at Work: Technical Communication and Translation in a Multilingual Corporate Environment

15.45-16.10, E 387
Atle Ro
A Set Theoretical Account of the Interlanguage Concept in Intelligent Computer Assisted Language Instruction (ICALI) Systems

16.15-16.40, E 387
Robert Eklund
A Probabilistic Word Class Tagging Module Based on Surface Pattern Matching

FREDAG 4/6

Session A

Ordförande: Ebbe Spang-Hansen

09.00-09.25, E 319
Torben Thrane
Constituency and Semantic Interpretation

09.30-09.55, E 319
Peter Ingels
Robust Parsing with Charts and Relaxation

Session B

Ordförande: Gunnar Eriksson

09.00-09.25, E 306
Gunnel Källgren
Aspects on Corpus Construction - Boring Things in Agonizing Detail

09.30-09.55, E 306
Martha Thunes
Machine Translation Strategies: A Comparison of F-Structure Transfer and Semantically Based Interlingua

09.55-10.30
KAFFE

10.30-10.55, E 319
Björn Rauch
Automatisk igenkänning av nominalfraser i löpande text

10.30-10.55, E 306
Benny Brodda
Turtagningsmönster i informella dialoger

11.00-12.00, E 319
Margareta Westman
Vad jag i min verksamhet som språkvårdare skulle vilja att datorlingvistiken bidrog med (Föredrag av inbjuden talare med diskussion.)

12.00-13.30
LUNCH, GRILLEN, LANTIS

Ordförande: Gunnel Källgren

13.30-13.55, E 319
Jussi Karlgren, Björn Gambäck, Christer Samuelsson
Clustering Sentences

14.00-14.25, E 319
Atro Voutilainen
A Noun Phrase Parser of English

Ordförande: Benny Brodda

13.30-13.55, E 306
Helle Wegener, Annie Stahel, Bodil Nistrup Madsen
Domænemodeller og vidensstrukturering

14.00-14.25, E 306
Steffen Leo Hansen
Domænemodeller og domaenemodellering

14.55-15.30
KAFFE

15.30-15.55, E 319
Lars Ahrenberg
Parsning med fältstruktur och konstituentklumpar

15.30-15.55, E 306
Per Anker Jensen, Bodil Nistrup Madsen, Karl Vikner
From Semantic Representation to SQL Queries

16.00-16.25, E 319
Jordan Zlatev
From English to PFO: A Formal Semantic Parser

16.00-16.25, E 306
Ole Norling-Christenssen
Metoder og værktøj til leksikografers arbejde med store tekstkorpora

16.30-16.55, E 319
Joakim Nivre
PLUS - ett pragmatikbaserat dialogsystem

19.00-01.00
BANKETT, LIDINGÖBRO VÄRDSHUS

LÖRDAG 5/6

Ordförande: Torbjørn Nordgård

Ordförande: Jordan Zlatev

Session A

Session B

10.00-10.15, E 319
Information om samnordiska forskarkurser i datorlingvistik finansierade av NorFA.

10.15-10.40, E 319
Claus Povlsen
Natursprogsprocessering i dialogsystemer med talt input

10.15-10.40, E 306
Douglass Cutting
A Practical Part-of-Speech Tagger

10.45-11.10, E 319
Sabine Kirchmeier-Andersen
Pronominal Feature Analysis with Special Reference to a Valency Description of Danish Verbs

10.45-11.10, E 306
Björn Gambäck
On Implementing Swedish Tense and Aspect

11.15-12.15, E 319
Jan Anward
Vad jag i min verksamhet som språkteoretiker skulle vilja att datorlingvistiken bidrog med (Föredrag av inbjuden talare med diskussion.)

12.15-12.45
KAFFE

12.45-13.30, E 319
Avslutande diskussion

Med reservation för ändringar.