Association for Computational Linguistics

EACL 2003

10th Conference of The European Chapter

Proceedings of the 9th European Workshop on Natural Language Generation (ENLG-2003)

April 13th - 14th 2003

Agro Hotel, Budapest, Hungary

Association for Computational Linguistics

EACL 2003

10th Conference of The European Chapter

Proceedings of the 9th European Workshop on Natural Language Generation (ENLG-2003)

April 13th - 14th 2003

Agro Hotel, Budapest, Hungary

The conference, the workshops and the tutorials are sponsored by:

Chief Patron of the Conference:
Dr. Ferenc Baja
Political State Secretary
Office of Government Information Technology and Civil Relations
Prime Minister's Office

Linguistic Systems BV
Leo Konst (Managing director)
Postbus 1186, 6501 BD Nijmegen, Nederland
tel: +31 24 322 63 02
fax: +31 24 324 21 16
e-mail: info@euroglot.nl, leokonst@telebyte.nl,
http://www.euroglot.nl

Xerox Research Centre Europe
Irene Maxwell
6 chemin de Maupertuis
38240 Meylan, France
Tel: +33 (0)4.76.61.50.83
Fax: +33 (0)4.76.61.50.99
email: info@xrce.xerox.com
website: www.xrce.xerox.com

ATALA
Jean Veronis
Jean.Veronis@up.univ-mrs.fr
45 rue d'Ulm
75230 Paris Cedex 5, France
http://www.atala.org

ELRA/ELDA
Khalid Choukri
choukri@elda.fr
55-57 rue Brillat Savarin
75013 Paris, France
Tel: (+33 1) 43 13 33 33,
Fax: (+33 1) 43 13 33 30
http://www.elda.fr

Order copies of ACL proceedings from: Priscilla Rasmussen, Association for Computational Linguistics, 3 Landmark Center, East Stroudsburg, PA 18301 USA, Phone +1-570-476-8006, Fax +1-570-476-0860, URL http://www.acl-web.org.

Preface

The 9th European workshop on Natural Language Generation (ENLG-2003) is part of a biennial series of workshops on Natural Language Generation (NLG) that has been running since 1987, and which alternates with a biennial series of international conferences on the same topic. Previous European workshops have been held at Royaumont, Edinburgh, Judenstein, Pisa, Leiden, Duisburg, and Toulouse (2x). The goal of the workshop is to be an informal meeting which facilitates the dissemination of knowledge and expertise in the field. The 9th issue of the workshop will focus on the following topics:

> *Formal semantics and NLG.* In several subareas of NLG (generation of referring expressions, lexicalization, among others) methods of formal semantics have already been used successfully. By focusing on this topic, we hope to strengthen the theoretical basis of NLG, complementing the application-oriented emphasis that has become dominant recently. The relation between NLG and Formal Semantics will be highlighted in a number of talks, including the invited presentation.
> *Evaluation of NLG systems.* Evaluation is an important factor for all NL systems, which is notoriously underrepresented in the generation field. By focusing on evaluation, we hope to strengthen the empirical basis of NLG, preparing the ground for dedicated activities on this topic. Discussion about evaluation will be stimulated by meetings in smaller groups, which will be asked to report at the end of the workshop.

Out of the 25 papers submitted, 17 were selected for presentation. We trust that the workshop will be as fruitful as its predecessors, complementing other NLG-related events such as the AAAI Spring Symposium 'Natural Language Generation in Spoken and Written Dialogue' (Palo Alto, March 24-26 2003).

WORKSHOP ORGANIZERS:

Ehud Reiter, Univ. of Aberdeen, UK
Helmut Horacek, Univ. of the Saarland, Germany
Kees van Deemter, Univ. of Brighton, UK

PROGRAMME COMMITTEE:

John Bateman, Univ. of Bremen, Germany; *Nadjet Bouayad-Agha*, Univ. Pompeu Fabra, Spain; *Stephan Busemann*, DFKI, Germany; *Jo Calder*, Fourth Person LtD, UK; *Charles Callaway*, IRST, Italy; *John Carroll*, Univ. of Sussex, UK; *Robert Dale*, Macquarie University, Australia; *Claire Gardent*, CNRS Nancy, France; *Pamela Jordan*, Univ. of Pittsburgh, USA; *Rodger Kibble*, Univ. of London, UK; *Emiel Krahmer*, Univ. of Tilburg, The Netherlands; *Chris Mellish*, Univ. of Edinburgh, UK; *Owen Rambow*, Columbia Univ., USA; *Leo Wanner*, Univ. of Stuttgart, Germany.

INVITED SPEAKER:

Henk Zeevat, Univ. of Amsterdam, The Netherlands

ENDORSED BY:

EACL (http://www.eacl.org/)
ACL SIGGEN (http://www.dynamicmultimedia.com.au/siggen/)
ACL SIGSEM (http://www.aclweb.org/siggen/)

WORKSHOP PROGRAM

Sunday, April 13

9:15-9:30	Welcome
9:30-10:00	*Deriving the Communicative Structure in Applied NLG* Leo Wanner, Bernd Bohnet and Mark Giereth
10:00-10:30	*Learning to Order Facts for Discourse Planning in Natural Language Generation* Aggeliki Dimitrimanolaki and Ion Androutsopoulos
10:30-11:00	Coffee break
11:00-11:30	*Restricting the Rhetorical Input for the Non-hierarchical Planning of Document Structures* Nadjet Bouayad-Agha
11:30-12:00	*Handling Dependencies in Reorganizing Content Specifications: A Case Study of Case Analysis* Helmut Horacek
12:00-12:30	Preparatory meetings on Evaluation
12:30-14:00	Lunch break
14:00-14:30	*Applied NLG System Evaluation: FlexyCAT* Nestor Miliaev, Alison Cawsey and Greg Michaelson
14:30-15:00	*Corpus Analysis for NLG* Sabine Geldof
15:00-15:30	*Acquiring and Using Limited User Models in NLG* Ehud Reiter, Somayajulu Sripada and Sandra Williams
15:30-16:00	Tea break
16:00-16:30	*Multilingual Revision* Charles B. Callaway
16:30-17:30	Invited presentation Henk Zeevat

Table of Contents

Dynamic Generation of Cooperative Natural Language Responses in WEBCOOP

Farah Benamara
IRIT
Toulouse III University
benamara@irit.fr

Patrick Saint Dizier
IRIT
Toulouse III University
stdizier@irit.fr

Abstract

We present in this paper a formal approach for the dynamic generation of cooperative NL responses in WEBCOOP, a system that provides intelligent responses in French to natural language queries on the Web. The system integrates reasoning procedures and NLG techniques paired with hypertext links. Content determination is organized in two steps: providing explanations that report user misconceptions and then offering flexible solutions that reflect the cooperative *know how* of the system.

1 Introduction

The main aim of WEBCOOP is to explore, develop and evaluate reasoning procedures and language technologies for the development of a system that provides intelligent cooperative responses in French to natural language queries on the Web. Besides the use of language at several levels (processing queries, generating responses, extracting knowledge from web pages), such a system requires the integration of knowledge representation and the use of advanced reasoning procedures. Moreover, the complexity of reasoning procedures must be kept reasonable in order to optimise tractability, efficiency, and re-usability.

In a first stage, the project is developed on a relatively limited domain that includes a number of aspects of tourism (accommodation and transportation, which have very different characteristics on the web). WEBCOOP is a direct question-answering system, it is not settled within a dialogue perspective and does not include any user model.

Our challenge is to develop a formal framework that integrates reasoning procedures with real-life data extracted from web pages in order to produce web style cooperative NL responses that reflect the accuracy of the reasoning procedures. Our ultimate goal is to develop a formal pragmatics for cooperative responses.

In this paper, we focus (1) on formal aspects of content determination and (2) on the dynamic and interactive generation of cooperative NL responses by integrating NLG techniques with hypertexts (Dale et al, 98). NL responses are produced from logical formulas constructed from reasoning processes. During the content determination stage, the system has to decide, via cooperative rules, what is relevant and then to organize it in a way that allows for the realization of a coherent response. Hyperlinks are dynamically created at generation time. This leaves up to the user the high-level planning tasks inherent to NLG (Reiter and Dale, 97) and improves readability and information access. The standard generation difficulties (lexicalisation, aggregation (Reape and Mellish, 99), argumentation) remain crucial to generate cooperative responses, but their web-style greatly reduces the overall complexity.

In the following sections, we first present a general typology of cooperative responses followed by a few motivational examples that explain the main current aspects of WEBCOOP. Then we analyse the content determination process, organized in two steps: production of explanations that report user misconceptions and then production of flexible solutions that reflect the cooperative *know how* of the system in order to provide help to the user. Finally, central elements of the surface generation component are briefly presented.

2 Motivations

(Grice, 75) maxims of conversation (*quality, quantity, relation, style*) are often used as a basis for designing cooperative answering systems. These systems are typically able to provide general, descriptive answers along with explanations about these answers. They can respond intelligently to false presuppositions and to various types of misconceptions. They can also relax constraints in a question and provide summaries or conditional responses. A number of cooperative systems were designed in the past for databases such as *COBASE* (Minock and Chu, 96) and *CARMIN* (Chakravarthy et al, 90). Most of the efforts were concentrated on fundamental reasoning procedures, while very little attention was paid to question analysis and to NL response generation. For an overview see (Gaasterland et ali, 94).

2.1 A General Typology of Cooperative Responses

Gricean maxims describe fundamental properties of cooperative behaviours. To address this, specific cooperative techniques have been developed and different types of cooperative responses have then been identified. We have structured and classified these types within a functional architecture. In WEBCOOP, cooperativity can be summarized as follows :

First, the elaboration of cooperative responses is based on the use of reasoning procedures that construct :

(1) explanations of a query failure when false presuppositions or misconceptions in the question conflict with knowledge in the database (figure 1 section 2.2.1) (Gal, 88),

(2) conditional responses reflecting relaxation procedures (Gaasterland et al, 92) when the system cannot find any response (figure 2 section 2.2.1),

(3) explanations related to the interpretation of fuzzy terms (figure 3 section 2.2.1),

(4) warnings indicating the temporal dependency of the response especially for elements with a high temporal variability, intrinsic or observed, such as seat availability or fares in air ticket reservation,

(5) conditions directly related to conditional knowledge in the knowledge base (KB) (Burhans and Shapiro, 01) such as constraints associated with the possibility of a service,

(6) textual information from web pages, describing procedures, definitions or causes, obtained from the query evaluation on the knowledge base of indexed web pages such as asking for visa formalities to enter a given country.

Next, the organisation of the informational content of the response before NL generation includes:

(7) cleaning redundant elements,

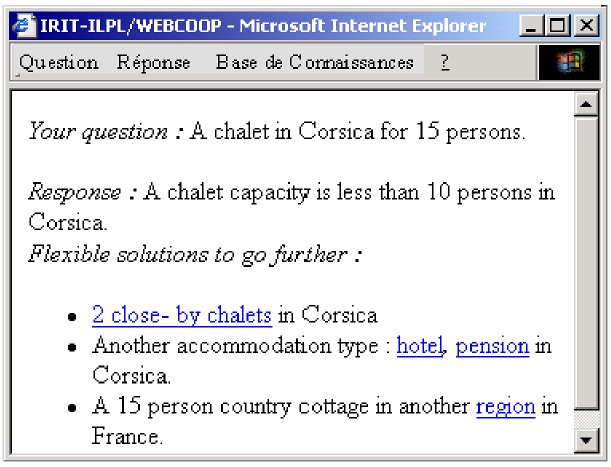
(8) providing intentional responses which are abstract descriptions of large sets of extensional responses (figure 3) (Burhans, 02).

(9) summarizing, sorting explanations or focussing on a particular one, using heuristics that apply to sets of integrity constraints, to produce synthetic responses (Gal, 88).

In this paper we focus on the content determination and the surface generation of (1), (2) and, to some extent, of (3).

2.2 Cooperative Responses in WEBCOOP

In WEBCOOP, user's questions may range from keywords to comprehensive natural language expressions. Responses provided to users are built in web style, by integrating natural language generation (NLG) techniques with hypertext links to produce "dynamic" responses. Responses are structured in two parts. The first part contains explanation elements in natural language. It is a first level of cooperativity that reports user misconceptions in relation with the domain knowledge. The second part is the most important and the most original. It reflects the *'know-how'* of the cooperative system, going beyond the cooperative statements given in part one. It is based on several compo-



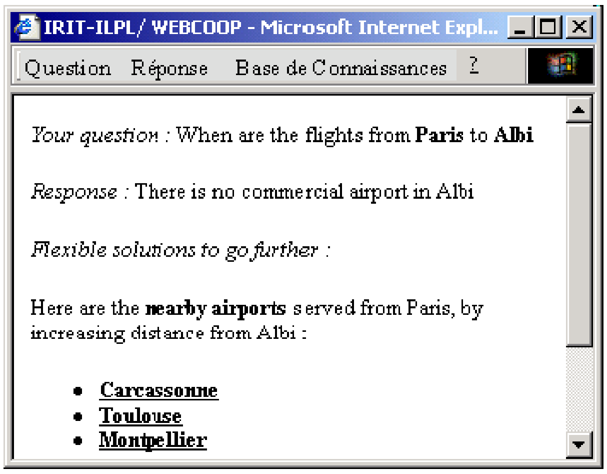
Your question : A chalet in Corsica for 15 persons.

Response : A chalet capacity is less than 10 persons in Corsica.

Flexible solutions to go further :

- 2 close- by chalets in Corsica
- Another accommodation type : hotel, pension in Corsica.
- A 15 person country cottage in another region in France.

Figure 1: Example 1.



Your question : When are the flights from **Paris** to **Albi**

Response : There is no commercial airport in Albi

Flexible solutions to go further :

Here are the **nearby airports** served from Paris, by increasing distance from Albi :

- **Carcassonne**
- **Toulouse**
- **Montpellier**

Figure 2: Example 2

nents: dedicated cooperative rules possibly using knowledge extracted from web pages, relaxation strategies and the domain ontology. The *know-how* component also allows for the dynamic determination of those text fragments to be defined as hypertext links, from which the user can get more information. This allows the user to control navigation within the response hyperlink network and, therefore, the general planning of the response.

2.2.1 A Few Examples

To better characterise our problem, we collected a corpus of question answer pairs in French that reflects different cooperative behaviours. The three examples below are extracted from this corpus. For the sake of readability, they are translated into English.

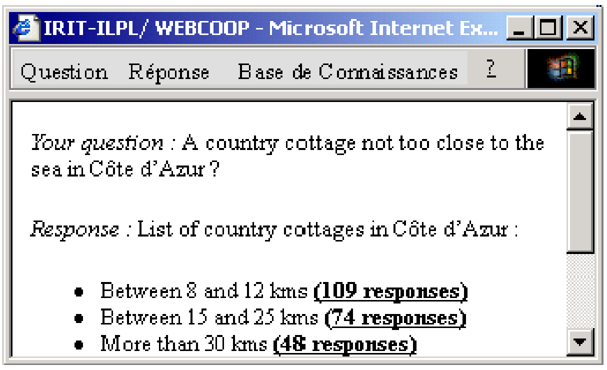
- **Example 1** : Suppose one wishes to rent a 15 person country cottage in Corsica and (1) that observations made on the related web pages or (2) that a constraint or a regulation, indicate that the maximum capacity of a country cottage in Corsica is 10 persons (figure 1).

The first part of the response relates the detection of a false presupposition or the violation of an integrity constraint for respectively cases (1) and (2) above. Case (2) entails the production of the following message, generated by a process that evaluates the question logical formula against the knowledge base: *A chalet capacity is less than 10 persons in Corsica.*

In a second step, the *know-how* of the cooperative system generates a set of flexible solutions as shown in figure 1, since the first part of the response is informative but not really productive for the user. The three flexible solutions proposed emerge from *know-how* cooperative rules based on minimal *relaxation procedures*. The first flexible solution is based on a cardinality relaxation, while in the last two solutions relaxation operates gradually on concepts such as the type of accommodation (hotel or pension) or the region, via the domain model and the ontology. Dynamically created links are underlined. The user can then, at will, get more precise information, dynamically generated from the data base of indexed web pages.

- **Example 2** : In the second example, the user asks for flights from Paris to Albi. The user's false presupposition is detected and a cooperative response is first generated by indicating that *there is no commercial airport in Albi* (figure 2). The *know-how* component goes further by providing responses using first relaxation procedures and then *scalar implicature* to rank the nearests airports served from Paris, by increasing distance from Albi. The airports names are generated as dynamic links in order to get flight information.

- **Example 3** : Another type of cooperativity is a flexible interpretation of fuzzy terms in questions (Figure 3). Suppose the user asks for : *a country cottage not too close to the sea in Côte d'Azur*. Since this question does not contain any false presupposition, the *know how* component produces


IRIT-ILPL/ WEBCOOP - Microsoft Internet Ex...
Question Réponse Base de Connaissances ?

Your question : A country cottage not too close to the sea in Côte d'Azur ?

Response : List of country cottages in Côte d'Azur :

- Between 8 and 12 kms (109 responses)
- Between 15 and 25 kms (74 responses)
- More than 30 kms (48 responses)

Figure 3: Example 3

a direct response. It is elaborated using, in our case, techniques based on geometrical considerations which compute the distance from each country cottage to the sea and group cottages according to dynamically determined distance intervals. When the user clicks on an interval, since there are many responses, the *intentional component* produces a synthesis of responses according, for example, to the localisation of country cottages such as *mountain, country side, river bank* which are at the same conceptual level as *sea side*.

3 Reasoning Procedures for Content Determination

3.1 Detection of Misconceptions

Misconceptions include, among others, a variety of false presuppositions. A false presupposition occurs when a user presupposes the existence of an entity that does not exist or presupposes a relation between entities (or entity type) that is inconsistent with the knowledge base. The detection of these phenomena in a question implies the generation of an explicative response that reports the conflict between that question and the knowledge base.

Formally, presuppositions in a logic conjunctive query, written in a *variable substituted form* (Gal, 88), are all formulas derived from the original query by removing one or several literals. Meaningful presuppositions are those which are *conceptually and logically coherent expressions*. The meaningful subsets of presuppositions are structured as a partial order using the generalisation relation as an order relation.

One of the most studied techniques in the literature for the detection of false presuppositions (Gaasterland et al, 96), is the *merge compatibility* between the query and the integrity constraints of the knowledge base. The merging process produces residues which allow for semantic information to be added to the query without changing its denotation. If the residue contains an incoherence, then a false presupposition is detected (table 1).

The Knowledge Base
Rule : $cottage(x) \rightarrow chalet(x)$ **Constraint** : $Fail \Leftarrow cottage(x) \wedge capacity(x, nb)$ $\wedge nb > 10$.
Interpreted User Question
Chalet in Corsica for 15 persons? $(Entity, x : chalet, in(place, x, Corsica) \wedge$ $capacity(x, 15))$
Deduction Process
Question variable substituted form : $chalet(x) \wedge in(place, x, y) \wedge capacity(x, nb) \wedge$ $nb = 15 \wedge y = Corsica$. Merging the rule and the constraint produces: $Fail \Leftarrow chalet(x) \wedge capacity(x, nb) \wedge nb > 10$ Merging the question and the new constraint produces : the incoherence : $nb = 15 \wedge nb \leq 10$
Generated Logical Response
$chalet(x) \wedge capacity(x, nb) \wedge nb \leq 10$
-Table 1-

In table 1, the knowledge base is first augmented with a new integrity constraint by merging a rule with an existing integrity constraint. Then merge compatibility can optimally be used involving the query and that new constraint. After detection of an incoherence, the following natural language response is produced: *a chalet capacity is less than 10 persons in Corsica.*

Another kind of incoherence is detected from the failure of a search in the extensional database. This is illustrated in the question : *A flight from Paris to Albi?* of example 2 where $airport(Albi)$ is not a fact of the knowledge base.

If more than 2 integrity constraints are violated, the system has to manage possible redundant explanations and to organize independent explanations. For that purpose, (Gal, 88) proposes heuristics to produce synthetic responses.

3.2 Content Determination in the Know How Component

The cooperative *know how* component aims at providing flexible solutions to go further (1) when there is a misconception as above, (2) when the set of extensional responses is too large or empty, or (3) simply when the response needs an elaboration to be useful. This component is based on intentional description techniques (not treated here) and on intelligent relaxation procedures going beyond classical generalization methods. This component also includes additional dedicated cooperative rules that make a thorough use of the domain ontology and general knowledge. This component has potentially a large number of capabilities. We focus here on a basic, but frequently used reasoning schema: the proximity relation that pairs accurate relaxation techniques with ontological knowledge. We first present the reasoning aspects (for content determination) and then the main natural language generation aspects.

Very roughly, a relaxation is a process that rewrites a query in some way to extend its denotation. (Gaasterland et al, 92) define three types of relaxation techniques : a rewrite of a predicate, the broadening of the domain of a variable or breaking a join dependency. In our approach, the notion of answer in the neighbourhood of a question requires that the data base must include a representation of the most salient object properties and their possible interactions. We present in this paper a relaxation technique based on a generic **proximity relation** that uses inherent properties of objects, a conceptual ontology and lexical semantics relations. This proximity relation applies to different ontological or technical domains such as : distance (example 2), fares, capacity (example 1), type of transportation... The proximity relation is associated with constraints (e.g. minimal relaxations, conceptually graded relaxations, which are relevant for the intentional component as well) to produce information useful and relevant for the user. It is one of the most productive know-how rule. It is an iterative process running till a flexible solution is found that leads to a non-empty and coherent solution.

Similarly to the merge compatibility, the proximity relation is a rewriting rule that operates on a formula. We now present its main logical features.

In the next paragraphs, we assume that the question is written in its variable substituted form (table 1). Let T be the set of variables at stake in the residue. Let X be the set of variables that appear in the predicates where an element of T occurs, and Var=$T \cup X$. Then, let F(X,T) be the set of predicates in the query where at least one of the variables in Var occurs, and R the remainder of the query. The general rewriting rule is:
$proximity(F(Var), T) \wedge R \rightarrow NewFormula$,
where $NewFormula$ realises several generic operations, among which:

1. considering several objects of the same type, instead of just one, till the constraint is satisfied. F(X,T) is duplicated to introduce several instances of the same object T till the desired quantity of T is reached (example 1),

2. proposing the same kind of resource but with a ranked set of values close to the initial value at the origin of the failure (example 2),

3. relaxing on the resource via the minimal generalisation that makes consistent the previously inconsistent user constraint, e.g. via the least upper bound in the generalizations lattice (example 1).

The notion of proximity is implemented by the predicate $near(CD, V, Y, Result)$ where CD is a type in a conceptual domain, Y is a resource resulting from the relaxation, of the same type than V. V is the resource on which relaxation operates. Result contains the sorted set of results according to the criteria associated with the conceptual category. W.r.t. the three operations described above, *NewFormula* has the following forms:

1. $F(X1, T1) \wedge F(X2, T2) \wedge X1 \neq X2 \wedge near(CD, X1, X2, Result) \wedge R \wedge (T1 + T2 \geq T)$ (T1, T2 and T contain a single variable).

2. $F(X, T) \wedge$
$near(CD, T : Type, Y : Type, Result) \wedge T \neq Y \wedge R$. T and Y are of the same type.

3. $near(typeof(V), V : Type, Y, Result) \wedge F'(X, V) \wedge R$.
F'(X,V) is F(X,V) without the predicates typing X. V belongs to X.

The instantiation of these formulas can respectively be illustrated by the following examples :

1. For example 1 in figure 1, the rewrite of the question formula is :
   $chalet(x) \wedge in(place, x, z) \wedge capacity(x, nb1) \wedge chalet(y) \wedge in(place, y, z) \wedge capacity(y, nb2) \wedge x \neq y \wedge near(place, x, y, result) \wedge z = Corsica \wedge (nb1 + nb2) \geq 15$,
   which proposes in *result* a set of two nearby cottages x and y instead of one.

2. The query :
   $flight(x, paris, T) \wedge T = albi$,
   in example 2, figure 2, is relaxed into
   $flight(x, paris, T) \wedge T = albi \wedge near(place, T, z, result) \wedge T \neq z$,
   which computes the nearests airports from Albi served from Paris.

3. The logical formula of example 1:
   $chalet(x) \wedge in(place, x, y)$
   $\wedge capacity(x, nb) \wedge nb = 15 \wedge y = corsica$
   is rewritten into :
   $near(typeof(x), x, v, result) \wedge in(place, v, y)$
   $\wedge capacity(v, nb) \wedge nb = 15 \wedge y = corsica$
   according to the domain ontology and the minimal generalization strategy. This allows us to propose objects *v* of a type close to x, e.g.: hotels or pensions (listed in the variable *result*) that can accommodate more than 15 persons in Corsica.

   In the know-how component, a response is often composed of an ordered sequence of proximity rule applications, starting with those responses which are the closests to the initial query. A priori, rules 1 to 3 above are organized by increasing generalizations.

4 Dynamic Surface Generation of Cooperative Responses

Our aim is to maximize over the hyperlinks network (Dale et al, 98) the cognitive as well as the communicative coherence of responses provided to the user. At our level, the main advantages are: (1) to leave up to the user the high-level planning tasks inherent to NLG and (2) to improve readability and information access. Each search may result ultimately in the display of a series of web pages related to the initial query.

4.1 Generating Misconception Reports

Generating misconception reports is relatively simple from a linguistic and NLG point of view when a single false presupposition or integrity constraint violation is detected. The main difficulties are: (1) the generation of expressions in the scope of a negation and (2) lexicalisation. Concerning negation, our strategy in the production of cooperative responses is to keep the scope of negation operators minimal, avoiding thus ambiguities. In that case, either generation is based on the equivalent contraposed form of the negated expression (e.g. not cheaper becomes more expensive) or, when impossible, a negation is generated. In this latter case, it often has the form *there is no, it is not possible to, etc.*. About lexicalisation, our strategy is to keep track of the terms used in the related query and to use them as much as possible. A particular case of lexicalization is the lack of a predicate corresponding to a verb in the formula. Our strategy is then to search for a predicative noun corresponding to a deverbal predicate (or a synonym) with the largest set of arguments possible (to guarantee its central relational role) and to lexicalise it as a verb. For example, cout(x,y) is lexicalized as the verb *coûter* (cost).

Organizing reports with more than one violation is more subtle and requires e.g. redundancy elimination (Gal, 88), ordering with adequate connectors, and pronominalization. Concerning explanation ordering, we are exploring heuristics where the formula which has the largest ratio 'number of free variables / number of predicates' is prefered because of its generality. Interestingly, this technique is the opposite of query optimization in data bases.

Let us now consider a simple example, the NLG of example 1 in figure 1 : $chalet(x) \wedge capacity(x, N) \wedge N \leq 10$. Generation proceeds roughly as follows: aggregation of capacity(x,N) with chalet(x), where capacity(x,N) is the head noun since it is relational, i.e.: *la capacité d'un chalet*. Then comes the VP represented by $N \leq 10$, N is a reference to the subject, while 10 is the object. $\leq$ lexicalises as *est inférieure à*. The whole sentence is then: *la capacité d'un chalet est inférieure à 10.* (A chalet capacity is less than 10).

4.2 Generating Cooperative Know-How Statements

Let us now explain how cooperative know-how statements are generated. Let us concentrate on the proximity rewriting rules given in section 3. An important feature is that the generation of a statement is based on instantiations of the general schemas of this rewriting rule. The NLG process is based on (1) a few predetermined sentence fragments and (2) on the assembling of underspecified fragments which are instantiated from the types of the objects at stake in the formula being treated and from references to lexicalised nodes in the domain ontology. A statement is generated only when the formula produced by the proximity relation rewriting rule has a solution in the knowledge base. Hypertext links range over generalized concepts (usually NPs) or on any term which is a part of the response that corresponds to a non-terminal concept in the ontology. In a response, concepts underlined as hypertext links are a priori in different branches in the ontology. The choice of hyper text links is also guided by the underlying intentions of the response. We give below the three generation schemas and briefly show their application on an example:

1. $F(X1, T1) \wedge F(X2, T2) \wedge X1 \neq X2 \wedge near(CD, X1, X2, Result) \wedge R \wedge (T1 + T2 \geq T)$.
   The duplication (possibly more than 2) of the formula F entails the generation of the determiner *deux* (two), then the abstract form F(X,V) is treated. $Near(-,-,-,-)$ is lexicalised e.g. as *proches* (depending on the ontological type of CD) and then the constraints in R are generated (this is close to their formulation in the query). Since no explicit list of chalets is given, the first NP is underlined as an hypertext link, pointing to a list which can be produced upon request. For example 1, we get: *Deux chalets proches en Corse* (two close-by chalets in Corsica).

2. $F(X, T) \wedge near(CD, T : Type, Y : Type, Result) \wedge T \neq Y \wedge R$.
   F(X, T) is first generated, using the principles shown in section 4.1. For example 2, where F(X, T) = $flight(id, paris, City)$, the following NL fragment is produced: *les vols de Paris* (Flights from Paris), the destination (City) is not lexicalised since it is a variable. Next, the predicate near() is treated, the ontological domain being localization in example 2, a preposition like *vers* is generated, expressing a notion of destination deduced from the thematic role of the argument Y, which is then lexicalised. This lexicalization is an hypertext link if no explicit list is given. Then, *near()* is lexicalised as *proche de* and then we have the proper noun of the original query *Albi* (in T). The sentence ends by an underspecified text fragment realized as: *par distance croissante d'Albi, sont:* (by increasing distance from Albi are:). The full sentence is: *Les vols de Paris vers des aéroports proches de Albi, par distance croissante de Albi sont:* (Flights to airports near Albi, by increasing distance from Albi are:). Then follows the list of city/airport names underlined as hypertext links (instead of the lexicalization of Y which is more general).

3. $near(typeof(V), V : Type, Y, Result) \wedge F'(X, V) \wedge R$.
   where F' keeps track of the constraint. The predicate near is here realized as: *nous vous proposons le/les* followed by the lexicalisation of *Type*, followed by the generation of the constraint(s) in F' as given in the original query. For example 1, we get: *Nous vous proposons les logements touristiques suivants pour 15 personnes:* (we propose you the following tourist accommodations for 15 persons) followed by a list of tourist accommodations Y elaborated in Result, and underlined as hyperlinks. This constitutes a simple intentional response.

5 Conclusion

We presented an approach to the generation of cooperative NL responses in WEBCOOP, a system that provides intelligent responses in French to natural language queries on the Web. We focused in this paper on formal aspects of content determination and on the surface generation of cooperative responses. The content determination process is organized in two steps: production of explanations that report user misconceptions and then production of flexible solutions that reflect the cooperative *know how* of the system in order to provide help to the user. The *know how* component is actually based on intelligent relaxation techniques using a generic **proximity relation**, which goes beyond classical generalization methods. This component also allows for the dynamic determination of the responses to be defined as hypertext links.

The WEBCOOP project is in an early stage of development and its implementation is underway. Reasoning and language generation procedures are implemented in Prolog (with constraint interpreters), while the external aspects are developed in Java.

This project has obviously has several future directions among which we plan to :

- Develop new cooperative know-how strategies and their related logical expressions and implementations (e.g. for fuzzy terms),
- Analyse how NLG argumentation techniques (Horacek, 99) can be used, particularly when a misconception is based on the interaction of several integrity constraints.
- Specify different strategies for generating intentional responses in the know how component, for example when the number of direct responses is very large.
- Study the external display of textual fragments extracted from web pages, in particular for queries requiring *narrative* responses such as procedures or regulations.

A thorough evaluation of the results is crucial in this project at two levels: the quality of the services offered to a user and the re-usability to other domains identifying where are the difficulties, what are the costs, what is domain specific and, finally, what can be shared.

Acknowledgements
We wish to thank Claire Gardent for her relevant suggestions, and the GENI project group.

References

Burhans D. 2002. *A Question Answering Interpretation of Refutation Resolution.* Phd thesis, Buffalo.

Burhans D. T., and Shapiro, S. C. 2001. *Abduction and Question Answering*. Proceedings of the IJCAI-01 Workshop on Abductive Reasoning, Seattle, WA.

Chakravarthy U, Grant J, and Minker J. 1990. *Logic-Based Approach to Semantic Query Optimisation*, ACM Transactions on Database Systems, 15(2):162-207, 1990.

Dale R., Oberlander J., Milosavljevic M., Knott A. 1998. *Integrating Natural Language Generation and Hypertext to Produce Dynamic Documents*, Interacting with Computers 11(2), pp. 109-135.

Gal, A. 1988. *Cooperative Responses in Deductive Databases*, Phd Thesis, University of Maryland, Department of Computer Science.

Gaasterland T, Godfrey P, Minker J. 1994. *An Overview of Cooperative Answering*. Papers in Non-standard Queries and Non-standard Answers, in series Studies in Logic and Computation, Clarendon Press, Oxford.

Gaasterland, T., Godfrey, P., Minker, J. 1992. *Relaxation as a Platform for Cooperative Answering*, Journal of Intelligent Information Systems, volume 1, pp : 293-321.

Grice H. 1990. *Logic and Conversation*, In Cole and Morgan editors, Syntax and Semantic, Academic Press.

Horacek, H. 1999. *Generating Deductive Argumentation in Variable Lengths*, Proceedings of the 7th European Workshop on Natural Language Generation, Toulouse, France.

Minock M, Chu W. 1996. *Explanation for Cooperative Information Systems*, International Symposium on Methodologies for Intelligent Systems, 264-273.

Reiter, R. and Dale, R. 1997. *Building Applied Natural Language Generation Systems*, Journal of Natural Language Engineering, volume 3, number 1, pp: 57–87.

Reape, M, Mellish, C. 1999. *Just What is Aggregation Anyway?*, Proceedings of the 7th European Workshop on Natural Language Generation, May, Toulouse, France.

Restricting the rhetorical input for the non-hierarchical planning of document structures

Nadjet Bouayad-Agha
Technology Department
University Pompeu Fabra
nadjet.bouayad@tecn.upf.es

Abstract

Whereas *rhetorical (or discourse) structuring* is the process of constructing rhetorical (or discourse) structures (RSs), *document structuring* is introduced by Power et al (in press) as the process of building *document structures* (DSs) from a discourse representation, where a DS represents the division of the text into abstract textual units such as paragraphs or orthographic sentences. In the context of the non-hierarchical planning of DSs, the problem that is addressed in the present paper is how to ensure that the DSs produced express all the information (i.e., rhetorical assertions) that has to be conveyed.

1 Introduction

Dynamic approaches to discourse planning have used Rhetorical Structure Theory or RST (Mann and Thompson, 1987) successfully as the basis for their plan representation. In hierarchical planning (Moore and Paris, 1994), RST relations have been translated into plan operators whose definition specifies which relations or propositions can realise their arguments. In non-hierarchical planning (Marcu, 1997a; Mellish et al., 1998a), a valid text plan is built from a set of rhetorical assertions using a general principle of strong compositionality, also called the nuclearity principle, and defined as follows: "a relation R holds between two spans of a text plan if it holds between the most important units (i.e., the nuclei) of the constituent spans". The best RS tree can be constructed using additional constraints based on ordering, adjacency, size of substructures, etc.

Whereas *rhetorical (or discourse) structuring* is the process of constructing rhetorical (or discourse) structures (RSs), *document structuring* is introduced by Power et al (in press) as the process of building *document structures* (DSs) from a discourse representation. A DS represents the division of the text into abstract textual units such as paragraphs or orthographic sentences. This representation is independent from discourse/rhetorical theories. For example, figure 1 illustrates the RS and DS of the text [1] below. The RS's leaves are the four clauses A to D cascaded into three elaboration relations. The DS's leaves are four text-phrases grouped into two text (i.e.,orthographic) sentences within the same paragraph. This example shows that the RS and DS need not be homomorphic structures. Indeed, the grouping of C and D in the RS is lost in the DS. This non homomorphism is due to the linearisation constraints (i.e., how to divide the message into syntactic and document units) which are applied to the message when generating a text, and is especially likely as the message gets bigger. Without a distinction between RS and DS the version [1] of the message could not be generated.

[1] Ciproxin may cause a problem with your kidneys^A called crystalluria^B which results in tiny crystals forming in the urine^C. These crystals cannot be seen by the naked eye.^{D} *Ciproxin tablets 250mg notice*

Power et al (in press) show how DSs with some

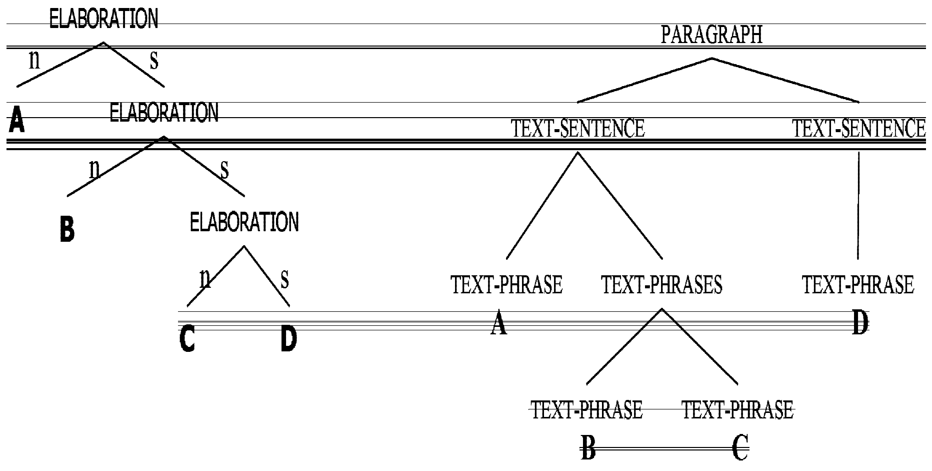


Figure 1: RS (left) and DS (right) for text [1]

degree of homomorphism with their input RS tree can be built systematically. Bouayad-Agha (2001) abolishes homomorphism altogether and presents an approach in which non-hierarchical document structuring is performed. The planning starts from a set of rhetorical assertions similar to Marcu(1997a) and builds valid DSs using a set of locally applied rhetorically motivated constraints on the DS constituents. These constraints are of two types: hard and soft. Constraints concerning the syntactic relations between the constituents are hard and thus *cannot* be violated. For example, when two rhetorically related propositions are in adjacent document constituents, say in two text-sentences, the linearly second of the two clauses must be a main clause (i.e., non subordinate). This is illustrated in the example below, where [2a] violates this constraint (i.e., B is not a main clause) and conveys the wrong message, whilst [2b] satisfies the constraint.

```
concession(sat:C,nuc:B)
explanation(sat:B,nuc:A)
```

[2*a*] #John is a good student^C. Nevertheless, because he failed his exam^B, he looks very upset^A.

[2*b*] John is a good student^C. Nevertheless, he failed his exam^B. As a result, he looks very upset^A.

On the other hand, constraints concerning the non-syntactic structural relations between the constituents of the DS are soft and *can* be violated. The main constraint of this kind is the one that requires that the groupings of non-syntactic DS constituents reflect rhetorical groupings. Its violation is not fatal as illustrated by example [1] above. Nevertheless, as the size of the message increases and the number of soft violations increases too, the resulting text may be infelicitous.

The DSs thus produced can be felicitous in the sense that the original message can be recovered. However, they are not necessarily homomorphic with their corresponding input RS. What has been built is not a rhetorical tree but a tree that respects the document grammar (i.e., grammar for deriving document structures) and the lexical realisation of the relations.

The problem that is addressed in the present paper is how to ensure that the document structures produced express all the information (i.e., rhetorical assertions) that has to be conveyed. No assumption is made about the nature of the rhetorical input. Since we do not impose a homomorphism between the document and the rhetorical structures, we cannot use the nuclearity principle for selecting our assertions for building valid DSs. Nevertheless, we show that the rhetorical input can still be restricted following some basic principles (section 2). The document structurer implicitly enforces those constraints whilst permitting the construction of non-homomorphic DSs, thus allowing a certain freedom of the input with respect to RST (section 3). This confirms that document structure is a useful distinction from discourse representation (section 4).

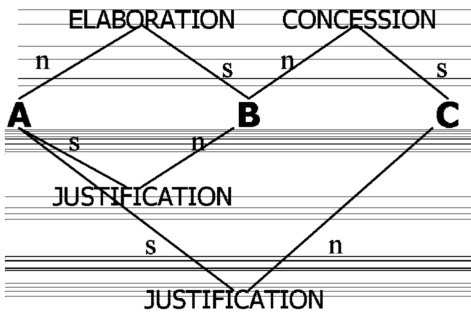


Figure 2: Graph corresponding to [1]

[1*c*] With the next music day scheduled for July 21 (Saturday), noon-midnight^A – I'll post more details later^B – this is a good time to reserve the place on your calendar^C.

[1*d*] With the next music day scheduled for July 21 (Saturday), noon-midnight^A, although I'll post more details later^B, this is a good time to reserve the place on your calendar^C.

[1*e*] With the next music day scheduled for July 21 (Saturday), noon-midnight^A:
- This is a good time to reserve the place on your calendar^C.
- I'll post more details later^B.

2 A rhetorical graph structure

The type of input we start from is a set of rhetorical assertions, a subset of which has to be selected to construct a valid text. This type of input has been used in NLG projects such as ILEX (Mellish et al., 1998b) and could be derived from a domain's knowledge base or from the input plan operators. The following input [1] is an illustrative example. It was introduced by Mann and Thompson (1987) and analysed in terms of rhetorical assertions following RST relations by Marcu (1997b). It is given together with output texts, each of which illustrating the realization of a possible structural configuration:

- [1a] is the original text and expresses R3 and R2;
- [1b] expresses R4 (the marker *with* is taken to give an interpretation of justification) and R2;
- [1c] expresses R1 and R3;
- [1d] expresses R1 and R2;
- [1e] expresses R1 and R4.

```
[1]     R1: justification(sat:A,nuc:C)
        R2: concession(sat:B,nuc:C)
        R3: elaboration(sat:B,nuc:A)
        R4: justification(sat:A,nuc:B)
```

[1*a*] The next music day is scheduled for July 21 (Saturday), noon-midnight^A. I'll post more details later^B but this is a good time to reserve the place on your calendar^C.

[1*b*] With the next music day scheduled for July 21 (Saturday), noon-midnight^A, I'll post more details later^B. However, this is a good time to reserve the place on your calendar^C.

Apart from [1e] (which may have been obtained following some rhetorical aggregation), all the texts obtained correspond to a valid RS. Thus, during rhetorical structuring, the appropriate subset of assertions is selected that forms a valid RS. Each text expresses all the propositions in the input. Note however that a constraint that all the propositions in the input are expressed in the output is not sufficient. For example, given an input with four propositions A, B, C, D, we may select the two assertions R1(sat:A,nuc:B) and R2(sat:C,nuc:D) which realise all the propositions but are not connected in any way.

In our approach, the subset of assertions expressed in the final document structure respects a couple of basic principles. We represent the initial set of rhetorical assertions as a graph as in figure 2 with the following properties:

Labeled. The relations and propositions are represented by nodes linked by edges labelled either as satellite or nucleus. This allows a relation to be the argument of another relation.

Connected. The input must be built into a single connected graph. This is a simplifying assumption for the document structurer as potentially, it would be possible to realise two unconnected graphs in two separate paragraphs or sections.

A valid subset of assertions corresponds to a connected subgraph with no rhetorical circuits, where a rhetorical circuit is a closed path of successive rhetorical assertions. Given these restrictions, there are five subgraphs (figure 3) that can

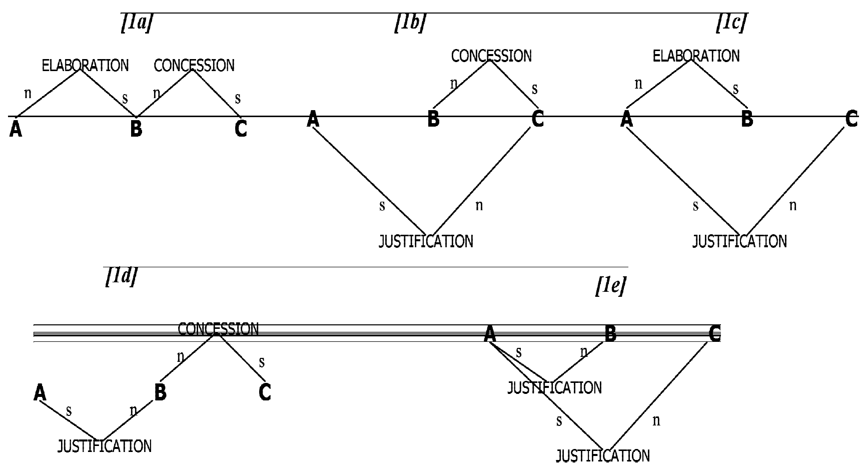


Figure 3: Five graphs derived from graph in figure 2, with no rhetorical circuit

be extracted from the main graph (figure 2). These correspond to all the open hamiltonian paths that can be found in the main graph, which is NP-complete problem. Each subgraph can be used by the document structurer, which, given the appropriate lexical and document resources, will realise all the assertions in that subgraph and produce a valid text.

A document structurer could in principle, in addition to consider non-connected graphs, consider all the possible subgraphs of the graph in figure 2. This would include subgraphs with (1) direct circuits, that is, multiple relations over the same pair of propositions as discussed in (Webber et al., 1999), and (2) indirect circuits, licensing texts like the one below, which expresses R1, R2 and R4:

[1*f*] With the next music day scheduled for July 21 (Saturday), noon-midnight^A:
- I'll post more details later^B.
- But this is a good time to reserve the place on your calendar^C.

3 Relation to RST

3.1 RST Properties

The properties of an RST structure defines it as a tree. They are (Mann and Thompson, 1987) :

Connectedness: except for the root, each text span in the analysis is either a minimal unit or a constituent of another schema application of the analysis.

Uniqueness: each schema application involves a different set of text spans.

Adjacency: the text spans of each schema application constitute one contiguous text spans.

Completeness: one schema application spans the entire text.

As described in the previous section, connectedness is enforced between the rhetorical assertions. In the current implementation, this happens because the non-hierarchical document structurer will only combine two (sub-)document structures together if each contains a proposition which is an argument of a rhetorical assertion to realise. Also, uniqueness is enforced because once a pair of sub-DSs has been used to realise an assertion, it cannot be re-used to realise another assertion. This enforces the no-circuit requirement described in the previous section. Adjacency is enforced by the document structurer which cannot interleave a DS inside another one. For example, given the following rhetorical input that satisfies our graph requirement:

```
R1(sat:A,nuc:B)
R2(sat:C,nuc:D)
R3(sat:E,nuc:F)
R4(sat:E,nuc:C)
R5(sat:F,nuc:B)
```

There are 6!=720 possible linear orderings for this rhetorical input, some of which are altogether invalid since they can't be grouped into a single span. For example, ACBDFE is not possible because AB and CD are interleaved. On the other hand, ABFECD is possible because it can be linearly spanned into rhetorically related propositions: ((AB)((FE)(CD))).

Finally, completeness is enforced too given that the resulting DS is a tree. It should be obvious now that the DSs currently generated are RS-like structures. However and as explained in section 1, they need not be strongly compositional. This aspect allows a certain freedom of the input with respect to RST.

3.2 non-RST Properties

The rhetorical input of the following texts respects our graph requirement and can be built into a valid DS.[1] However, their RST analysis poses a problem.

[2] You should continue taking the tablets for as long as your doctor has asked^A, *unless* you develop any problems^B, *in which case*, consult your doctor^C.

condition(sat:B,nuc:C)
unless(sat:B,nuc:A)

[3] Avoid driving^A because dizziness can occur with the first dose^B. You may also feel dizzy if you stand up quickly after lying down^C.

cause(sat:B,nuc:A)
list(nuc1:B,nuc2:C)

The rhetorical assertions for [2] cannot be built into a tree because RST, given the uniqueness principle, does not allow a span to be satellite of two spans. However, a DS can be built given this input since the DS's constraint on uniqueness applies on DS spans, not rhetorical spans.

The rhetorical assertions for [3] can be built into an RS-tree but that RS-tree does not satisfy the nuclearity principle. On the other hand, a DS can be built given this input since we do not rely on this principle for DS composition. Other examples might involve relations whose arguments are relations, or relations with a wide scope (i.e., spanning more than one proposition, such as a condition or a circumstance relation) which cannot be represented in a strongly compositional RS-tree but can be represented by a valid rhetorical graph and could be made into a text via a DS.

[1]Both examples are taken from ABPI (1997). [2] is taken from Innozide (Merck-Sharp Dohme) whilst [3] is a simplified excerpt from Hytrin BPH (Abbott).

4 Conclusion

The absence of a homomorphic requirement between the discourse structure and the document structure returns to the rhetorical input some of RST's original descriptive power it was deprived of for the purposes of NLG. In particular, the rhetorical input does not have to be built into a rhetorical structure following a strong compositionality principle, whose limitations to account for real texts has been pointed out. In particular, Knott et al (2001) have discussed the conflicts between the hierarchical discourse structure of a text and the focus relations that may exist between segments of the text.

By distinguishing between a rhetorical graph input structure and a document structure output, we avoid the difficulties associated with the definition of RST as a theory of text analysis, where the hierarchical structure of the surface text is intermingled with the non-hierarchical structure of the input message (Power et al., in press). In effect, this means that document structuring is not dependent on a particular discourse theory.

References

ABPI, editor. 1997. *Compendium of Patient Information Leaflets*. Association of British Pharmaceutical Industry.

N. Bouayad-Agha. 2001. *The Role of Document Structure in Text Generation*. Ph.D. thesis, Information Technology Research Institute (ITRI), University of Brighton. Also availaible as a Technical Report ITRI-01-24.

A. Knott, J. Oberlander, M. O'Donnell, and C. Mellish. 2001. Beyond elaboration: the interaction of relations and focus in coherent text. In T.Sanders, J.Schilperoord, and W.Spooren, editors, *Text Representation: Linguistic and Psycholinguistics Aspects*, pages 181–196. Benjamins, Amsterdam.

W.C. Mann and S.A. Thompson. 1987. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, Information Sciences Institute, University of Southern California.

D. Marcu. 1997a. From local to global coherence: A bottom-up approach to text planning. In *The Proceedings of the Fourteenth National Conference on*

Artificial Intelligence, pages 629–635, Providence, Rhode Island, July. AAAI.

D. Marcu. 1997b. *The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts*. Ph.D. thesis, Department of Computer Science, University of Toronto. Available at: http://www.isi.edu/ marcu/papers.html.

C. Mellish, A. Knott, J. Oberlander, and M. O'Donnell. 1998a. Experiments using stochastic search for text planning. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pages 98–107, Niagara-on-the-Lake, Ontario, Canada, 5-7 August.

C. Mellish, M. O'Donnell, J. Oberlander, and A. Knott. 1998b. An architecture for opportunistic text generation. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pages 28–37, Niagara-on-the-Lake, Ontario, Canada, 5-7 August.

J.D. Moore and C.L. Paris. 1994. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational linguistics*, 19(4):651–693.

R. Power, D. Scott, and N. Bouayad-Agha. (in press). Document structure. *Computational Linguistics*.

B.L. Webber, A.Knott, and A.Joshi. 1999. Multiple discourse connectives in a lexicalized grammar for discourse. In *Third International Workshop on Computational Semantics*, Tilburg, The Netherlands.

Multilingual Revision

Charles Callaway
Istituto per la Ricerca Scientifica e Tecnologica
Istituto Trentino di Cultura, Italy (ITC-IRST)
`callaway@irst.itc.it`

Abstract

Natural Language Generation has made great strides towards multilingual generation from large-scale knowledge sources. Meanwhile, current research in revision has vastly improved the quality of text that NLG systems produce. However, to-date there has been no attempt at combining revision and multilingual NLG. This paper presents research in multilingual revision, the last major pipelined NLG component to be studied from a multilingual perspective. We describe the linguistic difficulties in achieving multilingual revision, review recent work, and present an implemented framework for multilingual revision rules.

1 Introduction

The quality of initial, unimproved text produced by explanation generation systems has been notoriously poor (Lester and Porter, 1997). Recent work in revision (Dalianis and Hovy, 1993; Robin, 1994; Callaway and Lester, 1997; Harvey and Carberry, 1998; Shaw, 1998) has shown substantial progress towards qualitative and quantitative improvements in the text that explanation generators produce. By necessity, revision systems operate over deep linguistic structures rather than text strings from template generators. Indeed, the increase in variability and flexibility that deep generation systems provide is often touted as a major advantage over simpler, more easily implemented template generators (Reiter, 1995).

Multilinguality from deep linguistic representations (Paris et al., 1995; Stede, 1996; Bateman and Sharoff, 1998; Scott, 1999; Kruijff et al., 2000) is generally considered to be one of the advantages that deep generation systems possess over templates (although this depends heavily on the definitions of deep and template methods, such as in (Deemter et al., 1999)). By applying multilingual lexica and grammars to a single initial knowledge base, multilingual generators hope to leverage reusable components to produce texts in multiple languages with substantially less work than implementing an equivalent number of monolingual template or deep generators. But to be used effectively and efficiently in a multilingual generation system, these reusable components must be designed from the start for that purpose. Similarly, one of the main motivations for multilingual revision is efficiency: A single formalism for revision rules can greatly increase the amount of resource sharing in a manner analogous to that of grammars.

Just as in monolingual explanation generation, unrevised multilingual text is in general quite undesirable. And while multilingual components have been created for sentence planning (Bateman and Sharoff, 1998), surface realization (Netzer and Elhadad, 1999) and lexical choice (Stede, 1996), no attempt has been made to combine research in revision and multilingual NLG.

This paper presents research in multilingual revision, the last major pipelined NLG component to be studied from a multilingual perspective. We first present linguistic reasons for the difficulty of discovering "revision rules" in both monolingual and multilingual contexts. Next, we describe recent work in revision and then analyze the elements of revision rules to determine how each element affects the revision process from both a monolingual and multilingual perspective. Finally, we synthesize a revision rule model that performs multilingual revisions under a common formalism in an implemented NLG system.

2 "Unrevising" a Corpus

Text generation systems are notorious for producing correct but low quality text. In contrast, human writers directly produce fluent text even in initial drafts. Because of this, it is very difficult to find a version of a naturally occurring text similar to the types of protosentences produced by systems today. (Meteer, 1990) initially explored this problem by having writers of scientific articles generate paraphrases in order to create a corpus for studying revision rules. The lack of such corpora is a disadvantage to implementers of revision systems because they lack the original source materials with which to create revision rules that would then allow them to achieve results comparable to the original, polished texts.

For example, consider the four text fragments in Figure 1, where (a) represents an excerpt from an original passage in Spanish, (b) represents its "unrevised" version, (c) represents the translation of (a) into English, and (d) is the "unrevision" of (c). While (a) and (c) are easily found in available corpora, the sentence structures in (b) and (d) cannot be found in either existing corpora or authors' drafts. And yet these are precisely the types of texts needed to discover how simple "protosentences" are combined into larger clause structures.

Thus, one of the tasks that creators of revision systems must do is to collect example corpora from their generation domain and "unrevise" them to determine what types of revisions were performed by the original authors of the domain texts. The unrevised sentences should correspond to the types of protosentences produced by the initial discourse and sentence planners. The result of gathering examples from a set of corpora is a set of revision rules applicable to that particular domain which can then be used by a revision component to refine and polish the final text produced by a generation system.

In a multilingual context, this corpus analysis must be repeated for each language, because different languages typically have different revision rules (even when the content is the same, as in the translations in Figure 1 (a) and (c), a revision system may end up reorganizing the text in a different manner because different languages and cultures may have different modes of presentation). Thus a system for multilingual revision must be able to handle disparate sets of revision rules as needed, or else resort to having multiple revision components, with the corresponding decrease in efficiency and increase in effort that such an architecture would bring.

A more efficient architectural approach would be to make different multilingual revision decisions for a set of languages by only replacing sets of revision rules. To be effective, the revision module must be orchestrated in such a way that all of the decision-making information can be determined by the revision rules themselves. Thus, a detailed feature analysis is necessary to determine the structure and variability of typical revision rules. In addition, all of the information necessary to choose between different revision rules must be closely tied to the incoming rhetorical structure and protosentences that comprise the sentence plan.

3 Aspects of Revision Operations

Although much research has described the architectural and linguistic aspects of revision, relatively little has been done to describe a *feature-based model* of revision upon which revision decisions can be based. However, most research describing implemented revision systems provides insight into the high-level features that were found necessary for particular projects or domains.

Dalianis' work (Dalianis and Hovy, 1993) concentrated on the effect that a rule would have on the text in terms of carrying it out procedurally. His revision rules were the result of evaluating human revisions on a set of standardized textual propositions.

Pasé por alto el comentario. Me daba un poco de
vergüenza explicar que hacía todo esto porque
había conocido una vez a una chica pelirroja.
Después le dije que sí, que pensaba aprender teatro.
En realidad yo aborrecía a los actores. Eran
demasiado extravertidos para mi gusto, y me
impresionaban como gente que siempre se estaba
saludando y abrazando y eran amigos de todo el
mundo. No soporto a la gente que es amiga de todo
el mundo, como los animadores de televisión.

(a)

I skipped over her comment. I was a bit ashamed
to explain that I had done all this because I had
once met a redheaded girl. Later I told her that
yes, I was thinking about studying theater, although
in fact, I hated actors. They were too extroverted
for my taste, and seemed like people who were
always greeting and hugging each other as if
they were friends with the whole world. I don't
put up with someone who is friends with the
whole world, like one of those television hosts.

(c)

(1) Pasé por alto el comentario.
(2) Me daba un poco de vergüenza explicar que
hacía todo esto. ("porque")
(3) Había conocido una vez a una chica pelirroja.
("después")
(4) Le dije que sí, que pensaba aprender teatro.
(5) En realidad, yo aborrecía a los actores.
(6) Eran demasiado extravertidos para mi gusto. ("y")
(7) Me impresionaban como gente que siempre
se estaba saludando ("y")
(8) Se estaba abrazando ("y")
(9) Eran amigos de todo el mundo.
(10) No soporto a la gente ("que")
(11) La gente es amiga de todo el mundo. ("como")
(12) Los animadores de televisión ["son amigas..."]

(b)

(1) I skipped over her comment.
(2) I was a bit ashamed to explain I had done all this.
("because")
(3) I had once met a red-headed girl. ("later")
(4) I told her that yes, I was thinking about studying theater.
("although")
(5) In fact, I hated actors.
(6) They were too extroverted for my taste. ("and")
(7) They seemed like people who were always
greeting each other. ("and")
(8) They were always hugging each other. ("as if ")
(9) They were friends with the whole world.
(10) I don't put up with someone. ("who")
(11) The person is friends with the whole world. ("like")
(12) One of those television hosts ["is friends with..."]

(d)

Figure 1: "Unrevised" Spanish

- *Aggregation*: Operators which merge two previously separate clauses into one.
- *Ordering*: Operators that reverse the external ordering of clauses (migration) or their internal ordering (linear precedence).
- *Casting*: Operators that alter or enforce the regularity of syntactic structures over multiple clauses.
- *Parsimony*: Operators that prefer fewer overall words in a clause or other numeric quantities such as depth of prepositional phrase or relative clause embeddings.

Shaw's CASPER system of revision operators (Shaw, 1998) focuses on the syntactic dependency notion of hypotactic vs. paratactic operators. CASPER functions in the domain of medical report generation where Shaw noted that clause aggregations could frequently be classified on the basis of dependency. In both cases, the redundant element is deleted from one of the clauses.

- *Hypotactic*: Operators which take two sentences, a base and a modifier, convert the modifier into a dependent clause, and then attach it to the base sentence.
- *Paratactic*: Operators which attach two sentences at the same dependency level, such as with *and* or *or* coordination.

By far the most explicitly organized and classified set of revision operators is described in the work on STREAK (Robin, 1994, Appendix A), a system for writing summary descriptions of basketball games in English. Motivated by (but not implemented using) a Tree Adjoining Grammar approach, Robin makes the following classifications of revision operators:

- *Monotonic*: Operators which leave the base syntactic structure intact and result only in the attachment of a new phrase or clause to an unmodified existing structure. Examples include **Adjoin**, **Absorb**, **Conjoin**, and **Append** operators.

- *Nonmonotonic*: Operators which break up the base syntactic structure in various ways before attaching a new phrase or clause. Examples include **Recast**, **Adjunctization**, **Nominalization**, **Demotion**, and **Promotion** operators.

- *Side Transformations*: Operators that reduce redundant lexemes left over from previous revision operations. Examples include **Reference Adjustment**, **Argument Control**, **Ellipsis**, **Scope Marking**, **Ordering Adjustment**, and **Lexical Adjustment** operators.

Robin lists 18 distinct types of **adjoin** operators alone, organized according to the syntactic type of the phrase to be adjoined and the syntactic type and position of its attachment location. To illustrate, two of these subclasses of the adjoin operator are shown here:

- *Adjoin Relative Clause to Bottom-Level Nominal*: Attaches a relative clause to an immediately preceding noun phrase as in "to power them to a win over [(the Cavs), (who lost again)]".

- *Adjoin Relative Clause to Top-Level Nominal*: Attaches a relative clause to a preceding noun phrase which already has postmodifiers as in "to power them to [(a win over the Cavs), (that extended their streak)]".

However, there are a number of other classifications of revision operators not covered by these three approaches. In addition, these systems are not multilingual in nature and their corpora analyses covered only revisions in the English language. In order to fully understand the revision process both linguistically and computationally, as well as to ensure that this understanding is consistent across languages, it is important to discover and classify as many aspects of revision operators as possible. Some of the additional aspects we have found in our efforts to build a multilingual revision component are as follows:

- *Rhetorical Type*: Most revision systems assume some underlying theory of discourse structure, such as RST (Mann and Thompson, 1987). These theories define particular rhetorical types such as *greeting* or *persuade* which are used by revision systems to provide additional constraints when selecting revision operators. How these constraints are affected by multilingual text is unknown.

- *Perspective*: While revision systems seem identical in purpose, there are actually great differences in what they expect to accomplish. For example, Dalianis and Hovy's rules attempt to mimic human revision processes, Robin's STREAK system attempts to build a single sentence, Shaw's CASPER system focuses on eliminating redundancy, while other systems try to increase syntactic variety.

- *Syntactic structure*: Most revision systems start from sequences of protosentences and change a subset of those protosentences into different clauses or phrases. However, a complete set of target syntactic structures (especially identifying which of those structures overlap with the syntactic structures of other languages) has not yet been identified.

- *Attachment position*: When protosentences are converted into dependent circumstantial clauses (*e.g.*, when-clauses), a revision operator must choose to place it in front or at the end of a sentence. These operators must take into account whether a previous revision has already occupied one of those slots as well as whether a particular language allows a similar range of syntactic possibilities.

- *Depth of representation*: Robin's STREAK system explicitly represented multiple levels of syntax, semantics, and pragmatics. More recent systems have shied away from this approach and favored shallower representations to increase efficiency.

- *Scope*: Revision operators can be very local and examine only adjacent protosentences or, at the expense of efficiency, examine protosentences slightly farther away (either to actually use them in aggregating or merely for additional context when making a more local decision).

- *New lexicalizations*: Most sentence planners do not produce many of the discourse-level elements found in polished texts because they are not needed when generating protosentences. For instance, discourse markers (Vander Linden and Martin, 1995; Grote, 1998) are frequently used to show the relationships between individual clauses. If appropriate information were available, it would be possible for revision operators to add discourse markers as they perform clause aggregation.

These factors are important when building a revision system that is scalable in terms of size, genre and language.

4 Motivation *vs.* Action

Analyzing the structure of revision operators in this manner does not imply that any particular architecture is preferred over another. However, much like the STRIPS architecture (Fikes and Nilsson, 1972), our analysis of revision rules in a multilingual environment has shown that every revision operator can be broken down into two parts describing *when* a rule should be fired (motivation, expressed as triggering rules), and if it is fired, *what* are the effects of that rule on the original protosentences (action, expressed as a target syntactic modification).

- *Motivations*: The parts of a revision rule which deal with whether the rule is applicable and which differentiate it with respect to other rules. Aspects which are helpful in deciding applicability include *rhetorical type*, *perspective*, *syntactic structure*, *depth of representation*, *scope*, and if discourse markers are expected to be added, those features of the input which are salient.

- *Actions*: The parts of a revision rule that alter either the internal syntactic structure of clauses as they are aggregated or the rhetorical relationship(s) between multiple clauses. Aspects which are useful include *monotonicity*, *dependency*, *effect*, *perspective*, *syntactic structure*, and *attachment position*.

This division is apparent despite the language of the text being revised. For example, because languages have syntactic structure and at the lowest level revisions affect that syntactic structure,[1] the decision to alter that structure implies that the revision rule knows which syntactic category it is going to change it to. However, the particular syntactic category might be different given a different language. Given exactly similar circumstances, an English revision rule might prefer to change a protosentence into a prepositional phrase while a Spanish revision rule might prefer a relative clause. In addition, some syntactic options are available in certain languages but not in others (Netzer and Elhadad, 1999).

There are frequent similarities between languages, however. For example, our corpora analyses have shown that coordination with "and" usually occurs in similar situations. Also, the revision rules we have devised for English and Spanish in our domain almost always share identical *motivations* even if they differ in their *actions*. Given the overall structural similarity of revision rules despite their differences in details, the goal of designing a system capable of efficient multilingual revision is then to devise a single architectural component capable of carrying out revision operations by swapping out sets of revision rules rather than creating a separate revision component for each distinct language.

In our experience, different languages have similar sets of motivations for when to apply revision rules and similar sets of actions that carry them out. However, since the *mapping* from the set of initial structures (which drive the analysis of the motivation components of the revision rules) provided by the sentence planner to the set of actions which produce the final structures is language-specific, it is appropriate to apply a functional

[1]While revisions do ultimately alter morphological structure, they do so only indirectly. No revision rule to our knowledge either considers or implements decisions based on morphological information.

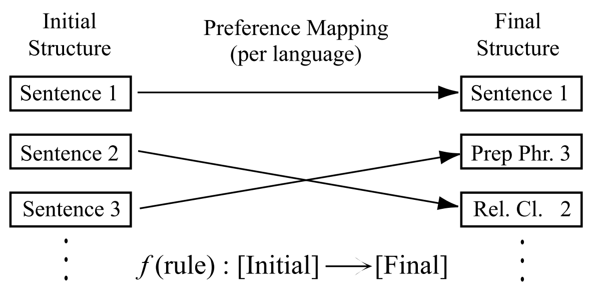


Figure 2: Revision Functions

model. Such a model (Figure 2) provides a separate function for each language which creates a set of revision rules by mapping from a set of motivations to a set of actions.

A multilingual revision architecture then could have a single set of language-neutral motivation criteria, a single set of language-neutral action effects, and a mapping function for each language desired (Figure 3). When the system is directed to switch from generating text for one language to another, the revision component needs only to switch in a new mapping function rather than using an entirely new revision component designed solely for the new language. Another benefit is that once sets of motivations and actions are encoded, it is relatively easy to adjust the effects of the revision module for different genres and styles.

5 Implementation

We started with an existing pipelined, multi-paragraph multilingual NLG system, STORYBOOK (Callaway, 2000; Callaway and Lester, 2002), that takes protosentences and revises them into text. While the major pipelined modules (discourse planner, sentence planner and surface realizer) were already capable of multilingual generation, the revision component, REVISOR (Callaway and Lester, 1997) only worked for English.

Our first step was to reorganize the English revision component following the architecture previously described. After analyzing the existing revision rules, we came up with a common set of 54 motivational triggers, 16 syntactic transformation actions, and a mapping between them that simulated the existing revision rule set. We then restructured the rule determination and application mechanisms before verifying that indeed the new

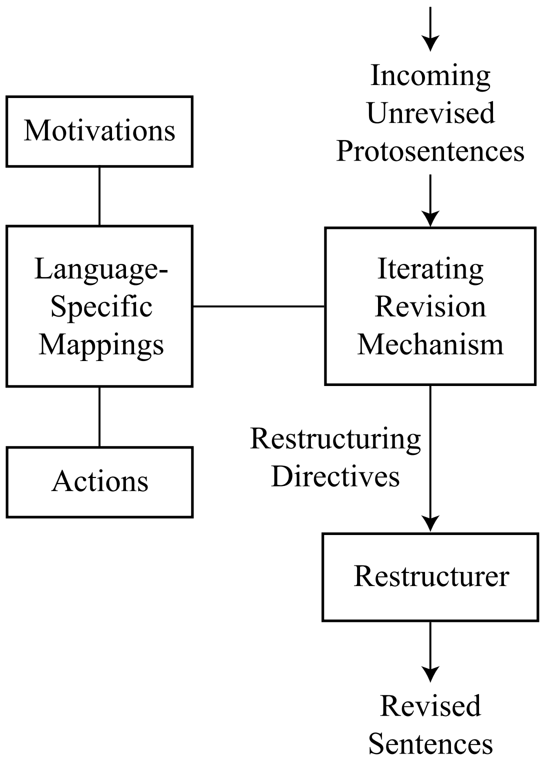


Figure 3: Abstract Revision Architecture

revision component made substantially the same revisions to the text as had the original revision component.

Next, we analyzed our translated Spanish corpus using the "unrevising" strategy described in Section 2. This yielded an additional 7 motivational triggers and 2 syntactic transformation actions for the Spanish corpus that were not accounted for in the original set of English revision rules. Afterwards, we created a mapping function from the appropriate motivational triggers found in both the English and Spanish set to the syntactic actions which we had found in the Spanish corpus.

For example, consider the sentences in Figure 4. REVISOR was initially capable of generating these simple protosentences in both English and Spanish, although initially only the English version worked well with revision, as it was keyed to look for specific information only present in the English output. We rewrote the English revision rules instead to look for more generic tags from the sentence planner indicating a particular protosentence had an intention such as `IDENTITY`

```
I don't put up with a person. [+animate-relative-clause = "who"]
The person is friends with the whole world. [+comparison = "like"]
Television hosts are friends with the whole world.

I don't put up with a person who is friends with the whole world,
like television hosts.

No soporto a la gente. [+animate-relative-clause = "que"]
La gente es amiga de todo el mundo. [+comparison = "como"]
Los animadores del televisio'n son amigas de todo el mundo.

No soporto a la gente que es amiga de todo el mundo, como los
animadores del televisio'n.
```

Figure 4: An example

or `DESCRIPTION`. We then extracted the syntactic aggregation operations, such as rules for generating relative clauses, from the various revision rules. Next, we wrote the mappings which combined the two (Figure 2), and checked to ensure that the original paragraph was regenerated. Finally, by substituting the appropriate mapping, we were able to also generate the revised Spanish version (Figure 4).

The result was an efficient revision system (execution time measured in tens of milliseconds) that produced the same or very similar revised paragraphs as the original English revision component as well as performing appropriate revisions to the Spanish text. This resulted in a substantial improvement in the amount of time required to create a traditional, standalone revision component for Spanish from scratch.

6 Conclusions

Efficient multilingual revision is possible within a single framework given a detailed analysis not only of a domain and its corpus and the types of revision operations conducted in each language, but by specifying the substructures of revision rules themselves. By isolating the differences in revision rules inherent in particular languages, we can increase the extent of language-neutral architectures and decrease the amount of work required to implement multilinguality in formerly monolingual systems.

A significant amount of the work involved in creating a multilingual revision system lies in conducting corpora analyses. One of the many problems faced by creaters of revision systems is that a large amount of text must be examined before one can be confident that a sufficient number of revision rules has been uncovered. And because NLG systems to date are not capable of generating large scale texts, it is extremely difficult to test theories of revision rules. Having modular revision systems that can be easily altered for new languages, styles and genres will improve the quality of texts produced by NLG systems.

7 Acknowledgements

This work was supported by the PEACH and TICCA projects, funded by the Autonomous Province of Trento.

References

J. Bateman and S. Sharoff. 1998. Multilingual grammars and multilingual lexicons for multilingual text generation. In *ECAI Workshop on Multilinguality in the Lexicon-II*, pages 1–8, Brighton, UK.

Charles Callaway. 2000. *Narrative Prose Generation*. Ph.D. thesis, North Carolina State University, Raleigh, NC.

Charles B. Callaway and James C. Lester. 1997. Dynamically improving explanations: A revision-based approach to explanation generation. In *Proceedings of the Fifteenth International Joint Conference* *on Artificial Intelligence*, pages 952–58, Nagoya, Japan.

Charles B. Callaway and James C. Lester. 2002. Narrative prose generation. *Artificial Intelligence*, 139(2):213–252, August.

Hercules Dalianis and Eduard Hovy. 1993. Aggregation in natural language generation. In *Proceedings of the Fourth European Workshop on Natural Language Generation*, Pisa, Italy.

Kees van Deemter, Emiel Krahmer, and Mariët Theune. 1999. Plan-based vs. template-based nlg: A false opposition? In *Proceedings of the KI-99 Workshop Between Templates and Free Choice in Natural Language Generation*, Bonn, Germany.

R. E. Fikes and N. J. Nilsson. 1972. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208.

Brigitte Grote. 1998. Representing temporal discourse markers for generation purposes. In *Proceedings of the Discourse Relations and Discourse Markers Workshop*, pages 22–28, Montréal, Canada.

Terrence Harvey and Sandra Carberry. 1998. Integrating text plans for conciseness and coherence. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 512–518, August.

Geert-Jan Kruijff, Elke Teich, John Bateman, Ivana Kruijff-Korbayová, Hana Skoumalová, Serge Sharoff, Lena Sokolova, Tony Hartley, Kamenka Staykova, and Jiří Hana. 2000. Multilinguality in a text generation system for 3 Slavic languages. In *COLING–2000: Proceedings of the 18th International Conference on Computational Linguistics*, Saarbruecken, Germany.

James C. Lester and Bruce W. Porter. 1997. Developing and empirically evaluating robust explanation generators: The KNIGHT experiments. *Computational Linguistics*, 23(1):65–101.

William C. Mann and Sandra A. Thompson. 1987. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, USC/Information Sciences Institute, Marina del Rey, CA, June.

Marie W. Meteer. 1990. *The Generation Gap: The Problem of Expressibility in Text Planning*. Ph.D. thesis, University of MA, February.

Yael Dahan Netzer and Michael Elhadad. 1999. Bilingual Hebrew-English generation of possessives and partitives: Raising the input abstraction level. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics*, pages 144–151, College Park, Maryland, June.

Cécile L. Paris, Keith Vander Linden, Markus Fischer, Anthony Hartley, Lyn Pemberton, Richard Power, and Donia Scott. 1995. A support tool for writing multilingual instructions. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1398–1404, Montréal, Canada.

Ehud Reiter. 1995. NLG vs. templates. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, Leiden, The Netherlands.

Jacques Robin. 1994. *Revision-Based Generation of Natural Language Summaries Providing Historical Background*. Ph.D. thesis, Columbia University, December.

Donia R. Scott. 1999. The multilingual generation game: Authoring fluent texts in unfamiliar languages. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden.

James Shaw. 1998. Clause aggregation using linguistic knowledge. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pages 138–147, Niagara-on-the-Lake, Canada.

Manfred Stede. 1996. *Lexical Semantics and Knowledge Representation in Multilingual Sentence Generation*. Ph.D. thesis, University of Toronto, Toronto, Ontario.

Keith Vander Linden and James H. Martin. 1995. Expressing rhetorical relations in instructional text: A case study of the purpose relation. *Computational Liguistics*, 21(1):29–57.

Learning to Order Facts for Discourse Planning in Natural Language Generation

Aggeliki Dimitromanolaki
Department of Information &
Communication Systems Engineering
University of the Aegean
Institute of Informatics &
Telecommunications
NCSR “Demokritos”
15310, Ag.Paraskeui, Greece
adimit@iit.demokritos.gr

Ion Androutsopoulos
Department of Informatics
Athens University of Economics &
Business
Patission 76, 10434, Athens, Greece
ion@aueb.gr

Abstract

This paper presents a machine learning approach to discourse planning in natural language generation. More specifically, we address the problem of learning the most natural ordering of facts in discourse plans for a specific domain. We discuss our methodology and how it was instantiated using two different machine learning algorithms. A quantitative evaluation performed in the domain of museum exhibit descriptions indicates that our approach performs significantly better than manually constructed ordering rules. Being retrainable, the resulting planners can be ported easily to other similar domains, without requiring language technology expertise.

1 Introduction

Along the lines of Reiter and Dale (2000), we view natural language generation (NLG) as consisting of six tasks: content determination, discourse planning, aggregation, lexicalization, referring expression generation, and linguistic realization. This paper is concerned with the second task, i.e., *discourse planning*. Discourse planning determines the ordering and rhetorical relations of the logical messages, hereafter called *facts*, that the generated document is intended to convey. Most existing approaches to discourse planning are based on either rhetorical structure theory (RST) (Mann and Thompson, 1988; Hovy, 1993) or schemata (McKeown, 1985). In both cases, the rules that determine the order and the rhetorical relations are typically written by hand. This is a time-consuming process, which requires domain and linguistic expertise, and has to be repeated whenever the system is ported to a new domain; see also Rambow (1990).

This paper presents a machine learning (ML) approach to the subtask of discourse planning that attempts to find the most natural ordering of facts in each generated document. Our approach was motivated by experience obtained from the M-PIRO project (Androutsopoulos et al., 2001). Building upon ILEX (O'Donnell et al., 2001), M-PIRO is developing technology that allows personalized descriptions of museum exhibits to be generated in several languages, starting from symbolic, language-independent information stored in a database, and small fragments of text (Isard et al., 2003). One of M-PIRO’s most ambitious goals is to develop authoring tools that will allow domain experts, e.g., museum curators, with no language technology expertise to configure the system for new application domains. While this goal has largely been achieved for resources such as the domain-dependent parts of the ontology, or domain-dependent settings that affect content selection, lexicalization, and referring expression generation (Androutsopoulos et al., 2002), designing tools that will allow domain experts to edit discourse planning rules has proven difficult. In contrast, domain experts, in our case museum curators, were happy to reorder the clauses of sample generated texts, thus indicating the preferred orderings of the facts in the corresponding discourse plans. We have, therefore, opted for a machine learning approach that allows fact-ordering rules to be captured automatically from sets of manually reordered facts. We view this approach as a first step towards learning richer discourse plans, which apart from ordering information will also include rhetorical relations, although the experience from M-PIRO indicates that even just ordering the facts in a natural way can lead to quite acceptable texts. Being automatically retrainable, the planners that our approach produces can be easily ported to other similar domains, e.g., descriptions of products for e-commerce catalogues, provided that samples of ideal fact orderings can be made available.

Our method introduces a new representation of the fact-ordering task, and employs supervised learning algorithms. It is assumed that the number of facts to be conveyed by each generated document, in effect the desired length of the generated texts, has been fixed to a particular value; i.e., all the documents contain the same number of facts. In ILEX and M-PIRO, this number is provided by the user model. Furthermore, it is assumed that a content determination module is available, which selects the particular facts to be conveyed by each document. Our method consists of a sequence of stages, the number of stages being equal to the number of facts to be conveyed by each document. Each stage is responsible for the selection of the fact to be placed at the corresponding position in the resulting document. In our experiments, we set the number of facts per document to six, which seems to be an appropriate value for our particular domain and an average adult user, but this number could vary depending on the application and user type. Two learning algorithms, decision trees (Quinlan, 1993) and instance-based learning (Aha and Kibler, 1991), were explored. The results are compared against two baselines: a simple hand-crafted planner, which always assigns a predefined order, and the majority scheme. The latter selects, among the facts that are available at each position, the fact that occurred most frequently at that position in the training data. Overall, the results indicate that with either of the two learning algorithms our method significantly outperforms both of the baselines, and that there is no significant difference in the performance of the two learning algorithms.

The remainder of this paper is organized as follows. Section 2 presents previous learning approaches to NLG, and discusses their relevance to the work presented here. Section 3 describes our learning approach, including issues such as data representation and system architecture. Section 4 discusses our experiments and their results. Section 5 concludes and highlights plans for future work.

2 Previous work

In recent years, ML approaches have been introduced to NLG to address problems such as the construction and maintenance of domain and language resources, which is a time-consuming process in systems that use hand-crafted rules.[1] To the best of our knowledge, only two of these approaches (Duboue and McKeown, 2001; Duboue and McKeown, 2002) consider discourse planning.

Duboue and McKeown (2001) present an unsupervised ML algorithm based on pattern matching and clustering, which is used to learn ordering constraints among facts. The same authors have also used evolutionary algorithms to learn the tree representation of a planner (Duboue and McKeown, 2002). These works are similar to ours in that we also address the problem of ordering facts. However, Duboue

[1] For an extensive bibliography on statistical and machine learning approaches to NLG, see: http://www.iit.demokritos.gr/~adimit/bibliography.html.

and McKeown follow the lines of schema-based planning, where content determination is not an independent stage, but is interleaved with discourse planning. This means that the discourse planner has the overall control of content determination, and cannot handle inputs from an independent content determination module. In contrast, our method can be used with any content determination mechanism that returns a fixed number of facts. This has the benefit that alternative content determination modules can be used without affecting the discourse planner. Moreover, while Duboue and McKeown (2002) learn a tree structure representing the best sequence of facts, our method directly manipulates facts.

Mellish et al. (1998) also experiment with genetic algorithms to find the optimal RST tree, which is then mapped to the corresponding sequence of facts. Karamanis and Manurung (2002) use a similar approach that employs constraints from Centering Theory in the genetic search. However, these approaches do not involve any learning: the genetic search is repeated every time the text planner is invoked, i.e., for each new document. In contrast, our method induces a single discourse planner from the training data, which is then used to order any set of facts provided by the content determinator.

ML approaches to NLG have also been used in syntactic and lexical realization (Langkilde and Knight, 1998; Bangalore and Rambow, 2000; Ratnaparkhi, 2000; Varges and Mellish, 2001; Shaw and Hatzivassiloglou, 1999; Malouf 2000), as well as in sentence planning tasks (Walker et al., 2001; Poesio et al., 2000). In the context of spoken dialogue systems, learning techniques have been used to select among different templates (Oh and Rudnicky, 2000; Walker, 2000). These approaches, however, are not directly relevant to discourse planning.

The problem of ordering semantic units has also been addressed in the context of summarization. Kan and McKeown (2002) use an n-gram model to infer ordering constraints between facts, while Barzilay et al. (2002) manually identify constraints on ordering, using a corpus of ordering preferences among subjects and clustering techniques that identify commonalities among these preferences. The approach presented here, instead of identifying ordering constraints, "learns" the overall ordering of the input facts.

3 Learning to order facts

In our approach, the discourse planner is trained on manually ordered sequences of facts of a fixed length. Once trained, it is able to determine what it considers to be the most natural ordering of any set of facts, as output by a content determination module, provided that the cardinality of the set is the same as the length of the training sequences. This section describes our approach in more detail, starting from the required data and the pre-processing that they undergo.

3.1 Data and pre-processing

Our data was derived from the database of M-PIRO. This database currently contains information about 50 museum exhibits, each of which is associated with a large number of facts. For example, the left column of Table 1 shows the database facts associated with the entity *exhibit9*. Each generated document is intended to describe a museum exhibit. As already mentioned, in our experiments the number of facts to be conveyed by each document was set to six. That is, when asked to describe *exhibit9*, the content determination module would choose six of the facts in the left column of Table 1, possibly depending on user modeling information, such as the interests and backgrounds of the users, or information indicating which facts have already been conveyed to the users. We did not use a particular content determination module, because we wanted the discourse planner to be independent from the content determination process. Our goal was to be able to order any set of six facts that could be provided as input by an arbitrary content determination module.

Database facts	Selected facts (input to discourse planner)
subclass(EXHIBIT9,RHYTON)	
current-location(EXHIBIT9,MUS-DU-PETIT-PALAIS)	
original-location(EXHIBIT9,ATTICA)	
potter-is(EXHIBIT9,SOTADES)	
exhibit-characteristics(EXHIBIT9,ENTITY-1796)	f1: subclass(EXHIBIT9,RHYTON)
painted-by(EXHIBIT9,PAINTER-OF-SOTADES)	f2: current-location(EXHIBIT9,MUS-DUPETITPALAIS)
creation-time(EXHIBIT9,DATE-1767)	f3: original-location(EXHIBIT9,ATTICA)
creation-period(EXHIBIT9,CLASSICAL-PERIOD)	f4: painted-by(EXHIBIT9,PAINTER-OF-SOTADES)
painting-technique-used(EXHIBIT9,RED-FIG-TECHN)	f5: creation-time(EXHIBIT9,DATE-1767)
exhibit-depicts(EXHIBIT9,ENTITY-1786)	f6: creation-period(EXHIBIT9,CLASSICAL-PERIOD)
opposite-technique(RED-FIG-TECHN,BLACK-FIG-TEC)	
technique-description(RED-FIG-TECHN,ENTITY-2474)	
person-information(SOTADES,ENTITY-2739)	
museum-country(MUS-DU-PETIT-PALAIS,FRANCE)	
period-story(CLASSICAL-PERIOD,STORY-NODE4019)	

Table 1: Database facts and facts selected as input to the discourse planner

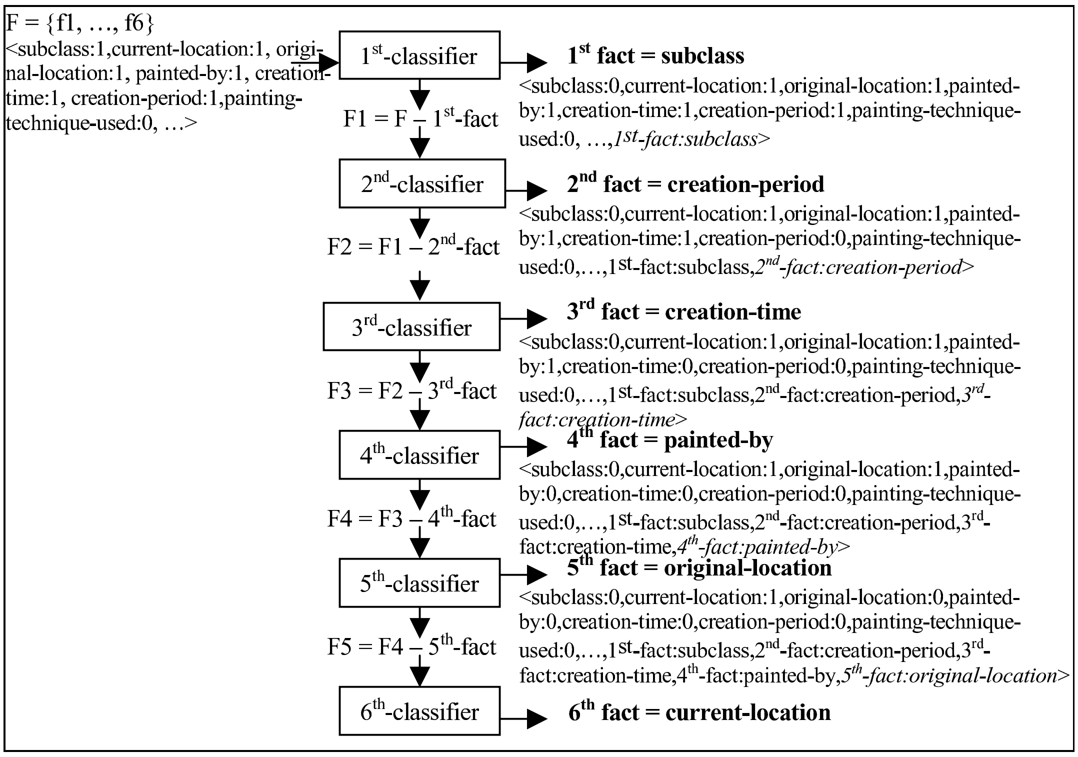

F = {f1, …, f6}
<subclass:1,current-location:1, original-location:1, painted-by:1, creation-time:1, creation-period:1,painting-technique-used:0, …>
1st-classifier
1st fact = subclass
<subclass:0,current-location:1,original-location:1,painted-by:1,creation-time:1,creation-period:1,painting-technique-used:0, …,1st-fact:subclass>
F1 = F – 1st-fact
2nd-classifier
2nd fact = creation-period
<subclass:0,current-location:1,original-location:1,painted-by:1,creation-time:1,creation-period:0,painting-technique-used:0,…,1st-fact:subclass,2nd-fact:creation-period>
F2 = F1 – 2nd-fact
3rd-classifier
3rd fact = creation-time
<subclass:0,current-location:1,original-location:1,painted-by:1,creation-time:0,creation-period:0,painting-technique-used:0,…,1st-fact:subclass,2nd-fact:creation-period,3rd-fact:creation-time>
F3 = F2 – 3rd-fact
4th-classifier
4th fact = painted-by
<subclass:0,current-location:1,original-location:1,painted-by:0,creation-time:0,creation-period:0,painting-technique-used:0,…,1st-fact:subclass,2nd-fact:creation-period,3rd-fact:creation-time,4th-fact:painted-by>
F4 = F3 – 4th-fact
5th-classifier
5th fact = original-location
<subclass:0,current-location:1,original-location:0,painted-by:0,creation-time:0,creation-period:0,painting-technique-used:0,…,1st-fact:subclass,2nd-fact:creation-period,3rd-fact:creation-time,4th-fact:painted-by,5th-fact:original-location>
F5 = F4 – 5th-fact
6th-classifier
6th fact = current-location

Figure 1: Architecture diagram

In order to create the dataset of our experiments, we used a program that yields all the possible combinations of six facts for each exhibit. The right column of Table 1 shows an example set of six facts, which can be used as input to the discourse planner. Many combinations, however, looked unreasonable in our domain; e.g., combinations that do not include the *subclass* fact (descriptions in the museum domain must always inform the reader about the type of the exhibit), or combinations that include facts providing background information about an entity that is not present in the discourse (for instance, combinations that include *opposite-technique* but not *painting-technique-used* in Table 1). A refinement operation was performed manually to discard such combinations. We note that in real-life applications, the combinations would be obtained by calling several times a content determination module; hence, no refinement operation would be necessary, as the content determination module would, presumably, never return unreasonable combinations of facts.

After the refinement operation, 880 combinations of 6 facts were left. The facts of each set were manually assigned an order, to reflect what a domain expert considered to be the most natural ordering of the corresponding clauses in the generated texts. Each one of the 880 sets was then used as an instance in the learning algorithms, as will be explained in the following section.

3.2 Instance representation and planner architecture

Figure 1 shows the discourse planning architecture that our approach adopts, along with an example of inputs and outputs at each stage. We decompose the fact-ordering task into six multi-class classification problems. Each of the six classifiers selects the fact to be placed at the corresponding position. Each input set of six facts is represented as a vector in a multi-dimensional space, where dimensions correspond to values of attributes. 42 binary attributes, representing the fact types of the domain, were used. The vector at the top left corner of Figure 1 represents the set of six facts of the right column of Table 1. Each attribute shows whether a particular fact type exists in the input (e.g., creation-period:1) or not (e.g., painting-technique-used:0). Classifiers 2–6 have additional attributes representing the fact types that have already been selected for positions 1–5. More specifically, as shown in Figure 1, the attribute *1st-fact* is added from the 2nd classifier onwards, the attribute *2nd-fact* is added from the 3rd classifier onwards, and so forth. Therefore, the classifiers make their decisions based on the fact types that are present in their inputs (set of remaining facts to be ordered) and the fact types that have been selected at the previous positions. We assume that it is not possible to have more than one fact of the same type in the input set of facts because this is the case in the M-PIRO domain (e.g., we cannot have two facts of type *creation-period*) as well as in other similar domains. In a more general case, however, one could differentiate between facts of the same type, by enriching, for instance, the attributes, so as to represent information about the entities related with each fact, or by adding new attributes.

The output of each classifier is the class value representing the fact type that has been selected for the corresponding position. In the example of Figure 1, the classifiers select the following order: *subclass, creation-period, creation-time, painted-by, original-location, current-location*. As shown in Figure 1, the sixth classifier has no substantial role, since there is only one fact left in the input, and, consequently, this fact will be placed at the sixth position.

In a similar manner, a sequence of n classifiers can be used when each document is to convey n, rather than 6, facts. A limitation of this approach is that it cannot be used when n varies across the documents. However, this is not a problem in M-PIRO, where n, in effect the length of the documents, is fixed for each user type: if there are t user types, we train t different document planners, one for each user type; each planner is a sequence of n_i classifiers, where n_i is the value of n for the corresponding user type ($i = 1, \ldots, t$).

4 Experiments and results

In order to evaluate our approach, we performed four experiments. The first experiment was conducted using the majority scheme, where each classifier selects among the available classes (i.e., among the facts that are present in the input set and have not been selected by the previous classifiers) the class (i.e., fact) that was most frequent in its training data. However, this scheme is too primitive, and could not be seen as a safe benchmark for our experiments. For this reason, we constructed a simple planner, hereafter *base planner*, which always assigns a predefined fixed order defined in collaboration with a museum expert; e.g., *subclass* should always be placed before *creation-period*, *creation-period* should always be placed before *creation-time*, etc. The base planner was used as a second baseline. In this way, we had a safer benchmark for the performance of the learning schemes. In the two remaining experiments we used instance-based and decision-tree learning. More specifically, we experimented with the k-nearest neighbour algorithm (Aha and Kibler, 1991), with $k = 1$, and the C4.5 algorithm (Quinlan, 1993). All the experiments were performed using the machine learning software of WEKA (Witten and Frank, 1999).

Figure 2 presents the accuracy scores of each of the six classifiers, for each learning scheme. The results were obtained using *10-fold cross-validation*. That is, the dataset (880 vectors) was divided into ten disjoint parts (folds), and each experiment was repeated 10 times. Each time, a different part was used for testing, and the remaining 9 parts were used for training. The dataset was *stratified*, i.e. the class distribution in each fold was approximately the same as in the full dataset. The reported scores are averaged over the 10 iterations. Accuracy measures the percentage of correct selections at each classifier (position) compared to the selections made by the human annotator. All schemes have 100% accuracy at the selection of the 1st and 6th fact. This happens because the first classifier always selects the fact *subclass*, which is always the first fact in our domain, while the sixth classifier has no alternative choice, since only one fact has been left in the input. At the other positions, both C4.5 and 1-NN perform better than the two baselines; C4.5 seems to have a slightly better performance than 1-NN. Paired two-tailed *t*-tests at p = 0.005 indicate that the observed differences in accuracy between baselines and ML schemes are statistically significant; the only exception is the selection of the 2nd fact, where there is no significant difference between the base planner and 1-NN.

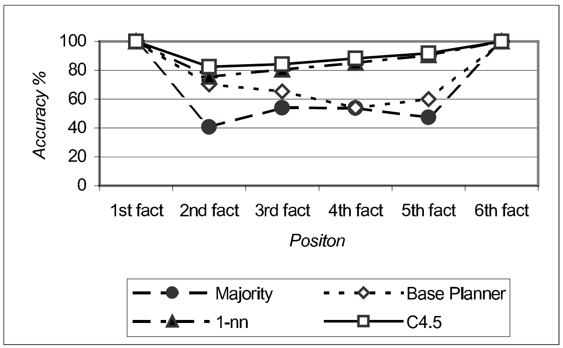


Figure 2: Accuracy scores at each classification stage

Figure 3 shows a text corresponding to the ordering produced by C4.5. The surface text, including aggregation and referring expression generation, was generated by hand, though we plan to automate this process using the corresponding modules of M-PIRO. The ordering of the facts, which are realized as natural language clauses, looks quite reasonable. The flow of information is not the optimal one, but does not cause problems to the understandability or readability of the text. Figure 4 shows the text that corresponds to the ordering of the human annotator. The two texts differ in the placement of the fact *made-of*, which is expressed as "it is made of marble"; C4.5 places this fact at the fourth position instead of the second, which is the right position according to the human annotator. The word "but" in the human text of Figure 4 implies the use of a rhetorical relation; the presence of this relation suggests a possible explanation of why the human text is ordered differently than the one produced by the system. The misplaced fact is penalized three times when computing the accuracy scores of the six classifiers: at the second classifier, where the fact *exhibit-portrays* is selected instead of *made-of*, at the third classifier, where *creation-period* is selected instead of *exhibit-portrays*, and at the fourth classifier, where *made-of* is selected instead of *creation-period*. This implies that the accuracy scores that were presented above are a very strict measure of the performance of our method, and, in fact, our method may actually be performing even better than what the scores indicate.

This exhibit is a portrait. It portrays Alexander the Great and was created during the Hellenistic period. It is made of marble. What we see in the picture is a roman copy. Today it is located at the archaeological museum of Thassos.

Figure 3: Ordering of facts produced using C4.5

This exhibit is a portrait. It is made of marble and portrays Alexander the Great. It was created during the Hellenistic period, but what we see in the picture is a roman copy. Today it is located in the archaeological museum of Thassos.

Figure 4: Ordering of facts as specified by the human annotator

We are currently trying to devise evaluation measures that are better suited to discourse planning, and to NLG in general. More specifically, we plan to apply metrics that assign different penalties depending on the importance of an error, based on the edit distance between the output of the discourse planner and the reference corpus. We also plan to correlate these metrics with human evaluation as proposed by Reiter and Sripada (2002).

5 Conclusions and future work

This paper has presented a machine learning approach to the fact-ordering subtask of discourse planning. We have decomposed the problem into a sequence of multi-class classification stages, where each stage selects the fact to be placed at the corresponding position. Experiments performed using the C4.5 and *k*-NN learning algorithms indicate that our method performs significantly better than both a sequence of simple majority classifiers and a set of manually constructed ordering rules.

Our method can be used with any content determination module that selects a fixed number of facts per document and user type, and gives rise to planners that can be easily retrained for other similar application domains, where sample manually ordered sequences of facts can be obtained. Compared to approaches that employ manually constructed rules, our method has the advantage that it does not require language technology expertise, and, hence, can be used to construct authoring tools that will allow domain experts to control the order of the facts in the generated documents. Furthermore, unlike previous machine learning approaches, our method does not interleave fact ordering with content determination.

As already mentioned, we plan to move towards learning richer discourse plans, which apart from ordering information will also include rhetorical relations, although our experience so far indicates that even just ordering the facts in a natural way can lead to quite acceptable texts. We are currently investigating a more flexible representation that will not be limited by a fixed number of facts per page and, apart from the absolute order of facts, will take into account the relative ordering between facts (e.g., by using n-grams). Further work is planned in order to devise better evaluation measures, and improve the performance of our planners by considering other learning algorithms.

References

Aha D., and Kibler D. 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.

Androutsopoulos I., Kokkinaki V., Dimitromanolaki A., Calder J., Oberlander J. and Not E. 2001. Generating multilingual personalized descriptions of museum exhibits – the M-PIRO project. In *Proceedings of the 29th Conference on Computer Applications and Quantitative Methods in Archaeology*, Gotland, Sweden.

Androutsopoulos I., Spiliotopoulos D., Stamatakis K., Dimitromanolaki A., Karkaletsis V. and Spyropoulos C.D. 2002. Symbolic authoring for multilingual natural language generation. In *Proceedings of the 2nd Hellenic Conference on Artificial Intelligence (SETN-02)*, Thessaloniki, Greece.

Bangalore S. and Rambow O. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrucken, Germany.

Barzilay R., Elhadad N. and McKeown K. 2002. Inferring Strategies For Sentence Ordering In Multidocument News Summarization. *Journal of Artificial Intelligence Research*, 17: 35-55.

Duboue P and McKeown K. 2002. Content Planner Construction via Evolutionary Algorithms and a Corpus-based Fitness Function. In *Proceedings of the 2nd International Natural Language Generation Conference (INLG'02)*, New York, USA, pp. 89-96.

Duboue P. and McKeown K. 2001. Empirically estimating order constraints for content planning in generation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001)*, Toulouse, France, pp. 172-179.

Hovy E. 1993. Automated Discourse Generation Using Discourse Structure Relations. *Artificial Intelligence*, 63(1–2):341–386.

A. Isard, J. Oberlander, I. Androutsopoulos and C. Matheson. 2003. "Speaking the Users' Languages". *IEEE Intelligent Systems*, 18(1):40-45.

Kan M. and McKeown K. 2002. Corpus-trained text generation for summarization. In *Proceedings of the 2nd International Natural Language Generation Conference (INLG'02)*, New York, USA, pp. 1-8.

Karamanis N. and Manurung H. M. 2002. Stochastic Text Structuring using the Principle of Continuity. In *Proceedings of the 2nd International Natural Language Generation Conference (INLG'02)*, New York, USA, pp. 81-88.

Langkilde I and Knight K. 1998. Generation that Exploits Corpus-Based Statistical Knowledge. In

Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL 1998), Montreal, Canada, pp. 704–710.

Malouf R. 2000. The order of prenominal adjectives in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-00)*, Hong Kong, pp. 85-92.

Mann W. and Thompson S. 1988. Rhetorical structure theory: towards a functional theory of text organization. *Text*, 3:243–281.

McKeown K. 1985. Discourse strategies for generating natural language text. *Artificial Intelligence*, 27:1–42.

Marcu D. 1997. From local to global coherence: A bottom-up approach to text planning. In *Proceedings of the 14th National Conference on Artificial Intelligence*, Providence, Rhode Island, pp. 629-635.

Mellish C., Knott A., Oberlander J. and O' Donnell M. 1998. Experiments using stochastic search for text planning. In *Proceedings of the 9th International Workshop on Natural Language Generation*, Ontario, Canada, pp. 97-108.

O'Donnell M., Mellish C., Oberlander J. and Knott A. 2001. ILEX: An Architecture for a Dynamic Hypertext Generation System. *Natural Language Engineering*, 7(3):225–250.

Oh A. and Rudnicky A. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the ANLP/NAACL 2000 Workshop on Conversational Systems*, Seattle, USA, pp. 27–32.

Poesio M., Henschel R. and Kibble R. 2000. Statistical NP generation: a first report. In *Proceedings of the ESSLLI Workshop on NP Generation*, Utrecht, Netherlands.

Quinlan R. 1993. *C4.5: programs for machine learning*. Morgan Kaufmann, 302 p.

Rambow O. 1990. Domain Communication Knowledge. In *Proceedings of the 5th International Workshop on Natural Language Generation*, Dawson, PA.

Ratnaparkhi A. 2000. Trainable methods for surface natural language generation. In *Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of ACL (ANLP-NAACL 2000)*, Seattle, USA, pp. 194–201.

Reiter E. and Dale R. 2000. *Building natural language generation systems*. Cambridge University Press, England, 248 p.

Reiter E. and Sripada S. 2002. Should Corpora Texts Be Gold Standards for NLG? In *Proceedings of the 2nd International Natural Language Generation Conference (INLG'02)*, New York, USA, pp. 97-104.

Shaw J. and Hatzivassiloglou V. 1999. Ordering among premodifiers. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, College Park, Maryland, pp. 135–143.

Varges S. and Mellish C. 2001. Instance-based natural language generation. In *Proceedings of the 2nd Meeting of the North American Chapter of ACL (NAACL-2001)*, Carnegie Mellon University, Pittsburgh, PA.

Walker M. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, 12:387–416.

Walker M., Rambow O. and Rogati M. 2001. SPoT: a trainable sentence planner. In *Proceedings of the 2nd Meeting of the North American Chapter of the ACL (NAACL-2001)*, Carnegie Mellon University, Pittsburgh, PA.

Witten I. and Frank E. 1999. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, 416 p.

Corpus-analysis for NLG

Sabine Geldof
Centre for Language Technology
Macquarie University
Sydney, Australia
sabine@ics.mq.edu.au

Abstract

There is a general interest in corpora of human authored texts as a source for acquiring domain knowledge useful for a natural language generation (NLG) system. It is less clear, however, how this can be done in a systematic way. We propose a principled approach towards acquiring domain knowledge through corpus analysis and illustrate its application in the domain of route descriptions. More specifically, we identify different types of knowledge needed in the NLG process and describe a procedure for systematically analyzing a corpus text and for inventorizing these different types of knowledge. We discuss how these procedures fit into a global approach to corpus analysis and into the natural language generation system development cycle.

1 Introduction

1.1 Knowledge requirements for NLG

The process of automatically generating natural language text from non-linguistic input data has been characterized in very general terms by Reiter and Dale (2000) as consisting of three major phases: document planning, micro-planning and surface realization. The simplest architecture to realize this model is a pipeline, where each of the major processes consumes and/or produces an intermediate representation: from a non-textual input over a document plan and a text specification to an output text (see Figure 1). An important argument for considering three phases (rather than only a planning and a realization one) is related to the type of knowledge used in each of these phases.

Document planning consists of transforming a set of non-textual input data into a document plan (i.e. a structured set of messages). At this point only high-level decisions are being made with respect to the contents of the messages and the text structure.

At the other end of the overall NLG process, the surface realization phase receives a detailed text specification ready to be transformed into a final textual form.

As Reiter and Dale note, the phase of document planning – directly related to the non-textual input data – is highly determined by domain specific knowledge, whereas linguistic realization requires mostly linguistic knowledge. However, the gap between the document plan and the text specification requires a number of processing steps involving a combination of linguistic and domain knowledge. These processes are grouped under the term ‘micro-planning’.

Hence it appears that the development of an NLG system, entails the acquisition of domain specific knowledge, especially to assist the first two phases of the process.

Reiter et al. (2000) describe a variety of techniques for acquiring knowledge for use in NLG systems. One specific technique is the analy-

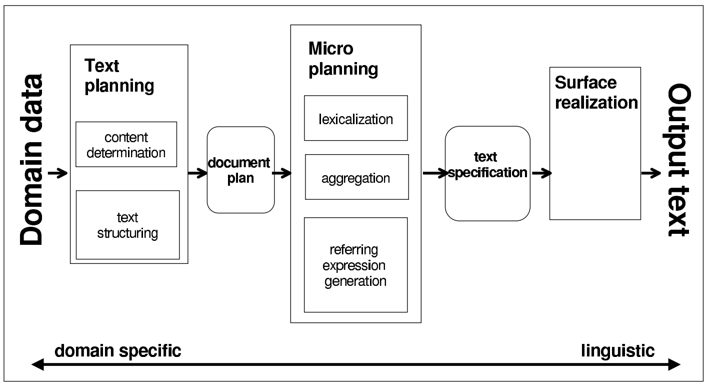
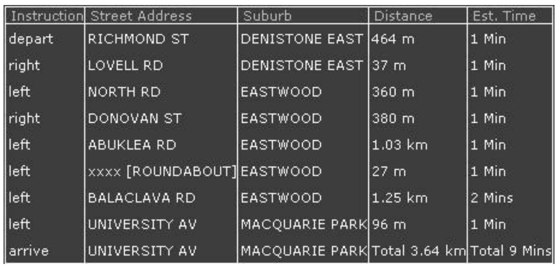


Figure 1: General architecture for NLG system (Reiter and Dale, 2000)

sis of human authored corpora. Reiter and Sripada (2002) note that there is a growing interest in this technique. They describe issues involved in interpreting information extracted from a corpus, warn against the exclusive reliance on corpora for knowledge acquisition and argue for using a good quality corpus rather than a large one. In this paper we investigate which different types of domain-specific knowledge are required (2.1), assuming the general architecture proposed by Reiter and Dale. We propose concrete steps towards analysing a corpus in view of acquiring these types of knowledge (2.2 and 2.3). Finally we describe how the acquired knowledge is integrated in the NLG development cycle (3).

1.2 The domain of route descriptions

The corpora used in this paper as an illustration belong to the domain of route descriptions: textual explanations of a procedure for reaching a specific destination from a particular point of departure. The issues involved in describing a route are distinct from those concerned with calculating the path sequence underlying the route. Systems providing route descriptions on demand via the web (see for example http://www.whereis.com.au) deliver a route description in both cartographic and textual form. The latter generally consists of a more or less straightforward mapping of the calculated route data (paths to follow and turns to take)



Instruction	Street Address	Suburb	Distance	Est. Time
depart	RICHMOND ST	DENISTONE EAST	464 m	1 Min
right	LOVELL RD	DENISTONE EAST	37 m	1 Min
left	NORTH RD	EASTWOOD	360 m	1 Min
right	DONOVAN ST	EASTWOOD	380 m	1 Min
left	ABUKLEA RD	EASTWOOD	1.03 km	1 Min
left	xxxx [ROUNDABOUT]	EASTWOOD	27 m	1 Min
left	BALACLAVA RD	EASTWOOD	1.25 km	2 Mins
left	UNIVERSITY AV	MACQUARIE PARK	96 m	1 Min
arrive	UNIVERSITY AV	MACQUARIE PARK	Total 3.64 km	Total 9 Mins

Figure 2: An automatically generated route description (obtained from http://www/whereis.com.au)

Leave the house and drive towards the Midway shops, at the end of the street turn right and then left at the roundabout. Drive along North road and take the third right turn, just after the first hump in the road. Go to the end of that road and then go straight ahead at the roundabout, there's a church on your left. Now go straight along Herring road for quite a way until you hit the main road (Epping Rd), go straight across at the lights and continue on until you get to the next set of lights. Turn right here into the university.

Figure 3: A human generated route description for the route in Figure 2

to text templates or even to a table, as shown in the example of Figure 2. In the CORAL project, we aim at building an NLG system for producing route descriptions that approximate the naturalness of those produced by humans (see Figure 3) while using input data available in GIS systems that calculate the path. Our approach, described in more detail in Dale et al.(2003), is different from other route description generation systems which assume the availability of rich perceptual (e.g. (Maaß, 1995; Raubal and Winter, 2002)) or semantic information (Pattabhiraman and Cercone, 1990; Moulin and Kettani, 1999). An extensive overview of the field of route description generation systems may be found in the latter paper.

Quite a few studies of route descriptions are based on the analysis of a corpus of human generated directions, often in view of a very specific aspect of this type of discourse. For instance, Klein (1982) illustrated his model for local deixis through examples of route descriptions. Fraczak et al. (1998) used corpus analysis to determine which information in a subway route description is optional or obligatory. Wunderlich and Reinelt (1982) derived from a corpus of route descriptions a model of the interlocutors' interactions. Only Denis (1997) describes a systematic approach to corpus analysis. His aim is to establish measures for qualitative variability among route descriptions. His analysis starts from an assumption of what are essential components in route descriptions. We start from the text, working out what are the building blocks so that an NLG program can approximate it. The corpus used as an illustration in this paper consist of 20 written directions within the urban road network: 9 subjects were asked to describe the route from their homes to the university to a visitor and to a neighbour, and from the university to a fixed, known destination.

2 Corpus analysis approach for NLG

2.1 Domain specific knowledge sources

Assuming the general NLG architecture proposed by Reiter and Dale (2000), we now examine which types of knowledge might be acquired from corpora to support the most domain-sensitive NLG processes.

Document planning decomposes into a process of content determination and one of text structuring. These processes use the following domain specific knowledge sources respectively.

Message types In many NLG systems, messages are created by instantiating message types with data received from another application. Thus the set of possible message types is a knowledge source for the content determination task. A message can generally be characterized as a predicate-argument structure: the predicate asserts a property of an argument or a relationship between arguments and is generally realized as a verb; the argument(s) consist(s) of the main domain entity/ies to which the predicate applies. For example, in our domain, a message might communicate an instruction to follow (predicate) a path (argument). It is not the task of the NLG system to compute this information but rather to determine what configuration of information is appropriate to be communicated as a message.

Text structure patterns The subprocess of text structuring makes explicit the relationship between successive messages in a text. Text structure usually takes the form of a hierarchical structure: a text is composed of different parts, which in turn consist of successive messages in a particular discourse relationship with each other.

In the micro-planning phase, three knowledge sources are used in the refinement of a structured set of messages (the document plan) into a text specification:

Lexemes for message predicates Lexicalization mainly consists in choosing a lexeme to realize each message predicate. Thus, a list of occurring lexemes for each message type predicate is a knowledge source for this process.

RE types for message arguments While the message type specifies which arguments to realize linguistically, the way these domain entities are referred to is still undecided. A list of types of referring expressions for each message type argument constitutes a knowledge source for this process.

Aggregation patterns Finally, human authors tend to combine several messages to form a sentence. Knowledge about which message can be aggregated in which syntactic constructs is needed in order to implement this process.

The following subsections describe major steps in eliciting the above identified knowledge sources.

2.2 Inventorize text planning knowledge

2.2.1 Message types

Analysing a corpus starts by identifying the basic message types occurring in the text. These are to corpus text analysis what objects are to domain modelling: key conceptual elements resulting from segmenting a space into manageable units. Thus the first step in corpus analysis consists in segmenting the text in meaningful units, the next step will be to classify these units into message types. These two analysis processes usually interact, as shown in the procedure described in Figure 5. In this procedure, a key question is what counts as a message. A pragmatic approach is to try to identify the message predicates first (see procedure in Figure 5) and then to further refine the obtained message types on the basis of occurring arguments (see procedure in Figure 6). We will illustrate these procedures with the corpus extract shown in Figure 4.

A first pass segmentation takes the full stops as a boundary (5.1). Then, examining each sentence, we look for re-occurring predicates: what is being communicated? A quite straightforward hypothesis is that route descriptions consist of instructions (hence our hypothesized segmentation criterion, 5.2), more specifically to TURN (in a direction) and to FOLLOW a path. As shown in Figure 4, quite a few sentences occur in between instructions, which we can not label as FOLLOWs or TURNs (5.3). A refinement of our message type list is necessary (5.4). Here the analysis of message arguments (as described in Figure 6) proves useful. A TURN typically takes a *dir* as argument (e.g. 'left' in 251-8-2), while a FOLLOW specifies a *path* ('Burns Bay Rd' in 251-7-3) (6.1). However, we notice that both TURNs and FOLLOWs can take a *point* (i.e. location where a path ends and where a turn is to be taken) as argument too (6.2b), e.g. 'after the second one' in 251-7-2 and 'right till the end' in 251-8-1). This leads to the refinement of the TURN and FOLLOW message types into TURN(DIR), TURN(POINT,DIR) and FOLLOW(PATH), FOLLOW(PATH,POINT) respectively. Moreover, both the *point* and *path* arguments occur in non-instructive clauses, as elaborations in between instructions (e.g. 251-10 and 251-7-3). After a few iterations, our analysis results in the following list of message types: TURN(DIR), TURN(POINT,DIR), FOLLOW(PATH), FOLLOW(PATH, POINT), STATE(POINT), STATE(PATH) and the notion of an instruction remains the syntactico-semantic criterion for determining the message boundary.

5.1 Segment the corpus text into sentences;

5.2 List possible message types (on the basis of predicates);

5.3 Identify textual units according to the list of messages identified in the previous step (this may involve taking sentences together or further segmenting them);

5.4 Refine the list of possible message types (on the basis of message arguments (see procedure in Figure 6);

5.5 Go back to step 3 unless

   a. all textual units are labelled according to a coherent list of message types **and**
   b. syntactico-semantic criteria are defined for message boundaries.

Figure 5: Procedure for segmenting text and identifying message predicates

6.1 List for every message type, the set of occurring arguments;

6.2 Check whether each message type selects a disjunct list of arguments,

   a. If yes: done
   b. If no: go to 6.3

6.3 Add a new message sub-type for each frequently occurring combination of predicate and arguments;

Figure 6: Procedure for identifying message-arguments

2.2.2 Text structure patterns

Both the global structure of a text and the relationship among subsequent messages usually af-

Message ID	Text	4.5b Segm.	4.2 Pred.	4.4 Arg.	7.3 RE type	Text Struct.
251-7-1	You'll go over two bridges	DESCR	STATE	*(path)*	*path*-lm	
251-7-2	and after the 2nd one veer to the right	INSTR	TURN	*(point,dir)*	*point*-deic,*dir*	
251-7-3	and you'll be on Burns Bay Rd.	DESCR	STATE	*(path)*	*path*-name	
251-8-1	Stick on this right till the end, which is at Epping Road,	INSTR	FOLLOW	*(path,point)*	*path*-deic, *point*-type-name	
251-8-2	which you'll want to turn left onto	INSTR	TURN	*(dir)*	*dir*-deic	
251-9-1	Epping Road will take you right near the uni,	DESCR	STATE	*(path,point)*	*path*-name, *point*-lm	META
251-9-2	but don't get onto the M2 mistake.	INSTR	TURN	*(dir)*	*dir*-name	META
251-10-1	A couple of kms after the M2 turn off is Herring Road, at the top of a hill.	DESCR	STATE	*(point)*	*point*-name-descr	

Figure 4: Extract from our analysed corpus

fects other decisions along the NLG process (in our domain the message type selection in the first place). It is therefore useful to investigate whether a document typically consists of different parts and how these can be delimited. A well established structure analysis proposed by Wunderlich and Reinelt (1982) distinguishes among the initial route, intermediate routes and the final route. The initial route is defined as the part of the route that can be seen from the departure point. The final route starts where the destination comes within sight of the route taker. We can delimit these parts (DEP, INT, ARR) in our corpus on the basis of message types occurring in these parts of the route descriptions. For instance, nearly all departure messages imply some notion of direction ((TURN(POINT,DIR)), INSTR(POINT)[1], FOLLOW(PATH,DIR)).

The relationship between successive intermediate route messages is quite specific, consisting mainly of an alternation between FOLLOW and TURN messages. However, we already mentioned that elaborations on *point* or *path* arguments may be included to further clarify an element deemed important in the navigation process. These elaborations relate to the instructional messages as satellites to nuclei according to Rhetorical Structure Theory (Mann and Thompson, 1988). There may also be sporadically occurring 'other' messages, which don't follow the just identified rhetorical relationships. In the case of route descriptions, we noted a number of clauses expressing meta information about the route (e.g. 251-9-1) We categorize also negative instructions as META (e.g. 251-9-2).

Thus, our corpus analysis revealed as canonical text structure: three major parts (which select particular message types) and the relationship within the middle part consisting of sequences of nuclei (TURN and FOLLOW messages) with possible elaborations on arguments (points or paths). This knowledge can be formalized as a rewrite grammar.

2.3 Inventorize micro-planning knowledge

2.3.1 Lexemes for predicates

Once the set of message types are determined and the corpus is completely annotated, it is quite straightforward to order the messages by type and to inventorize the lexemes that are representative of each of the message type classes. Table 1 provides an overview of the lexicalization options encountered in our corpus. Lexemes in *italics* face are possible though not encountered in the corpus. The lexicalization patterns also determine how arguments are realized, the ones between brackets are optional. With a few exceptions, lexicalization options are identical across the message subtypes and we have the choice among at least two lexemes for each message (sub)type, e.g. 'follow'

[1]This new message sub-type accounts for a frequently occurring type of instructions: Leave the house (221-1-1), which is neither a TURN, nor a FOLLOW instruction, but rather makes explicit the point of departure in the form of an instruction.

Message type	Sub-type Lexicalization	Sub-type Lexicalization
TURN	TURN(DIR)	TURN(POINT,DIR)
	turn/*veer* <dir$_{ADVP}$ $>$ (onto/into <street$_{NP}$ $>$)	(at $< point_{NP} >$) turn/veer <dir$_{ADVP}$ $>$ (at $< point_{NP} >$)
	take <dir$_{NP}$ $>$	(at $< point_{NP} >$) take <dir$_{NP}$ $>$ (at $< point_{NP} >$)
	go <dir$_{ADVP}$ $>$	(at $< point_{NP} >$) go <dir$_{ADVP}$ $>$ (at $< point_{NP} >$)
FOLLOW	FOLLOW(PATH)	FOLLOW(PATH,POINT)
	follow $< path_{NP} >$	follow $< path_{NP} >$ until $< point_{NP} >$
	continue along $< path_{NP} >$	continue along $< path_{NP} >$ until $< point_{NP} >$
	stay on $< path_{NP} >$	stay on $< path_{NP} >$until $< point_{NP} >$
	keep going on $< path_{NP} >$	keep going on $< path_{NP} >$ until $< point_{NP} >$
	go down $< path_{NP} >$	go down $< path_{NP} >$ until $< point_{NP} >$
	-	go to the end
	-	go straight until $< point_{NP} >$

Table 1: Overview of lexicalisation options in route descriptions

7.1 Determine the domain entities for which a referring expression generation strategy is needed;

7.2 Determine the analysis parameters that may interact with the RE strategy;

7.3 Identify types of RE for each entity;

7.4 Inventorize REs for the entities identified in 7.1, according to the RE types obtained from 7.3, taking into account the parameters identified in 7.2.

7.5 Go back to 7.3 **unless** the current list of RE types allows to annotate the domain instances.

Figure 7: Procedure for inventorizing referring expressions

and 'continue along' for FOLLOW messages, 'turn' and 'take' for TURN messages.

2.3.2 Types of referring expressions

The procedure for inventorizing types of referring expressions (RE) is similar to the one for lexicalization options but now applies to the arguments of messages instead of the predicates (see Figure 7). In general, the list of arguments identified as part of the message types are the domain entities for which an RE strategy is needed. Thus, in our domain, we inventorized RE types for *path*, *point* and *dir* entities (7.1).

Next, we need to find out whether some aspects of the context in which they appear may have an impact on the choice of a referring expression. We call these analysis parameters. Obviously, the message type is one such parameter. For instance a deictic reference for a *point* occurs only in message type TURN(POINT,DIR). Furthermore, we noticed that in our corpus, an entity might be referred to by a single expression (e.g. 'Burns Bay Rd. for *path* in 251-7-3) or by a combination of expressions (e.g. 'Herring Rd' and 'at the top of a hill' for *point* in 251-10) (7.2). Analysis shows that *points* can be referred to through the name of the intersecting street (*point*-name), the type of intersection (*point*-type, as in 'the end' for a T-junction in 251-8-1), a landmark that occurs at that point (*point*-lm, as 'the uni' in 251-9-1) or a reference to an earlier mention of that point (*point*-deictic, as 'the second one' in 251-7-2). It has to be noted that people use world and perceptual knowledge that is not readily available to the NLG system, such as about the topology of the environment ('at the top of a hill' in 251-10). We have grouped REs for which the underlying information is not available under the common denominator *descriptive* (7.3).

An inventory of the RE types for point shows that mentioning the type of intersection is the only type of RE that occurs across all message types and that it is the most or second most frequently used expression in each message type. It is followed closely (in frequency) by the use of the name of the intersecting street. In contrast to many route descriptions studies (e.g. (Denis, 1997)), which emphasize the importance of landmarks, these don't occur more frequently than the previous types. However, given that they are often used as a single RE, it is still an important RE type.

2.3.3 Aggregation patterns

As for the preceding procedures, we need to determine a set of aggregation patterns for the com-

Aggregation type	**# cases**	**# messages**	**% of total messages**
0	132	133	48
1	57	116	42
2	4	8	3
3	9	18	6
1+	2	2	1
Total	**277 messages**		

Table 2: Occurrence of aggregation patterns in the corpus

bination of messages in sentences and the analysis parameters that might affect their occurrence. Following Reiter and Dale (2000), we distinguish among simple conjunctions (type 1), aggregation with shared constituents or structure (type 2) and syntactic embedding (type 3). Aggregation type 2 occurs only rarely in our corpus. Not surprisingly, since instructive sentences, have no linguistically realized subject (to share). Also the object of a FOLLOW and a TURN message are, by definition, different. Examples of the other two patterns of aggregation do occur frequently: 251-7-1/3 exemplifies the coordination of three messages in one sentence (type 1) and 251-8-1/2 illustrates how a TURN message is syntactically embedded in the preceding FOLLOW message. Furthermore, we labelled the messages which are realized by one sentence (or more) as type 0.

Since we are interested in finding out how messages are combined into sentences, the message type is an obvious analysis parameter, but also the position in the sentence (does this message occur in the first or second position in a conjunction or as main or subordinate clause in the syntactic embedding?). Table 2 shows that most messages are either mapped onto a single sentence (type 0) or simply coordinated (type 1).

As to the relationship between the aggregation type and the message type, further analysis of type 1 aggregation shows that a TURN(POINT-DIR) message is most likely to be coordinated, especially in first position (27/57). In fact, half of these coordinated TURN(POINT,DIR) messages (13/25) are followed by a FOLLOW message. Thus, we might consider TURN(POINT,DIR) *and* FOLLOW (see 251-7-2/3) to be a likely candidate for an aggregation pattern.

3 Integration in NLG development cycle

As described so far, the knowledge sources consist of inventories of constructions occurring in the corpus. They provide a range of choice options for every step in the generation process. We have built and NLG system for route descriptions by combining each of the above knowledge sources with generic NLG processes. The 6 elicited message types (predicates and arguments) constitute the backbone of our system, lexicalisation is kept fairly straightforward and we have implemented an RE strategy for *point* based on the 4 elicited RE types (see details in Dale et al. (2003)). Given input information for a particular route, the system realizes the first choice option for each step in the generation process and, using the mechanism of backtracking, it can realize every possible combination of choice options. The following two fragments (for the route presented in Figure 2) illustrate the range of variation resulting from multiple choice options w.r.t. message type selection and REG.

> Follow Richmond Street for 146m.
> Turn right onto Lovell Road.
> Follow Lovell Road for 37m. until you reach North Road.
> Turn left.

> Follow Richmond Street until you reach the end.
> Turn right. Follow Lovell Road for 37m.
> At the roundabout, turn left onto North Road.

Additional control knowledge is needed to reduce the number of generated alternatives and to determine which constructs to apply in a given situation. Elicitation of control knowledge requires to take into account the context, both linguistic (which message precedes, follows, in which part of the text does this message appear?) and non-linguistic context (i.e. application dependent features of the instances which form the arguments of the messages). This is a multi-dimensional problem, again highly domain dependent, which is the object of our on-going work.

4 Conclusion

As pointed out by Reiter et al.(Reiter et al., 2000), the evaluation of the knowledge acquisition process is very costly and yields only suggestive outcomes. This holds also for our domain, where the effectiveness of descriptions depends on many extra linguistic factors, such as user preferences and physical properties of the environment. At this stage in the development, we consider the capability of generating a range of possible formulations (as cited in Section 3), occurring in a corpus of human generated descriptions to be an important step towards more natural route descriptions (compare these with Figure 2 and 3). We plan however small scale informal evaluations of the system including control knowledge.

While the examples cited are from one corpus, the approach has been used in other corpora belonging to a different sub-domain: directions for navigation by foot on two different campuses. Interestingly, comparing these analyses with the one presented here sheds a light on the differences between these sub-domains and allows us to term these in concrete figures, e.g. about the distribution of message types, RE types and aggregation patterns. Corpus analysis in view of NLG thus contributes to the understanding of the domain.

More importantly, this work is part of an ongoing endeavour to formalize and clearly distinguish NLG knowledge that is generic (hence reusable) from domain specific knowledge which has to be acquired for every new application domain. A systematic approach to corpus analysis contributes to the bottom-up elicitation of these distinctions.

References

R. Dale, S. Geldof, and J.-P. Prost. 2003. Coral: Using natural language generation for navigational assistance. In *Proceedings of the 26th Australasian Computer Science Conference ACSC'03*, Adelaide, South Australia.

M. Denis. 1997. The description of routes: A cognitive approach to the production of spatial discourse. *Current Psychology of Cognition*, 16(4):409–458.

L. Fraczak, G.Lapalme, and M.Zock. 1998. Automatic generation of subway directions: salience gradation as a factor for determining message and form. In *Proceedings of the International Workshop on Natural Language Generation*, pages 58–67, Niagara-on-the-Lake, CA.

W. Klein. 1982. Local deixis in route directions. In R. Jarvella and W. Klein, editors, *Speech, Place and Action. Studies in deixis and related topics.*, pages 161–182. John Wiley & Sons, Ltd.

W. Maaß. 1995. How spatial information connects visual perception and natural language generation in dynamic environments: towards a computational model. Technical Report 116, Universitaet des Saarlandes, FB 14 Informatik IV.

W.C. Mann and S.A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

B. Moulin and D. Kettani. 1999. Route generation and description using the notions of object's influence area and spatial conceptual map. *Spatial Cognition and Computation*, 1:227–259.

T. Pattabhiraman and N. Cercone. 1990. Selection: Salience, relevance and the coupling between domain-level tasks and text planning. In K. McKeown, J.Moore, and S. Nirenburg, editors, *Proceedings of the 5th International Workshop on Natural Langauge Generation*, Dawson, Pennsylvania.

M. Raubal and S. Winter. 2002. Enriching wayfinding instructions with local landmarks. In *GISscience 2002*, Lecture Notes in Computer Science. Springer.

E. Reiter and R. Dale. 2000. *Building natural language generation systems*. Cambridge University Press.

E. Reiter and S. Sripada. 2002. Should corpora text be gold standards for nlg? In *Proceedings of the Second International Conference on Natural Language Generation*, pages 97–104, New York, USA.

E. Reiter, R. Robertson, and L. Osman. 2000. Knowledge acquisition for natural language generation. In *Proceedings of the First International Conference on Natural Language Generation*, pages 217–225, Mitzpe Ramon, Israel.

D. Wunderlich and R. Reinelt. 1982. How to get there from here. In R. Jarvella and W. Klein, editors, *Speech, Place and Action. Studies in deixis and related topics.*, pages 182–201. John Wiley & Sons, Ltd.

Handling Dependencies in Reorganizing Content Specifications A Case Study of Case Analysis

Helmut Horacek
Universität des Saarlandes, FR 6.2 Informatik
Postfach 151150, D-66041 Saarbrücken, Germany
email: horacek@cs.uni-sb.de

Abstract

Handling the dependencies among alternatives in composing expressions in an efficient and qualitatively accurate manner is a fundamental problem of NLG. To pursue this goal effectively, simplifications are put forward in practical approaches, but also ambitious control regimes are tried out occasionally. However, neither of these is able to operate adequately on larger and involved structures. Approaching this issue in a methodological way, we present a case study from the area of mathematical proofs that illustrates the rhetorically motivated reorganization of machine-generated case analyses. Ingredients of this investigation are the design of optimization operations, effect-minimizing orderings on groups of operations, and tentative applications of local operations to test the effects of crucial dependencies. Our approach conceives NLG as a standard pipe-line architecture putting emphasis on orderings, with local revisions as a minor extension. This is particularly effective when text planning is organized as an optimization rather than as a construction process, such as for the presentation of mathematical proofs.

1 Introduction

Handling the dependencies among alternatives in composing expressions in an efficient and qualitatively accurate manner is a fundamental problem of NLG. To pursue this goal effectively, simplifications are typically put forward in practical approaches, following the standard, pipe-line architecture (Reiter, 1994). In order to handle dependencies in a more profound manner, some ambitious control regimes have been tried out to compute the best combination of alternatives within a local task that satisfies the constraints imposed by these dependencies. Tasks addressed include lexicalization (Beale, 1997) and referring expression generation (Gardent, 2002), which transfer the problem, making it accessible to a general and fast search procedure (on the lines of (Germann et al. 2001) for machine translation). However, neither of these approaches is able to operate adequately on larger and involved structures.

Approaching this issue in a methodological way, we present a case study from the area of mathematical proofs, that illustrates the rhetorically motivated reorganization of machine-generated case analyses. In order to adapt the structure of machine-generated proofs better on human needs, we have analyzed differences between internal proof representations and some structural properties underlying textbook proofs, focusing on case analyses. The differences observed inspired us in defining a set of restructuring operations for proof graph representations whose contextually motivated application enables expressing these reasoning structures in a rhetorically adequate manner. Ingredients of this investigation are the design of optimization operations, effect-minimizing orderings on groups of operations, and tentative applications of local operations to test the effects of crucial dependencies. Our approach conceives NLG as a standard pipe-line architecture putting emphasis on orderings, with local revisions as a minor extension. This is particularly effective when text planning is organized as an optimization rather than as a construction process, such as for the presentation of mathematical proofs.

This paper is organized as follows. We examine the role of case analysis in book presentations. We formalize a set of reorganization operations, and describe their contextually motivated application. We illustrate the effect of these operations by reorganizing a moderately complex proof for NL presentation. Finally, we discuss impacts of our approach.

2 Motivation – Needs of the Domain

Representations of results of automatically found proofs differ from rhetorically adequate natural language (NL) presentations of these proofs not only in their format and conventions but more fundamentally in their content and structure. While the adequacy of a presentation's content has been addressed by a number of transformation and abstraction techniques, a proof's structure is typically preserved by today's proof presentation systems. This is hardly surprising – preserving given structures in specifications is also what NL generation systems typically do.

If either a or b is 0, then both $|ab|$ and $|a||b|$ are equal to 0.
If $a > 0$ and $b > 0$, then $ab > 0$ so that $|ab| = ab = |a||b|$.
If $a > 0$ and $b < 0$, then $ab < 0$ so that $|ab| = -ab = a(-b) = |a||b|$. The other two cases are treated similarly.

Figure 1. The proof of the lemma $|ab| = |a||b|$ in an analysis textbook (Bartle, Sherbert, 1991)

Case analysis is a common reasoning structure in several areas of mathematics. It is applied to solve subproblems which require distinct inferencing in dependency of the values that some expression can take. The simplest and rather common distinction is truth or falsity of a logical expression. Another common situation is the distinction of values of a numerical expression when broken down into intervals. As an example, consider a proof of the elementary lemma $|ab| = |a||b|$. In an introductory book on analysis (Bartle, Sherbert, 1991), it runs as indicated in Figure 1. A verbalization in the style of the proof presentation system *P.rex* (Fiedler, 2001) is given in Figure 2. In contrast to the stereotype machine-generated text, the textbook version is better for two main reasons: (1) each case conveys a reasoning chain distinct from other cases, and (2) concise conditional clauses are formulated instead of overloaded textual markings of underlying case structures. Further effective presentation means used in the textbook are leaving out easily inferable intermediate steps ($|0b|$, $|a0|$, $0|b|$, $|a|0$ in the machine-generated version), referring to analogy, thereby trusting the addressee to be able to make use of it, and extraposing crucial justifications ($ab < 0$, $ab > 0$). In order to present case analyses of automatically found proofs in a similar manner, making adequate use of these presentation means is required. The inference issue, that is

To prove the lemma, let us consider the cases where $a = 0$, $a > 0$, and $a < 0$, respectively.
Case 1: $a = 0$. Then $|ab| = |0b| = 0 = 0|b| = |a||b|$.
Case 2: $a > 0$. Let us consider the cases where $b = 0$, $b > 0$, and $b < 0$, respectively.
  Case 2.1: $b = 0$. Then $|ab| = |a0| = 0 = |a|0 = |a||b|$.
  Case 2.2: $b > 0$. Then $|ab| = ab = |a||b|$.
  Case 2.3: $b < 0$. Then $|ab| = -ab = a(-b) = |a||b|$.
Case 3: $a < 0$. Let us consider the cases where $b = 0$, $b > 0$, and $b < 0$, respectively.
  Case 3.1: $b = 0$. Then $|ab| = |a0| = 0 = |a|0 = |a||b|$.
  Case 3.2: $b > 0$. Then $|ab| = -ab = (-a)b = |a||b|$.
  Case 3.3: $b < 0$. Then $|ab| = ab = (-a)(-b) = |a||b|$.

Figure 2. The proof of the lemma $|ab| = |a||b|$ in the style of the system *P.rex* (Fiedler, 2001)

crucial for many machine-found proofs, can be handled reasonably well (Horacek, 1999). Therefore, we can concentrate our effort on transducing case analysis structures underlying machine-found proofs into structures that mimic those found in textbooks. For this purpose, we have analyzed presentations of comparably simple proofs in two textbooks: a book on analysis (Bartle, Sherbert, 1991) and another one on algebra (Lamprecht, 1981). The proofs examined are likely to be subject to explanations or to exercises in tutorial sessions.

The central issue is to characterize situations for the use of implicit forms of expressing case analyses, in terms simple enough for estimating them on the level of the proof graph. Implicit forms may be sequences of conditional clauses, followed by stating that the conclusion is independent of the conditions, or preceding a critical case, introduced by "it remains to show that this also holds for <condition>". Therefore, we have looked at complexities of the inference chains and expressions involved. For the formal account, we have approximated these properties in terms of the number of clauses (a series of equations was counted as a single clause). We found out that case analyses are expressed explicitly only if the presentation of two or more individual cases requires a certain degree of complexity. Therefore, we consider the length of the second largest branch a crucial parameter (P_1, default value 2) according to the results in Table 1 (26 examples found in the two books). Short case analyses marked explicitly contain untypically large expressions (we neglect this because of the low frequency), or they combine several cases (e.g., by the union of intervals).

Further characterizations gained from the textbook proofs are as follows: The case analyses contain mostly two, rarely more than three

length of inference chain	*per case*						*non-largest cases*					
number of clauses	1	2	3	4	5	> 5	1	2	3	4	5	> 5
frequencies implicit forms	12	12	3	1	1	1	8	6	0	0	0	0
frequencies explicit forms	1	9	5	2	4	4	3	5	0	3	2	2

Table 1. Textual forms of expressing case analyses and number of clauses

cases. Therefore, we limit the number of cases by P_2 (with default 5), for case analyses built by reorganization. Moreover, case expressions found are not more complex than a disjunction of two relations of compatible type (as in Figure 1). We take this into account by using a parameter for the maximum number of operators in case expressions, P_3 (with default 3). Furthermore, all case expressions with an equivalence treated as a conjunction of two implications with opposite directions were marked explicitly, which is easy to obey as a qualitative criterion. Finally, we encountered only one nested case analysis (a longer induction proof), although it is clear from the literature that this occurs more frequently in advanced proofs.

The estimated parameter values guide pursuing the requirements for a rhetorically adequate presentation of case analyses:

- Be economic by keeping the number of cases small.
- Produce structures that enable the use of implicit textual forms.
- Avoid nested case analyses, values of P_2 and P_3 permitting.

In order to fulfill these requirements, some measures are applied, including the reduction of inference chains within individual cases, aggregation of nearly identical cases, and the linearization of nested case analyses. The structures obtained can be explored in a human-oriented way by focusing on crucial distinctions, and they can be expressed more naturally and concisely by NL texts.

3 Restructuring Operations

As the texts in Figures 1 and 2 demonstrate, automatically generated proofs typically meet these requirements to an insufficient degree only. Since there are several logically equivalent forms of a proof, building alternative representations can be pursued that are preferable in rhetorical terms. This is done by condensing a proof, to meet the constraints just elaborated. Three operations aim at this goal, through reducing case analyses in terms of their *length*, *number* of cases, and *depth*:

Operation 1. The *length* of the reasoning line within a single case can be reduced if a subsequence heading this inference line is independent of the case assumption. This subsequence can be lifted out of the scope of the case analysis, and precede the case analysis in the presentation. This operation may be applicable several times, to multiple branches within the same case analysis. Lifting out subsequences is then delicate due to coherence matters, since reasoning chains that are originally sequential would become interleaved. Therefore, if several cases can be reduced by this operation, it is only applied to the one leading to the largest reduction, provided the second largest branch gets reduced enough so that its length falls below threshold P_1 (see the example in Section 5).

Operation 2. The *number* of cases can be reduced if the reasoning lines in some of the cases are identical but for references to the corresponding case assumptions. Such cases can be aggregated by building a disjunction of their assumptions, and by substituting this expression for the references to the original assumptions, provided that case expression is less complex than P_3 allows. Remaining within complexity limits may be possible by simplifications (e.g., union of overlapping or adjacent intervals). If not only some cases, but all cases can be aggregated this way, the entire case analysis collapses into a single inference chain. This can occur frequently in connection with calls to external systems, such as computer algebra systems – a case analysis has to be carried out for each value separately, but it turns out that the inference lines differ only in these values. We encountered such a situation for categorizing residue classes (Meier, Pollet, and Sorge, 2002). In our current context, operation 2 is applicable to cases 2.1 and 3.1 in the proof in Figure 2, provided these cases are lifted from the embedding case analyses (through operation 3). Operation 2 would also be applicable to any of these cases and case 1, provided the intermediate expressions ($|a0]$, $|a|0$, $|0b|$, $0|b|$) are left out – this illustrates the importance of handling the inferability issue.

Operation 3. The *depth* of a case analysis with further embedded case analyses can be reduced by lifting an embedded case analysis to the level of the embedding one. This is done by moving copies of the inferences of the embedding case that precede the embedded case analysis into that structure, and by merging the case assumption associated with the embedding case with each of the case assumptions associated with the embedded structure. Requirements for this operation are that the types of relations in the merged case assumptions are compatible, that the resulting expression is less complex than P_2 and P_3 allow, and that the original structures would be expressed explicitly; otherwise, nested structures are maintained. Moreover, if a length increase is the result of copying inference sequences, subsequent applications of operation 2 to cases from different levels in the original structure must yield reductions compensating that increase.

These operations are formally defined in Figure 3, using generalized case schemata with an arbitrary number of cases n:

$$\frac{\Delta \vdash \bigvee^{i=1,n} F_i \qquad \overset{[F_i]}{\Delta, F_i \vdash G}}{\Delta \vdash G}\ \text{CASE}$$

Δ represents a set of assumptions, and F, G, and H (with or without indexes) are metavariables representing expressions. The case schema represents the conclusion that G is derivable from the assumptions Δ ($\Delta \vdash G$), on the basis of two premises: 1) there is a case split, that is, one of the cases F_i holds, given the assumptions Δ ($\Delta \vdash \vee F_i$) for $1 \leq i \leq n$, and 2) asssuming an arbitrary one of these cases ($[F_i]$) enables the derivation of G from the assumptions Δ and the case assumption F_i ($\Delta, F_i \vdash G$).

In operation 1, the derivation $\Delta_0 \vdash F_0$, is lifted out of the scope of the embedding case analysis, since it is independent of the case assumption $[F_k]$. In operation 2, the two cases $[F_j]$ and $[F_k]$ are aggregated into a single case $[F_j \vee F_k]$, provided the associated derivation trees (tree($\Delta, F_j \vdash G$) and tree($\Delta, F_k \vdash G$), respectively), are identical but for the appearances of F_j and F_k, so that substituting $F_j \vee F_k$

Operation	*Original case schema*		*Modified case schema*	*Constraints*
1)	$\dfrac{\Delta \vdash \vee_{i=1,n} F_i \qquad \dfrac{\overset{[F_k]}{\dfrac{\Delta_0 \vdash F_0}{} \quad \dfrac{\Delta, F_k \vdash \Delta_1}{}}}{F_0, \Delta_1 \vdash G} \qquad \dfrac{\overset{[F_{i\neq k}]}{\ldots}}{\Delta, F_{i\neq k} \vdash G}}{\Delta \vdash G}\ \text{CASE}$	$\Rightarrow$	$\dfrac{\Delta \vdash \vee_{i=1,n} F_i \qquad \dfrac{\dfrac{\Delta_0 \vdash F_0}{} \quad \dfrac{\overset{[F_k]}{}}{\Delta, \{F_k\} \vdash \Delta_1}}{\Delta_1, F_0 \vdash G} \qquad \dfrac{\overset{[F_{i\neq k}]}{\ldots}}{\Delta, F_{i\neq k} \vdash G}}{\Delta \vdash G}\ \text{CASE}$	$\Delta \supseteq \Delta_0$
2)	$\dfrac{\Delta \vdash \vee_{i=1,n} F_i \quad \dfrac{[F_j]}{\Delta, F_j \vdash G} \quad \dfrac{[F_k]}{\Delta, F_k \vdash G} \quad \dfrac{[F_{i\neq j,k}]}{\Delta, F_{i\neq j,k} \vdash G}}{\Delta \vdash G}\ \text{CASE}$	$\Rightarrow$	$\dfrac{\Delta \vdash \vee_{i=1,n} F_i \quad \dfrac{[F_j \vee F_k]}{\Delta, F_j \vee F_k \vdash G} \quad \dfrac{[F_{i\neq j,k}]}{\Delta, F_{i\neq j,k} \vdash G}}{\Delta \vdash G}\ \text{CASE}$	tree($\Delta, F_j \vdash G$) $[F_j \vert F_j \vee F_k]$ $\equiv$ tree($\Delta, F_k \vdash G$) $[F_k \vert F_j \vee F_k]$
3)	$\dfrac{\Delta \vdash \vee_{i=1,n} F_i \qquad \dfrac{\overset{[F_k]}{\dfrac{\Delta_0 \vdash G_0}{} \quad \dfrac{\vee_{j=1,m} G_j \quad \overset{[G_j]}{}}{G_0, G_j, F_k \vdash F_0}}}{\Delta, F_0 \vdash G} \qquad \dfrac{\overset{[F_{i\neq k}]}{\ldots}}{\Delta, F_{i\neq k} \vdash G}}{\Delta \vdash G}\ \text{CASE}$	$\Rightarrow$	$\dfrac{\Delta \vdash \vee_{i=1,n+m\ (i\neq k)} H_i \qquad \dfrac{\overset{[H_i]}{\ldots}}{\Delta, H_i \vdash G}}{\Delta \vdash G}\ \text{CASE}$	$i=1,n\ (i\neq k)$: $H_i \equiv F_i$; $i=n+1,n+m$: $H_i \equiv F_k \vee G_{i-n}$; tree($\Delta, H_i \vdash G$) $\equiv$ Adjoin(tree($\Delta_0 \vdash G_0$), tree($G_i, G_0, F_k \vdash F_0, \Delta \vdash G$))

Figure 3. Proof reorganizing operations that reduce case analyses in terms of length (1), number of cases (2), and depth (3)

for each of them leads to identical derivation trees. Finally, operation 3 merges the cases $[G_j]$ with the embedding case $[F_k]$, which replaces case F_k by m new cases in the embedding case analysis. The new case assumptions are $F_k \vee G_{i-n}$, and the derivation trees (tree($\Delta_0 \vdash G_0$) from F_k, and tree($G_0, G_j, F_k \vdash F_0, \Delta \vdash G$) from each G_j) are composed through adjoinment.

4 Organizing the Reorganization

Judging the adequacy of individual operation applications in context is delicate since it is not locally clear whether or not a case analysis can ultimately be expressed by a concise sentence pattern. There are two reasons for this difficulty, which are typical for processes in NL generation and organizing their specifications:

External dependencies, which appear in terms of how inference chains within a case analysis are verbalized. This is determined by decisions about the content (building abstractions due to the inferability issue) and about form, the verbalization proper.

Internal dependencies, since reductions are usually brought about by the composition of several operation applications.

In order to handle external dependencies, the content-related processes (here: the abstraction process as defined in (Horacek, 1999)) are carried out prior to dealing with reorganization, since this usually leads to significant length reductions that are unforeseeable otherwise. Doing this independently of case analysis reorganizations is also possible, because the inferability addresses sequences of inferences rather than tree-like structures. Conversely, effects of proper verbalization have to be anticipated, because interleaving them with reorganization would be very expensive – verbalizing some substructures several times would be required, since the reorganization process uses local backtracking (see below). Therefore, length estimates are required for the resulting structures. We obtain them by simply counting the number of inference steps, adding one point for required structural markings. The numbers obtained this way are comparable to the threshold parameters estimated on the basis of the empirical analysis.

Concerning internal dependencies, determining the local suitability is only difficult for operation 1. In theory, multiple applications to the same case may be possible and required to be tested; in practice, it is mostly a single application. However, testing multiple applications in an efficient manner is not trivial. It is done by traversing the proof subgraphs of each case separately, starting from the leaf nodes, until a use of the case assumption is encountered; the size of the remaining portion of the proof

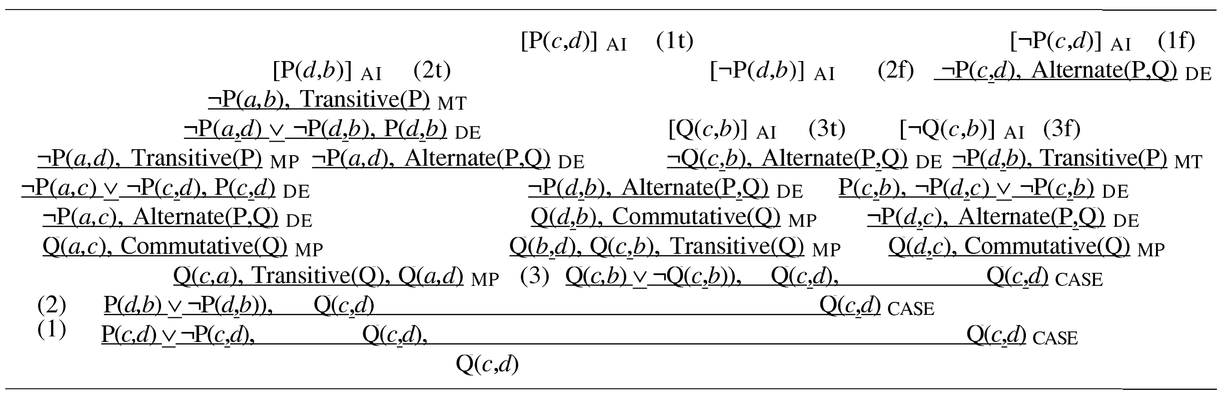

[P(c,d)] AI (1t)
[¬P(c,d)] AI (1f)
[P(d,b)] AI (2t)
[¬P(d,b)] AI (2f)
¬P(c,d), Alternate(P,Q) DE
¬P(a,b), Transitive(P) MT
¬P(a,d) ∨ ¬P(d,b), P(d,b) DE
[Q(c,b)] AI (3t)
[¬Q(c,b)] AI (3f)
¬P(a,d), Transitive(P) MP
¬P(a,d), Alternate(P,Q) DE
¬Q(c,b), Alternate(P,Q) DE
¬P(d,b), Transitive(P) MT
¬P(a,c) ∨ ¬P(c,d), P(c,d) DE
¬P(d,b), Alternate(P,Q) DE
P(c,b), ¬P(d,c) ∨ ¬P(c,b) DE
¬P(a,c), Alternate(P,Q) DE
Q(d,b), Commutative(Q) MP
¬P(d,c), Alternate(P,Q) DE
Q(a,c), Commutative(Q) MP
Q(b,d), Q(c,b), Transitive(Q) MP
Q(d,c), Commutative(Q) MP
Q(c,a), Transitive(Q), Q(a,d) MP
(3) Q(c,b) ∨ ¬Q(c,b)), Q(c,d), Q(c,d) CASE
(2) P(d,b) ∨ ¬P(d,b)), Q(c,d) Q(c,d) CASE
(1) P(c,d) ∨ ¬P(c,d), Q(c,d), Q(c,d) CASE
Q(c,d)

Figure 5. Initial representation of the proof, abstracted to applications of commutativity, transitivity, and alternativity axioms

subgraph is then compared with P_1. That leaves the dependencies among these operation to be dealt with. Within a full proof, this is done by traversing the entire proof graph starting from its leaf nodes. If a case analysis is encountered, operation 2 is tried first, for all pairs of cases (quadratic complexity). Next, operation 1 is tested for each branch, and the suitability can be decided due to the prior application of operation 2. Finally, linearization with eventually embedded case analyses is treated. If this application of operation 3 leads to a number of cases larger than P_2, it is applied tentatively, testing the effect of operation 2 for the combinations of cases not yet considered (one from each of the case analyses combined tentatively). If this leads to a sufficient reduction, the process is resumed with the structures built tentatively; otherwise, with those built before.

5 A Moderately Complex Example

In this section, we illustrate the capablities of our method by a proof to the problem specified in Figure 4. The abstracted proof representation is given in Figure 5, with abbreviations for the use of axioms: the predicates *transitive*, *commutative*, and *alternate* (of P or Q) stand for the instantiated forms of these axioms in the specific proof line. Moreover, MP, MT, DE, AI, and CASE stand for modus ponens, modus tollens, disjunction elimination, assumption introduction, and case analysis, respectively. This form is obtained from the machine-oriented proof by abstracting to the *partial assertion representation level* according to (Horacek, 1999), that is, applications of axioms, unless their use is considered cognitively difficult (e.g., modus tollens, as motivated in (Johnson-Laird, Byrne, 1990)).

The proof contains three case analyses, discriminating according to the truth values of P(*c*,*d*) (1), P(*d*,*b*) (2), and Q(*c*,*b*) (3), with true (t) and false (f), labeled accordingly in Figure 5. Due to their nestings, a completely explicit verbalization of this structure would result in a

Let Q be a transitive and symmetric relation and P a transitive relation. Let P and Q hold 'alternatively', i.e., P(*x*,*y*) ∨ Q(*x*,*y*) for all *x* and *y*. Let also ¬P(*a*,*b*) for some arbitrary *a* and *b*.
Then Q(*c*,*d*) holds for arbitrary *c* and *d*.

Figure 4. Problem definition

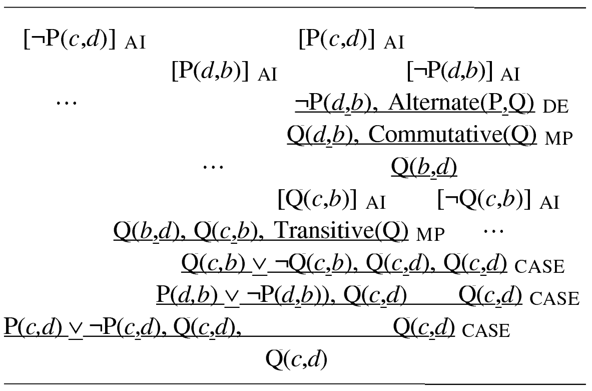

[¬P(c,d)] AI
[P(c,d)] AI
[P(d,b)] AI
[¬P(d,b)] AI
...
¬P(d,b), Alternate(P,Q) DE
Q(d,b), Commutative(Q) MP
...
Q(b,d)
[Q(c,b)] AI
[¬Q(c,b)] AI
Q(b,d), Q(c,b), Transitive(Q) MP
...
Q(c,b) ∨ ¬Q(c,b), Q(c,d), Q(c,d) CASE
P(d,b) ∨ ¬P(d,b)), Q(c,d) Q(c,d) CASE
P(c,d) ∨ ¬P(c,d), Q(c,d), Q(c,d) CASE
Q(c,d)

Figure 6. Final representation of the proof, focusing on modified parts

rather cumbersome text. Since operation 2 is not applicable to any substructure here, we focus on the other operations. Starting with the most embedded case analysis, (3), the applicability of operation 1 is tested for both of its branches. In (3t), the first three inference steps are independent of Q(*c*,*b*), while it is only the first one in (3f) for ¬Q(*c*,*b*). Since the structure resulting from lifting the longer inference subsequence (of (3t)) out of the scope of (3) remains within the limits P_2 and P_3, operation 1 is applied within (3t). Moving on to the embedding case analysis, (2), operation 1 is only applicable to the first inference step in (2t), but with an insufficient reduction. Operation 3 is not applied because of the expectation that the embedded case analysis can be expressed in an implicit form. Conversely, this operation is also not applied in (1), because this expectation holds for the embedding case analysis. Figure 6 illustrates the structural changes imposed on the proof graph in Figure 5, and the later reordering of (1). Figure 7 gives a possible verbalization. It is based on the assumption that axiom uses, unless being in modus tollens direction, are inferable and can be omitted, and that facts used in inferences need to be reintroduced after not being mentioned for some time (see the model in *P.rex*). Altogether, the example demonstrates that only a few of all possible operator applications are truly effective, guided by anticipations about options available in subsequent processing.

If ¬P(*c*,*d*) holds, Q(*c*,*d*) follows. Hence, it remains to show, that Q(*c*,*d*) also holds if P(*c*,*d*). To prove that, let us consider the cases where P(*d*,*b*) and ¬P(*d*,*b*) hold, respectively.

1. P(*d*,*b*) holds. Then ¬P(*a*,*b*) implies ¬P(*a*,*d*) ∨ ¬P(*d*,*b*) due to transitivity, which implies ¬P(*a*,*d*) and further Q(*a*,*d*). ¬P(*a*,*d*) implies ¬P(*a*,*c*) ∨ ¬P(*c*,*d*) due to transitivity, and further ¬P(*a*,*c*) due to P(*c*,*d*). This implies Q(*a*,*c*) and further Q(*c*,*a*). Then Q(*a*,*d*) implies Q(*c*,*d*).
2. ¬P(*d*,*b*) holds. This implies Q(*d*,*b*) and further Q(*b*,*d*). If Q(*c*,*b*) holds, then Q(*c*,*d*) is valid. If ¬Q(*c*,*b*) holds, then P(c,*b*). ¬P(*d*,*b*) implies ¬P(*d*,*c*) ∨ ¬P(*c*,*b*) due to transitivity. This implies ¬P(*d*,*c*) and further Q(*c*,*d*).

Hence Q(*c*,*d*) irrespective of the truth of P(*d*,*b*).

Figure 7. A possible verbalization of the proof sketched in Figure 6

6 Discussion

Proof presentation systems, as NLG systems in general, tend to express all specifications explicitly and in a form widely corresponding to these specifications. Text planning, notably approaches based on Rhetorical Structure Theory (RST) (Mann, Thompson, 1983) address the choice of connectives and clause structures, such as subordinates versus embedding. Exceptions deal with differences between intentional and ideational structures (Maier, 1985) and elaborations on sequences (Horacek, 1998). The latter measure is also incorporated into skillful presentations of the specific form of series of inequations, for remarks on individual steps in such a series (Fehrer, Horacek, 1997). Altogether, no approach to proof presentation or to NLG in general is able to deal with structures of a case analysis in a rhetorically adequate manner.

The task of proof presentation in natural language can be considered a special form of NLG with restricted language and embedding of formulas, where the machine-generated proof is interpreted as a complete and correct, but rhetorically inadequate text plan. Consequently, reorganizations are essential contributions for making machine-generated solutions (here, proofs) better accessible for tutorial purposes (Melis, Horacek, 2000). The modified proofs indicate more adequately which distinctions to make and what cases to consider first – these are essential aids for supporting humans when searching for a proof in a tutorial environment.

The use of reorganization operations can be considered a first step towards a new approach to text planning. It is conceptualized as an optimization process to some externally given highly structured specification, rather than a homogeneous process (Marcu, 1997) that is applied to sets of facts and relations. Many choices are motivated by soft, size-related criteria, as opposed to the more usual qualitative rhetorical preferences. Text planning is broken into a sequence of subphases, so that the most intertwined dependencies can be dealt within the same subphase in a computationally reasonable manner. The position of a subphase is determined by assessing whether results from other subphases should be available, or estimating them is sufficient for well-motivated choices. Hence, we would incorporate lemmatization into our present model between the inferability and the case analysis task. The deletions obtained by the inferability task are the most crucial information, while the effects of lemmatization are more global than those of handling case analyses. This process organization mediates between opportunistic application of operators and a systematic procedure (as (Dalianis, 1999) and (Shaw, 1998) for aggregation, respectively). It constitutes a compromise between potentially considering all possible combinations and a strict pipe-line.

7 Conclusion

In this paper, we have addressed dependencies among alternatives in composing expressions, for larger and involved structures. We have presented a case study from the area of mathematical proofs that illustrates the rhetorically motivated reorganization of machine-generated case analyses. A set of restructuring operations is defined on the proof graph whose contextually motivated application enables expressing these reasoning structures in a rhetorically adequate manner. Operations include shortening inference chains within individual cases, aggregation of nearly identical cases, and the linearization of nested case analyses. We will soon integrate the stand-alone process into the system *P.rex*. Our approach shows a new kind of handling dependencies in text planning, and it contributes to expressing mathematical proofs for didactic purposes.

References

Bartle, R., Sherbert, D. 1991: Introduction to Real Analysis.

Beale, S. 1997: HUNTER-GATHERER: Applying Constraint Satisfaction, Branch-and-Bound and Solution Synthesis to Computational Semantics. PhD thesis, Carnegie-Mellon University.

Dalianis, H. 1999. Aggregation in Natural Language Generation, *Computational Intelligence* 15(4).

Fehrer, D., Horacek, H. 1999. Presenting Inequations in Mathematical Proofs. In *Information Sciences* 116, pp. 3-23.

Fiedler A. 2001. Dialog-driven Adaptation of Explanations of Proofs. In *Proc. of IJCAI-2001,* pp. 1296-1300.

Gardent, C. 2002. Generating Minimal Definite Descriptions. In Proc. of *ACL-2002*, pp. 96-103.

Germann, U., Jahr, M., Knight, K., Marcu, D., Yamada, K. 2001. Fast and Optimal Decoding for Machine Translation. In Proc. of *ACL-2001*, pp. 228-235.

Horacek, H. 1998. Generating Inference-Rich Discourse Through Revisions of RST-Trees. In *Proc. of AAAI-98,* 814-820.

Horacek, H. 1999. Presenting Proofs in a Human-Oriented Way. In *Proc. of CADE-99*, pp. 142-156.

Johnson-Laird, P., Byrne, R. 1990. Deduction. Ablex Publishing.

Lamprecht E. 1981. Einführung in die Algebra, Birkhäuser Verlag.

Maier, E. 1985. Textual Relations as Parts of Multiple Links Between Text Segments. In Trends in Natural Language Generation – An Artificial Intelligence Perspective, pp. 68-87.

Mann, W., Thompson, S. 1983. Rhetorical Structure Theory: A Theory of Text Organization. Technical Report, ISI/RR-83-115, ISI at University of Southern California.

Marcu, D. 1997. From Local to Global Coherence: A Bottom-Up Approach to Text Planning. In Proc. of *AAAI-97*, pp. 629-635.

Meier, A., Pollet, M., and Sorge V. 2002. Comparing Approaches to Explore the Domain of Residue Classes. *Journal of Symbolic Computation*, 34(4), pp. 287-306.

Melis, E., Horacek, H. 2000. Dialog Issues for a Tutor System Incorporating Expert Problem Solvers. AAAA-Fall Symposium on Building Dialog Systems for Tutorial Applications.

Reiter, E. 1994. Has a Consensus Architecure Appeared, and is it Psycholinguistically Plausible? In Proc. of the *7th International Workshop on Natural Language Generation*, pp. 163-170.

Shaw, J. 1998. Clause Aggregation Using Linguistic Knowledge. In Proc. of the *9th International Workshop on Natural Language Generation*, pp. 138-147.

A New Model for Generating Multimodal Referring Expressions

Emiel Krahmer
Communication and Cognition
Tilburg University
E.J.Krahmer@uvt.nl

Ielka van der Sluis
Computational Linguistics and AI
Tilburg University
I.F.vdrSluis@uvt.nl

Abstract

We present a new algorithm for the generation of multimodal referring expressions (combining language and deictic gestures).[1] The approach differs from earlier work in that we allow for various gradations of preciseness in pointing, ranging from unambiguous to vague pointing gestures. The model predicts that linguistic properties realized in the generated expression are co-dependent on the kind of pointing gesture included. The decision to point is based on a trade-off between the costs of pointing and the costs of linguistic properties, where both kinds of costs are computed in empirically motivated ways. The model has been implemented using a graph-based generation algorithm.

1 Introduction

The generation of referring expressions is a central task in Natural Language Generation (NLG), and various useful algorithms which automatically produce referring expressions have been developed (recent examples are van Deemter 2002, Gardent 2002 and Krahmer et al. 2003). A typical algorithm takes as input a single object v (**the target object**) and a set of objects (**the distractors**) from which the target object needs to be distinguished (borrowing terminology from Dale and Reiter 1995). The task of the algorithm is to determine which set of properties is needed to single out the target object from the distractors. This is known as the **content determination** problem for referring expressions. On the basis of this set of properties a **distinguishing description** in natural language can be generated; a description which applies to v but not to any of the distractors.

We describe a new algorithm which aims at producing **multimodal** referring expressions: natural language referring expressions which may include deictic pointing gestures. There are at least two motivations for such an extension. First, in various situations a purely linguistic description may simply be too complex, e.g., because the domain contains many highly similar objects. In those cases, including a deictic pointing gesture may be the most efficient way to single out the intended referent. Second, if we look at human communication it soon becomes apparent that referring expressions which include pointing gestures are rather common (Beun and Cremers 1998). Various algorithms for the generation of multimodal referring expressions have been proposed (e.g., Cohen 1984, Claassen 1992, Huls et al. 1995, André and Rist 1996, Lester et al. 1999, van der Sluis and Krahmer 2001).[2] Most of these are based

[1]This paper greatly benefitted from discussions with Mariët Theune and Kees van Deemter. Thanks are also due to Sebastiaan van Erk, Fons Maes, Paul Piwek and André Verleg. Krahmer's work was done within the context of the TUNA project, funded by Engineering and Physical Sciences Research Council (EPSRC) in the UK, under grant reference GR/S13330/01.

[2]These algorithms all operate on domains which are in the direct visual field of both speaker and hearer. Throughout this paper we will make this assumption as well.

on the assumption that a pointing gesture is precise and unambiguous. As soon as a pointing gesture is included, it directly eliminates the distractors and singles out the intended referent. As a consequence, the generated expressions tend to be relatively simple and usually contain no more than a head noun (*this block*) in combination with a pointing gesture. Moreover, most algorithms tend to be based on relatively simple, context-independent criteria for the decision whether a pointing gesture should be included or not. For instance, Claassen 1992 only generates a pointing gesture when referring to an object for which no distinguishing linguistic description can be produced. Lester et al. 1999 generate pointing gestures for all objects which cannot be referred to with a pronoun. Van der Sluis and Krahmer (2001) use pointing if the object is close or when a purely linguistic description is too complex, where both closeness and complexity are measured with respect to a predefined threshold.

The approach described in this paper differs from these earlier proposals in a number of ways. We do not assume that pointing is always precise and unambiguous. Rather we allow for various gradations of preciseness in pointing, ranging from unambiguous to vague pointing gestures. Precise pointing has a high precision. Its scope is restricted to the target object, and this directly rules out the distractors. But, arguably, precise pointing is 'expensive'; the speaker has to make sure she points precisely to the target object in such a way that the hearer will be able to unambiguously interpret the referring expression. Imprecise pointing, on the other hand, has a lower precision —it generally includes some distractors in its scope— but is intuitively less 'expensive'.[3]

The model for pointing we propose may be likened to a **flashlight**.[4] If one holds a flashlight just above a surface, it will cover only a small area (the target object). Moving the flashlight away will enlarge the cone of light (shining on the target object but probably also on one or more distractors). A direct consequence of this "Flashlight model for pointing" is that we predict that the amount of linguistic properties required to generate a distinguishing multimodal referring expression is dependent on the *kind* of pointing gesture. Imprecise pointing will require more additional linguistic properties to single out the intended referent than precise pointing.

In our proposal, the decision to point is based on a trade-off between the costs of pointing and the costs of a linguistic description. The latter are determined by summing over the costs of the individual linguistic properties used in the description. Arguably, the costs of precise pointing are determined by two factors: the size of the target object (a big object is easier to point at than a small objects) and the distance between the target object and the pointing device (objects which are near are easier to point to than objects that are further away). As we shall see, Fitts' law —a fundamental empirical law about the human motor-system due to Fitts (1954)— can be used to model the costs of precise pointing. In addition, we shall argue that Fitts' law allows us to capture the intuition that imprecise pointing is cheaper than precise pointing.

The algorithm we describe in this paper is a variant of the graph-based generation algorithm described in Krahmer et al. (2003). It models scenes as labelled directed graphs, in which objects are represented as vertices (or nodes) and the properties and relations of these objects are represented as edges (or arcs). Cost functions are used to assign weights to edges. The problem of finding a referring expression for an object is treated as finding the *cheapest* subgraph of the scene graph which uniquely characterizes the intended referent. For the generation of multimodal referring expressions, the scene graph is enriched with edges representing the various kinds of pointing gestures. Since the algorithm looks for the cheapest subgraph, pointing edges will only be selected when linguistic edges are relatively expensive or when pointing is relatively cheap.

The rest of this paper is organized as follows. In section 2 we describe the ingredients of the multimodal graph-based approach to the generation

[3]This intuition is in line with the alleged existence of neurological differences between precise and imprecise pointing. The former is argued to be monitored by a slow and conscious feedback control system, while the latter is governed by a faster and non-conscious control system located in the center and lower-back parts of the brain (see e.g., Smyth and Wing 1984, Bizzi and Mussa-Ivaldi 1990).

[4]This analogy was suggested by Mariët Theune (*p.c.*)

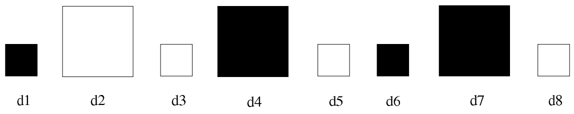


Figure 1: An example scene.

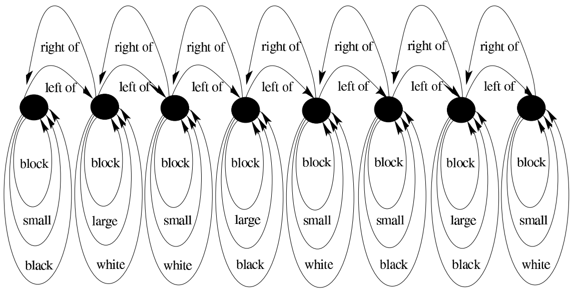


Figure 2: Example scene as a graph.

of referring expressions. Section 3 is devoted to determining the costs of linguistic properties and gestures. Section 4 describes the algorithm, and illustrates it with a worked example. In section 5, we summarize and discuss some of the properties and predictions of the model.

2 Generating multimodal referring expressions

2.1 Scene graphs Consider the visual scene depicted in Figure 1, consisting of a set of objects with various properties and relations. In this particular scene $M = \{d_1, \ldots, d_8\}$ is the set of entities, *Prop* $=$ { small, large, black, white, block } is the set of properties of these objects and *Rel* $=$ { left-of, right-of } the set of relations. We represent a scene as a **labelled directed graph**. Let L $=$ *Prop* $\cup$ *Rel* be the set of labels with *Prop* and *Rel* disjoint, then $G = \langle V_G, E_G \rangle$ is a labelled directed graph, where $V_G \subseteq M$ is the set of vertices and $E_G \subseteq V_G \times L \times V_G$ is the set of labelled directed edges.[5] Two other notions that we use in this paper are graph union and graph extension. The **union** of graphs $F = \langle V_F, E_F \rangle$ and $G = \langle V_G, E_G \rangle$ is the graph $F \cup G = \langle V_F \cup V_G, E_F \cup E_G \rangle$. If $G = \langle V, E \rangle$ is a graph and $e = (v, l, w)$ is an edge between vertices v and w and with label $l \in L$, then the **extension** of G with e (notated $G + e$) is the graph $\langle V \cup \{v, w\}, E \cup e \rangle$.

Figure 2 contains a graph representation of the scene depicted in Figure 1.[6] Notice that properties are represented as **loops**, while relations are modelled as edges between different vertices.

2.2 Referring graphs Suppose we want to generate a distinguishing description referring to d_4. Then we have to determine which properties and/or relations are required to single out d_4 from its distractors This is done by creating *referring graphs*, which at least include a vertex representing the target object. Informally, a vertex v (the target object) in a referring graph H **refers** to a given entity in the scene graph G iff the graph H can be "placed" over the scene graph G in such a way that v can be placed over the vertex of the given entity in G and each edge from H with label l can be "placed over" a corresponding edge in G with the same label. Furthermore, a vertex-graph pair is **distinguishing** iff it refers to exactly one vertex in the scene graph.[7]

Consider Figure 3, containing a number of potential referring graphs for d_4, each time with a circle around the intended referent. The first one, H_1 has all the properties of d_4 and hence can refer to d_4. It is not distinguishing, however: it fails to rule out d_7 (the other large black block). Graph H_2 *is* distinguishing. Here, the circled vertex can only be "placed over" the intended referent d_4 in the scene graph. A straightforward linguistic realization (expressing properties as adjectives and relations as prepositional phrases) would be something like "the large black block to the left of a small white block and to the right of another small

[5] Here and elsewhere subscripts are omitted when this can be done without creating confusion.

[6] We only model the direct spatial relations under the assumption that a distinguishing description would not use a distant object as a relatum when a closer one can be selected.

[7] The informal notion of one graph being "placed over" another corresponds with a well-known mathematical construction on graphs, namely **subgraph isomorphism**. $H = \langle V_H, E_H \rangle$ can be "placed over" $G = \langle V_G, E_G \rangle$ iff there exists a subgraph G' of G such that H is isomorphic to G'. H is isomorphic to G' iff there exists a bijection $\pi : V_H \longrightarrow V_{G'}$, such that for all vertices $v, w \in V_H$ and all $l \in L$:

$$(v, l, w) \in E_H \Leftrightarrow (\pi.v, l, \pi.w) \in E_{G'}$$

Given a graph H and a vertex v in H, and a graph G and a vertex w in G, we define that the pair (v, H) **refers** to the pair (w, G) iff H is connected and H is mapped to a subgraph of G by an isomorphism π and $\pi.v = w$.

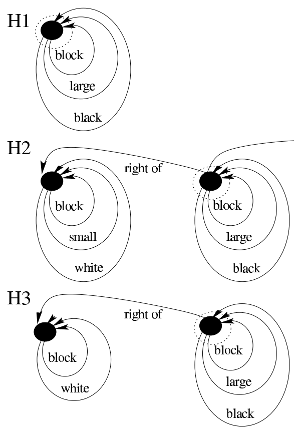


Figure 3: Three potential referring graphs for d_4.

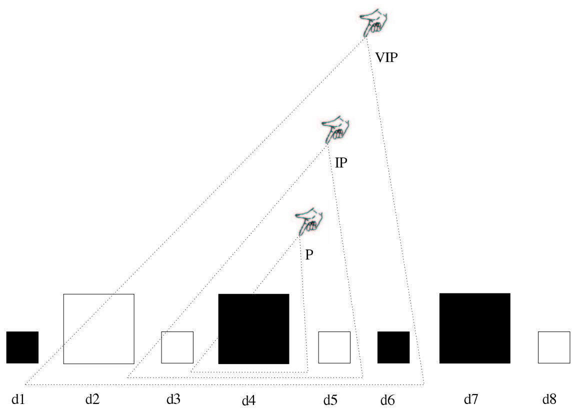


Figure 4: Pointing into the scene

white block".[8] Generally there is more than one distinguishing graph referring to an object. In fact, H_2 is not the *smallest* distinguishing graph referring to d_4. This is H_3. It might be realized as "the large black block to the right of a white block". This is a distinguishing description but not a particular natural one; it is complex and arguably difficult for the hearer to interpret. In such cases, having the possibility to simply point to the intended referent would be very useful.

2.3 Gesture graphs Suppose we want to *point* to d_4. Clearly this can be done from various distances and under various angles. The various hands in Figure 4 illustrate three levels of deictic pointing gestures, all under the same angle but each with different distances to the target object: **precise** pointing (P), **imprecise** pointing (IP) and **very imprecise** pointing (VIP). We shall limit the presentation here to these three levels of precision and a fixed angle, although nothing hinges on this. Naturally, the respective positions of the speaker and the target object co-determine the angle under which the pointing gesture occurs; this in turn fixes the 'scope' of the pointing gesture and thus which objects are ruled out by it.[9] If these respective positions are known, computing the scope of a pointing gesture is straightforward, but the actual mathematics falls outside the scope of this paper.

Just as properties and relations of objects can be expressed in a graph, so can various pointing gestures to these objects. All objects in the scope of a potential pointing gesture (with a certain degree of precision) are associated with an edge labelled with an indexed pointing gesture. Selecting this edge implies that all objects which fall outside the scope of the gesture are ruled out. We represent this information using a **gesture graph**. Let $PG_v = \{\mathsf{P}_v, \mathsf{IP}_v, \mathsf{VIP}_v\}$ be the set of pointing gestures to a target object v. Then, given a scene graph $G = \langle V_G, E_G \rangle$, a gesture graph $D_v = \langle V_G, E_D \rangle$ is a labelled directed graph, where V_G is the set of vertices from the scene graph and $E_D = V_G \times PG_v \times V_G$ the set of pointing edges. Figure 5 displays a graph modelling the various pointing gestures in Figure 4. Notice that there is one gesture edge which is only associated with d_4, the one representing precise pointing to the target object (modelled by edge P_4). No other pointing gesture eliminates all distractors.

2.4 Multimodal graphs Now the generation of multimodal referring graphs is based on the union of the scene graph G (which is relatively fixed) with the deictic gesture graph D (which varies with the target object). Figure 6 shows three distinguishing multimodal referring graphs for our target object d_4. H_1 is the smallest, only consisting of an edge modelling a precise pointing ges-

[8] A somewhat more involved lexicalization module (using aggregation) might realize this graph as "The large black block in between the two small white blocks".

[9] Here, for the sake of simplicity, we assume that an object falls inside the scope of a pointing gesture if the 'cone' shines on part of it. A more fine-grained approach might distinguish between objects in the center (where the light shines brightly) and objects in the periphery (where the light is more blurred).

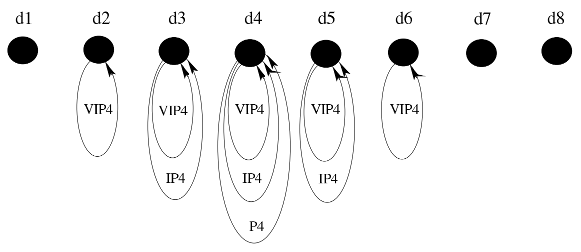


Figure 5: Deictic gesture graph

ture. It might be realized as “this one” combined with a precise pointing gesture. H_2 incorporates an imprecise pointing gesture (of the kind shown in Figure 4). Since this imprecise pointing gesture does not eliminate the distractors d_3 and d_5, a further edge is required, expressing that d_4 is black. This graph could be realized as “this black one” combined with an imprecise pointing gesture. Finally, H_3 is a distinguishing graph which incorporates a very imprecise pointing gesture. Including such an edge only rules out the distractors d_1, d_7 and d_8. At least two additional edges are required for the construction of a distinguishing graph, expressing that d_4 is both large and black. The resulting graph might be realized as “this large black one” in combination with a very imprecise pointing gesture. Arguably, in the scene of interest these multimodal referring expressions seem preferable to the linguistic expression from section 2 (*the large black block to the right of a white one*).

3 Cost functions

We now have many ways to generate a distinguishing referring expression for an object. Cost functions are used to give preference to some solutions over others. Costs are associated with subgraphs H of the scene graph G. We require the cost function to be **monotonic**. This implies that extending a graph H with an edge e can never result in a graph which is cheaper than H.[10] We assume that if H is a subgraph of G, the costs of H (notated *cost*(H)) can be determined by summing over the costs associated with the edges of H.

3.1 The costs of properties The idea that certain linguistic properties are ‘cheaper’ than others

[10]Formally, $\forall H \subseteq G \; \forall e \in E_G$: *cost*$(H) \leq$ *cost*$(H + e)$.

Figure 6: Three distinguishing multimodal referring graphs for d_4.

is already implicit in the notion of *preferred attributes* in the incremental algorithm of Dale and Reiter (1995), and is based on psycholinguistic evidence. If someone wants to describe an object, (s)he will first describe the “type” (what *kind* of object it is; a block, an animal or whatever). If that does not suffice, first **absolute** properties like color may be used, followed by **relative** ones such as size. In terms of costs, we assume that type properties (block) are for free. Other properties are more expensive. Absolute properties (colors such as black and white) are cheaper than relative ones (representing size, such as small or large). There is little empirical work on the costs of relations, but it seems safe to assume that for our example scene atomic relations are more expensive than atomic properties. First, relations are comparable to relative properties (they can not be verified on the basis of the intended referent alone). In addition, using a relation implies that a second object (the **relatum**) needs to be described as well and describing two objects generally requires more effort than describing a single object.

3.2 The costs of pointing Arguably, at least two factors co-determine the costs of pointing: (*i*) the size S of the target object (the bigger the object, the easier, and hence cheaper, the reference), and (*ii*) the distance D which the pointing device (in our case the hand) has to travel in the direction of the target object (a short distance is cheaper than a long one).[11] Interestingly, the pioneering work of Fitts (1954) captures these two factors in the *Index of Difficulty*, which states that the difficulty to reach a target is a function of the size of and the distance to a target: $ID = \log_2(\frac{2D}{S})$. Thus with each doubling of distance and with each halving

[11]A third factor which seems to be relevant is the *salience* of the target. For a detailed discussion of this aspect we refer to van der Sluis and Krahmer (2001). See also Section 5.

of size the index of difficulty increases with 1 bit. The addition of the factor 2 in the numerator is unmotivated; Fitts added it to make sure that in his experimental conditions the *ID* was always positive. He performed three experiments (a tapping, a disk transfer and a pin transfer task) and in all three found a high correlation between the time subjects required to perform the task and the index of difficulty. In recent years various alternatives for the original *ID* have been proposed. MacKenzie's (1991) alternative removes the unmotivated 2 from the numerator and starts counting from 1 assuring that the *ID* is always positive.

$$ID = \log_2(\frac{D}{S} + 1)$$

MacKenzie shows that this version of the *ID* fits the experimental data slightly better. Below we derive the costs of pointing from this index of difficulty. As argued, it seems a reasonable assumption that *imprecise* pointing is cheaper than precise pointing; it rules out fewer distractors, but also requires less motoric precision and effort from the speaker. The index of difficulty allows us to capture this intuition. We do not interpret the distance D as the distance from the neutral, current position of the hand to the target object, but rather as the distance from the current position of the hand to the target position of the hand. For the imprecise variants of pointing this distance will be smaller and hence the index of difficulty will be lower.

4 Sketch of the algorithm

In this section we describe an algorithm which outputs the cheapest distinguishing graph for a target object, and illustrate it with an example. Whether this cheapest graph will include pointing edges, and if so, of what level of precision, is determined by a trade-off between the costs of the linguistic edges representing properties and relations of the target object and the costs of pointing. The algorithm is a multimodal extension of the algorithm described in Krahmer et al. (2003), to which paper we refer for more details about complexity, motivation and implementation.

Suppose we want to generate a description for d_4 from the scene graph G in Figure 2. Before we illustrate the workings of this function we need to

```
**makeReferringExpression**($v$, $G$) {
    **construct** $D_v$;
    $M := D_v \cup G$;
    *bestGraph* := $\perp$;
    $H := \langle\{v\}, \emptyset\rangle$;
    **return findGraph**($v$, *bestGraph*, $H$, $M$); }

**findGraph**($v$, *bestGraph*, $H$, $M$) {
    **if** [*bestGraph* $\neq \perp$ **and** *cost*(*bestGraph*) $\leq$ *cost*($H$)]
    **then return** *bestGraph*;
    *distr* := $\{n \neq v \mid n \in V_M \wedge (v, H)$ refers to $(n, M)\}$;
    **if** *distr* = $\emptyset$ **then return** $H$;
    **for** each adjacent edge $e$ **do**
        $I$ := **findGraph**($v$, *bestGraph*, $H + e$, $M$ );
        **if** [*bestGraph* = $\perp$ **or** *cost*($I$) $\leq$ *cost*(*bestGraph*)]
        **then** *bestGraph* := $I$;
    **return** *bestGraph*; }
```

Figure 7: Sketch of the algorithm.

specify a cost function. Let us assume that d_4 is a cube with sides of 1 inch, and that 31 inches is the distance from the current neutral position of the hand to the target position required for precise pointing, 15 inches for imprecise pointing and 7 inches for very imprecise pointing. Some easy calculations will show that the index of difficulty in the three cases is 5 bits, 4 bits and 3 bits respectively. Thus, precise pointing (P) costs 5.00 points, imprecise pointing (IP) 4.00 and very imprecise pointing (VIP) 3.00. The preferred order for attributes in the current domain is (1) type, (2) color, (3) size and (4) relations. In terms of costs, let us assume for the sake of illustration that type edges (block) are for free, color edges cost 0.75, size edges cost 1.50 and relational edges 2.25.

We call the function **makeReferringExpression** (d_4, G), outlined in figure 7. First of all the deictic gesture graph D_{d_4}, adding pointing edges of various levels of precision to d_4, is constructed (see Figure 4), and merged with G. This gives us a multi-modal graph M. The variable *bestGraph*, for the cheapest solution found so far, is initialized as the undefined graph $\perp$ (no solution was found yet), and the referring graph under construction H is initialized as the graph only consisting of the vertex d_4. We call the function **findGraph** with as parameters the target object d_4, the best graph so far ($\perp$), the graph under construction H and the multi-modal graph M. Now the algorithm systematically tries all relevant subgraphs H of M. It starts from the graph which only contains the vertex d_4 and the algorithm recursively tries to extend this graph by adding *adjacent* edges (that is edges which start in d_4 or possibly in any of the other vertices added later on to the H under construction). For each graph H it checks to which objects in M (different from d_4) the vertex-graph pair (d_4, H) may refer; these are the *distr*actors. As soon as this set is empty we have found a distinguishing graph referring to d_4. This graph is stored in the variable *bestGraph* for the cheapest distinguishing graph found so far. In the end the algorithm returns the cheapest distinguishing graph which refers to the target object, if one exists, otherwise it returns the undefined null graph $\perp$. In the current set up the latter possibility will never arise due to the presence of unambiguous pointing gestures (expensive though they may be). Which referring graph is the first to be found depends on the order in which the edges are tried (clearly this is a place where heuristics are helpful, e.g., it will generally be beneficial to try cheap edges before expensive ones). Let us say, for the sake of argument, that the first distinguishing graph which the algorithm finds is H_3 from Figure 3. This graph costs 5.25. At this point, graphs which are as expensive as this graph can be discarded (since due to the monotonicity constraint they will never end up being cheaper than the best solution found so far). In the current situation, the cheapest solution is H_2 from Figure 6, which costs a mere 4.75.[12] The resulting graph could be realized as "this black one" combined with an imprecise pointing gesture.

5 Discussion

We have described a new model for the generation of multimodal referring expressions. The approach is based on only a few, independently motivated assumptions. The starting point is a graph-based algorithm which tries to find the cheapest referring expression for a particular target object (Krahmer et al. 2003). We assume that linguistic properties have certain costs (c.f., the preferred attributes from Dale & Reiter 1995). And, finally, we propose a "flashlight" model of pointing allowing for different gradations of pointing precision, ranging from precise and unambiguous to imprecise and ambiguous. The costs of these various pointing gestures are derived from an empirically motivated adaptation of Fitts' (1954) law.

The model has a number of nice consequences. We have described two in detail: (1) we do not need an *a priori* criterion to decide when to include a pointing gesture in a distinguishing description. Rather the decision to point is based on a trade-off between the costs of pointing and the costs of a linguistic description. And (2) we predict that the amount of linguistic properties required to generate a distinguishing multimodal referring expression is dependent on the *kind* of pointing gesture. One further neat consequence of the model is that an isolated object does not require precise pointing; there will always be a graph containing a less precise (and hence cheaper) pointing edge which has the same objects in its scope as the more precise pointing act. Notice also that the algorithm will never output a graph with multiple pointing edges, since there would always be a cheaper graph which omits the less precise one. In most situations, it will also not happen that a distinguishing graph will include both an imprecise pointing gesture and a relational edge. Under most cost functions it will be more 'cost effective' to include a precise pointing edge than an imprecise pointing edge *plus* a relational edge *plus* the edges associated with the relatum.

The algorithm we have described has been implemented in Java 2 (J2SE, version 1.4). The computation described in section 4 requires 110 ms. on a PC with a 900 mHz AMD Athlon Processor and 128 Mb RAM. Due to the presence of precise pointing edges it will always be possible to single out one object from the others. As a side effect of this we obtain a polynomial upperbound for the theoretical complexity.[13] It has been argued that

[12]Note that if pointing would have been cheaper (because the distance between the current position of the hand and the required position for precise pointing was, say, 3 inches), the algorithm would output "this one" plus a precise pointing edge (i.e., H_1 from Figure 6, for 2.00). If pointing would be more expensive (because even for very imprecise pointing the distance would be substantial), the algorithm would output H_3 from Figure 3, for 5.25.

[13]We know the costs of at least one distinguishing graph

some notion of *focus of attention* could be used to tackle the computational complexity. We may assume that objects which are currently in the focus of attention are more salient than objects which are not in focus. Now the distractor set for a target object need not include *all* objects in the domain, but only those that are at least as salient as the target object. A distinguishing description only needs to rule out those objects. There are two interesting connections between focus of attention and multimodality. First, pointing gestures typically serve to demarcate the focus of attention. Second, the model described in this paper predicts that a distinguishing description for an object which is salient is less likely to contain a pointing gesture. If an object is salient, this generally implies that its distractor set is relatively small (typically, only a few objects are somehow salient). This in turn implies that fewer (or less expensive) edges are required to rule out the distractors, hence there is less need for deictic pointing gestures.

It is interesting to observe that, even though we borrow the idea of preferred attributes from the Incremental Algorithm (arguably the most influential algorithm for the generation of referring expressions), an incremental approach to multimodal descriptions does not seem to be straightforward. One might consider extending the list of preferred attributes with VIP, IP and P (in that preference order, modelling the increase in costs). On this approach, we would first select a number of linguistic edges (independent of the kind of pointing gesture) followed by one or more pointing edges. But that would not work, since the lack of backtracking (which is inherent to incrementality) entails that all selected properties will be realized. This seems to suggest that the model outlined in this paper is inherently non-incremental.

We are currently running an experimental evaluation of the model, particularly addressing the vague pointing gestures and their interaction with linguistic realization. We hope to present the results of this evaluation in a sequel to this paper.

for our target object; the graph consisting of only a vertex for the target object and a precise pointing edge. This means that we do not have to inspect all subgraphs of the merged multimodal graph M, but only those subgraphs which do not cost more than the precise pointing graph. Thus, we only need to inspect graphs with less than K edges (for some K depending on the costs of precise pointing), which requires in the worst case $\mathcal{O}(n^K)$, with n the number of edges in the graph M. It should be added that this worst case complexity is computationally rather unattractive for larger values of K.

References

André, E. and T. Rist (1996), Coping with Temporal Constraints in Multimedia Presentation Planning, *Proceedings of the 13th AAAI*, 142–147.

Beun, R.J. & A. Cremers (1998), Object reference in a shared domain of conversation, *Pragmatics & Cognition* **6**(1/2):121–152.

Bizzi, E. and F. Mussa-Ivaldi (1990), Muscle properties and the control of arm movement, In: *Visual Cognition and Action (vol 2)*, D. Osherson, et al. (eds.), MIT Press.

Cohen, P. (1984), The pragmatics of referring and the modality of communication, *Computational Linguistics* **10**(2):97–125.

Claassen, W. (1992), Generating referring expressions in a multimodal environment, in: *Aspects of Automated Natural Language Generation*, R. Dale et al. (eds.), Springer Verlag, Berlin.

Dale, R. and E. Reiter (1995), Computational interpretations of the Gricean maxims in the generation of referring expressions, *Cognitive Science* **18**:233–263.

van Deemter, K. (2002), Generating referring expressions: Beyond the Incremental Algorithm, *Computational Linguistics* **28**(1):37–52.

Fitts, P. (1954), The information capacity of the human motor system in controlling amplitude of movement, *Journal of Experimental Psychology* **47**:381–391.

Gardent, C. (2002), Generating minimal definite descriptions, *Proceedings of the 40th ACL*, Philadelphia, USA.

Huls, C. E. Bos & W. Claassen (1995), Automatic referent resolution of deictic and anaphoric expressions, *Computational Linguistics* **21**(1):59–79.

Krahmer, E. S. van Erk & A. Verleg (2003), Graph-based Generation of Referring Expressions, *Computational Linguistics*, **29**(1): 53–72.

Lester, J., J. Voerman, S. Towns and C. Callaway (1999), Deictic Believability: Coordinating gesture, locomotion, and speech in lifelike pedagogical agents, *Applied Artificial Intelligence* **13**(4-5):383–414.

Smyth, M. and Wing, A. (1984), *The Psychology of Human Movement*, New York: Academic Press.

MacKenzie, I.S. (1991), *Fitts' law as a performance model in human-computer interaction*, doctoral dissertation, University of Toronto, Canada.

van der Sluis, I. and E. Krahmer (2001), Generating Referring Expressions in a Multimodal Context: An empirically motivated approach. *Selected Papers from the 11th CLIN Meeting*, W. Daelemans et al. (eds.), Rodopi, Amsterdam.

Applied NLG system evaluation: FlexyCAT

Nestor Miliaev, Alison Cawsey and Greg Michaelson
Department of Computer Science
Heriot-Watt University
{ceenym, alison, greg}@macs.hw.ac.uk

Abstract

Evaluation is an important part of NLG projects, however NLG system evaluation often consists of usability or static text quality assessment. This paper presents an NLG system, FlexyCAT, and experiments that enabled us to evaluate the degree of knowledge re-use and the task-specific value of generated texts.

1 Introduction

Many applied natural language generation (NLG) systems have been created recently producing various kinds of texts for different applications. However, only a few of these systems have been formally evaluated, and the evaluation performed have focused on the grammaticality and fluency of the output text, rather that on its effectiveness (Colineau et al., 2002). Issues such as knowledge re-use were largely ignored in these projects.

This paper describes the evaluation of an NLG system Flexible Computer-Aided Technical Writer (FlexyCAT) focusing on the assessment of the task-specific quality of generated texts and knowledge re-use when the system is used for the description of different devices. This work has shown, firstly, that the task-oriented quality of generated texts can be comparable to that of human-crafted texts; and secondly, how knowledge re-use allows us to extend the applicability of an NLG system for the description of different technical systems and to reduce time taken to create a document.

2 Existing practise of NLG System Evaluation

Recently, it had become widely accepted that work in NLG should pay closer attention to the evaluation of results. The aspects of an NLG system to evaluate and the metrics to use for this are defined by the goals of the NLG system. The most common advantages of using NLG comparable with, e.g., Machine Translation (MT) are considered the following (Reiter and Dale, 2000):

- High quality output text that is generated based on machine data and does not require post-processing
- Simultaneous production of text versions in different languages
- Consistency of text between the versions and with a domain model
- Lower cost and time of text revision (e.g., when domain changes)
- Dynamic text generation upon user query
- Potentially good knowledge re-use

Unfortunately, most NLG systems evaluation carried out to date did not cover all these aspects. Usually NLG evaluation deals with 1) System quality; 2) Text quality.

System quality evaluation consists of measuring characteristics like usability, generation speed and system robustness. Such evaluation follows common patterns of software systems evaluation, see e.g., (Newman and Lamming, 1995). NLG-specific principles and metrics of system evaluation are described in (Mellish and Dale, 1998) and (EAGLES, 1995).

One of the most thorough usability evaluation of an NLG system has been carried out in the AGILE project; the course of experiments and their results are presented in (Hartley et al., 2000).

Assessing the quality of a generated text (based on (Mellish and Dale, 1998) and (Hartley et al., 2000)) includes rating of the text against such criteria as *Accuracy*, *Fluency* (including *Acceptability* and *Grammaticality*) and *Lexico-grammar coverage*.

Methods of evaluation appropriate to these criteria can be grouped into three classes, as suggested in (Bangalore et al., 1998):

Intrinsic that typically consists in asking human judges to rate the quality of generated texts. Intrinsic evaluation is most common, and was carried out e.g., in AGILE (Hartley et al., 2000) and MIRADOR (Eddy and Cawsey, 2002); it is fairly simple and straightforward, however depends a great deal on the evaluators' expertise and personal preferences.

Extrinsic or *task evaluation*, where the user's ability to perform some task using a generated text or the impact of the text on the user behaviour is assessed. Extrinsic evaluation is less common because experiments are more difficult to carry out. However, it allows us to estimate the task-oriented worth of a document. This kind of evaluation was used in Isolde (Colineau et al., 2002), STOP (Reiter et al., 2001), and other NLG projects.

Comparative where the objective is to directly compare the performance of different generation systems and formalisms. Comparative evaluation is not often used because of its complexity. An example of this kind of trial for the XTAG project is described in (Bangalore et al., 1998).

Other aspects of NLG, such as knowledge reuse, time of document creation or system flexibility have not received enough attention so far. Although there were some reports on task-oriented evaluation, these experiments are not common and many of those did not include the comparison of the user performance on human-crafted and generated manuals, like in IDAS (Levine and Mellish, 1995).

During FlexyCAT evaluation, we paid extra attention to task-oriented text quality, knowledge reuse and the benefits the author obtains by using a flexible planning technique.

3 FlexyCAT: System Description

FlexyCAT is an NLG system for producing manuals for technical devices, mainly home appliances, e.g., TVs, VCRs, cameras, etc. Manuals consist of texts divided up into sections, each of which describes an individual procedure, consisting of a sequence of steps that the user is to perform to attain his/her goal. No generation of explanatory or warning information is included; generated texts are purely instructional and are thus like the 'minimalist instructions' described in (Colineau et al., 2002).

Texts are generated in two languages (English and Russian) given a domain model. The domain model represents an object-oriented description of a device being documented. Such a representation contains a description of all device elementary constituents and their functioning. The description of the functioning of elementary objects includes events (user actions) and object functions, that have preconditions and actions. Both preconditions and actions are described in terms of elementary object properties. Objects, events and actions are represented using linguistic classes. A generation system uses both linguistic classes and domain knowledge for the production of a grammatical manual text in two languages. The output of FlexyCAT is a ready to use manual for the device. FlexyCAT's GUI provides the user with facilities for creating and editing a domain model and dictionaries of linguistic classes. It also allows the user to easily assign linguistic values to the elements of the domain model. For more information see (Miliaev et al., 2002).

Two main advantages of FlexyCAT compared with other NLG systems are:

- FlexyCAT is an integrated tool allowing extensive editing of both linguistic and domain knowledge. This, and an object-oriented design of the knowledge, enables, firstly, good knowledge re-use, and, secondly, the production of manuals for a variety of different technical devices using a single NLG tool.
- FlexyCAT offers a flexible approach to text planning. The planner produces a candidate text plan automatically, based on the domain model. When there is a necessity to obtain a better quality text, the user edits this plan using an interactive planning utility. A text is generated based on the text plan. For automatically generated plan it is of a draft quality, while a better quality text is obtained from an edited plan. The automatic feature facilitates the work of the technical author by producing a text draft quickly and at fairly little cost.

An example original text from the corpus is given below; it contains a considerable proportion of explanatory and causative information:

```
How to play a CD
1. Press the PLAY/PAUSE button
The CD will begin to play and
the track number will be shown
in the display
2. To pause the CD press the
PLAY/PAUSE button
The CD will stop and the PAUSE
INDICATOR will flash in the
display
```

An automatically generated corresponding text looks like this:

```
Playback CD
1. Press PLAY/PAUSE button to
start CD playback
2. Press PLAY/PAUSE button to
pause CD
```

After the plan refinement, we get the following text, with richer rhetorical structure (the rhetorical relation in the first sentence is changed from 'motivation' to 'means'; in the second sentence the sequence of the nucleus and satellite clauses changed) and, we believe, a better quality:

```
Playback CD
1. Start CD playback by pressing
PLAY/PAUSE button
2. To pause CD, press PLAY/PAUSE
button
```

Below we will describe our experience and results of the evaluation of the FlexyCAT NLG system.

4 FlexyCAT Evaluation

The evaluation of FlexyCAT is ongoing. This section describes evaluation experiments that have been carried out to date and their results.

So far, three stages of the evaluation have been performed. These are:

- Experiments on knowledge re-use and text production for different devices
- Subjective assessment of text quality
- Task-oriented text quality evaluation

4.1 Knowledge Re-use and Resource Management

This experiment was targeted, firstly, to assess the appropriateness and effectiveness of the chosen domain description structure and the domain engineering tool for documenting different technical devices. The second goal was to assess the degree of knowledge (both linguistic and domain) re-use.

The experiment was set up as follows. Manual texts, each about 3 pages long, were selected for three different pieces of hardware. All texts contained sections, subsections and paragraphs. FlexyCAT was used to build a domain model for the pertinent subset of each of these devices and manual texts were generated in both English and Russian. The interactive planning utility was used to obtain text as close to the original manual as possible, in terms of wording and word order, see the description of the metrics for generation accuracy and their importance in (Hartley et al., 2000). In many cases the generated texts were identical to the original ones. The time taken to complete the entire process of document creation was measured and the number of knowledge base (KB) elements re-used was estimated for each task.

The course of the experiment was as follows. A domain model and a manual for the first VCR (VCR1) was created. Then the manual for a similar device, VCR2 was created. Both manuals contained similar set of functions (however the controls and operation were different in some cases). During the creation of the manual for VCR2 both linguistic and domain resources created during the production of the manual for VCR1 were available and were re-used.

The second phase of the experiment consisted in the creation of a manual for a fairly distant device, a combined CD-Radio-Cassette (Combine). That device contained parts similar to that of a VCR, namely a tape recorder and a CD player; and a part different from any known in a VCR, a radio tuner. The manual for Combine was created twice – once from scratch without using any of the existing resources, and a second time making use of the resources created for VCR1. The role of a technical author was played by the system author, who knows the system very well and had created manuals for these and similar devices number of times in the course of the system development and evaluation. Thus, the learning effect that may have affected time of subsequent document creation may be largely ignored.

FlexyCAT uses three classes of linguistic primitives: nominal expressions (consisting of up to two nouns), verbs and adverbials. Adverbials were not considered in the evaluation experiments. There exist two versions of each nominal expression and verb, one for English and one for Russian. Complex classes, that encapsulate the references to both versions, are called *Nominals* and *Verbals* respectively. *Nominals* denote nominal expressions that could be used to name a domain entry. *Verbals* are used to specify events and actions in a domain model. A *Domain object* represents a single elementary constituent of a device; an object's description may include a number of references to nominals and verbals.

Figures 1, 2 and 3 show the results of knowledge re-use which occurred in the course of the experiment.

KB element	**Reused**	**Total**	**Percentage**
Nominals	30	84	36%
Nominals*	12	84	14%
English nouns	41	83	49%
Russian nouns	52	87	60%
Verbals	32	46	70%
English verbs	31	38	82%
Russian verbs	33	35	94%
Domain objects	23	38	61%

Figure 1: Knowledge re-use for VCR2

In Figures 1, 2 and 3 Nominals* denote nominals that were amended slightly for the description of new objects in a new device.

Domain objects almost always underwent change to conform the description of a new device. 'Re-used' objects, hence, are those that were amended only slightly.

KB element	**Reused**	**Total**	**Percentage**
Nominals	12	24	50%
Nominals*	5	24	21%
English nouns	23	30	77%
Russian nouns	23	27	85%
Verbals	19	24	79%
English verbs	18	18	100%
Russian verbs	20	21	95%
Domain objects	10	15	67%

Figure 2: Knowledge re-use for CD: Tape recorder sub-set

KB element	**Reused**	**Total**	**Percentage**
Nominals	13	51	25%
Nominals*	8	51	16%
English nouns	28	58	48%
Russian nouns	30	59	51%
Verbals	22	31	71%
English verbs	23	31	74%
Russian verbs	23	34	68%
Domain objects	15	28	54%

Figure 3: Knowledge re-use for CD: whole device

As can be seen from Figures 1 - 3, the percentage of knowledge re-use is fairly high. By that, the percentage is higher between similar devices and increases for simpler elements; nouns and verbs, show the best degree of re-use, with verbs in some cases being re-used by 100%. This could be explained by the fact that the way controls operate are fairly persistent across many devices. Nominals are re-used a little less. That is caused mainly by the difference in control names in different devices.

For the comparison of time taken to produce documents using FlexyCAT it can be assumed that manuals for VCR1 and VCR2 are pretty much identical in size and labour-intensity to produce. The manual for Combine is slightly shorter. That explains the necessity of producing it from scratch first – we needed it to assess the initial time required to compare it with that when re-using existing resources. Figure 4 shows the time measures for each experiment.

Experiment	**Time**
VCR1, no resource exists	580 mins
VCR2, VCR1 resource used	340 mins
Combine, no resource used	430 mins
Combine, VCR1 resource used	270 mins

Figure 4: Time of manual creation in FlexyCAT

As can be seen, the time to create a manual is being reduced significantly when re-using existing resources. This suggests good knowledge re-use and effort reduction when making use of existing resources, even for a fairly different device.

4.2 Subjective Assessment of Text Quality

An experiment on subjective assessment of text quality was conceived as a pilot study for the task-oriented text quality evaluation. However, it has yielded some interesting results, especially in comparison with the latter, so we will refer to both.

The experiment was set up as follows. We used a within subject design. A group of nine native speakers of English were offered a set of nine text excerpts each. Text excerpts were short instructions up to half a page long. Texts were given in three different versions: original manual text; automatically generated draft; and a text generated after plan editing (further regarded as 'original', 'generated' and 'edited' respectively). See text examples in section 3. Each set contained only one version of each text (either original, automatically generated or generated after plan editing). Text versions were evenly distributed across different sets and the evaluators did not know which version of each text excerpt s/he got. Illustrations from original manuals were included in all versions, where applicable. Text layout was the same in all versions.

The evaluators were asked to assess quality and understandability of each text on the scale from 1 to 5 (1 is bad, 5 is excellent).

The average scores and standard deviation across different document versions were as follows, see Figure 5.

Text quality	
Original	4.07(0.92)
Generated	4.19(0.79)
Generated+edited	4.00(0.68)
Text understandability	
Original	3.85(1.03)
Generated	3.93(1.04)
Generated+edited	3.81(1.08)

Figure 5: Mean(standard deviation) of subjective scores of text quality

This figure shows that different text versions were ranked quite closely, which lets us conclude that all text versions are similar from the reader's point of view. This is similar to the results obtained in the AGILE evaluation.

Surprisingly, automatically generated text drafts (having no diverse sentence structures and lacking rhetorical markers) have been preferred over other text versions. However, a t-test has shown that the difference in user preference is not significant. The results of the t-test at df=57 are: t=-0.408, p=0.685 between original and generated; t=0.272, p=0.7866 between original and edited; and t=-0.680, p=0.4993 between generated and edited texts.

After this pilot study, a larger scale task-oriented experiment was run.

4.3 Task-oriented text quality evaluation

This time the user had to carry out real tasks with actual pieces of hardware, instead of subjectively assessing text qualities. There were three sets of hardware: VCR1+TV, VCR2 and a CD player.

The tasks included: 1) Assemble all components of VCR1+TV; start, stop and eject the videotape; 2) Program VCR1 to automatically record a certain program; 3) Set clock of VCR2 to a certain time; 4) Power a CD player, load a CD, start playback and program the CD player to play certain tracks.

As in the pilot study, sets of manuals were prepared, produced according the same principles; the only difference was that this time the manuals were longer– up to 3 pages. There were 21 evaluators – seven for each set of manuals.

The users were encouraged to use the manual as much as possible, but they were free to use their prior knowledge as well. However, because of the complexity of the tasks the users had to use manuals for almost all tasks.

The user performance of each task was timed. At the end of each task, the users were asked to assess the following: "Task difficulty", "How useful was manual to cope with the task" and "Quality of the manual text". The scale was 1 to 5 (1 is an easy task or useless manual or bad text quality; 5 is a difficult task, helpful manual or good text quality respectively). Also the users were encouraged to give their informal comments about tasks and used manuals at the end of the experimental session.

Figure 6 shows average time taken to perform tasks given different sets of manuals and standard deviation of the task accomplishment time. As can be seen, the time of task completion varied greatly between different tasks and users.

Task.No	**Orig**	**Gen**	**Edit**
1	3:37(0:58)	3:19(0:47)	2:58(0:31)
2	6:02(1:26)	5:53(1:21)	4:38(1:11)
3	3:59(1:17)	2:43(0:58)	4:20(2:26)
4	4:29(1:44)	3:48(0:50)	2:57(0:37)

Figure 6: Mean(standard deviation) of task accomplishment time

While performing the first task the users often did not resort to the manual, so we discard the results of this test from the further discussion as not being reliable. After normalising experiment time (mean time(deviation) is 1.10(0.22), 0.96(0.29) and 0.98(0.48) for original, generated and edited texts respectively), a t-test has been done to estimate the difference in user performance depending on the text version used.

The results of the t-test indicate that users performed faster on generated versions of the manuals; however the difference in the user performance on different text versions is not significant, only approaching the level of significance between original and generated texts. The results at df=40 are: t=1.731, p=0.0912 between original and generated; t=1.032, p=0.3083; and t=0.152, p=0.880 between edited and generated text versions.

The small dependency of the task accomplishment time on the text version may suggest that users seldom pay much attention to the text detail, preferring to quickly skim through the text looking for the pertinent information or use the diagrams. Any attempts to improve text quality (adding rhetorical markers, etc.) in our case made little difference to the user performance.

The results of user assessment of the text quality/usefulness after performing their tasks are presented in Figure 7. This figure shows average score of text quality and usefulness for completing the task and standard deviation of these scores.

Text quality	
Original	3.45(0.80)
Generated	2.75(1.36)
Generated+edited	3.25(1.62)
Text usefulness	
Original	3.90(0.92)
Generated	3.25(1.18)
Generated+edited	3.90(1.64)

Figure 7: Mean(standard deviation) of text quality/usefulness results

As is the case with the time of task accomplishment, scores varied a lot between different users.

The results of a t-test of text quality at df=40 are: t=1.936, p=0.06 between original and generated; t=0.481, p=0.633 between original and edited; and t=1.030, p=0.3092 between edited and generated texts.

The results of a t-test of text usefulness at df=40 are: t=1.898, p=0.065 between original and generated; t=0, p=1 between original and edited; and t=1.403, p=0.168 between edited and generated texts.

These results show that on average the users regarded the quality and usefulness of the original texts being equal or slightly better than these of the generated or edited texts. However the difference is not significant.

The comparison of data in Figures 6 and 7 gives rise to an interesting paradox. The users have assessed the quality and usefulness of original manuals slightly higher than of their generated counterparts. Nonetheless, they performed slightly better on the generated versions. This shows that the subjective judgement of text quality may not reliably represent its task-oriented worth. A similar opinion was expressed by the participants of the AGILE evaluation that 'it is difficult to evaluate the acceptability of a technical instruction text *per se*, without real knowledge of the ... system it describes'.

Another discrepancy is that in the pilot study the users preferred generated texts over other versions, whereas in the main experiment original texts were favourites. We do not have other explanation than either users' subjectivity or a low number of participants that played role. The first premise indicates the insufficiency of subjective methods of text quality evaluation; the second one entails us to re-do the experiments with greater number of participants.

However, any differences in user performance and text quality scores described above were insignificant, which means that both subjective and task-specific quality of all manual versions was pretty much equal.

5 Evaluation Results

The results obtained in the course of evaluation of FlexyCAT have confirmed again the advantages of NLG systems and emphasised the aspects which had previously received less attention.

Firstly, FlexyCAT has proved a versatile tool allowing us to create manuals for different devices. The extent of knowledge re-use (both linguistic and domain) is very promising not only across similar devices, but even between fairly distant ones.

Re-using existing knowledge also allows the user to save valuable time in the production of subsequent documents. We have found that the time of manual creation in two languages, when re-using existing linguistic and domain resources, is comparable to that of manual document creation. These results suggest that NLG could become a promising competitor to other ways of multilingual document creation in terms of time and effort saving.

Secondly, the users did not show any significant preference of any text version over others, not did they perform significantly better on a certain version of text. This is a very promising result suggesting that the quality of generated texts was as good as that of manually-crafted ones.

Thirdly, time of task completion varies a great deal between different users, but generally depends little on the manual text quality. An attempt to improve text quality by enriching its rhetorical structure made little difference to the user performance. This also indicates that text *fluency* matters little for accomplishing user task. Sometimes users subjectively assessed a manual as having poor quality, nonetheless finding it useful for completing their task – 'Bad manual is better than no manual'.

Fourthly, a subjective evaluation of text quality may not be a true indication of manual worth with regard to performing a task. Task-oriented evaluation is a more objective way of assessing how helpful the manual is.

The collected data and informal comments have given rise to the following suggestions to improve manual quality, that support those found in (Haydon, 1995):

Firstly, the consistency and structure of a text are very important. All pertinent information should be presented in an overt form and at the place where it is vital.

Secondly, small pieces of text are better than big dense chunks. Each piece should describe a single function.

6 Conclusions and Further Work

In the course of FlexyCAT evaluation some novel experiments were carried out that have shown that the NLG system may be used to produce texts for different devices and that it is possible to achieve good knowledge re-use, that allows significant saving of time and effort for the production of subsequent manuals.

The text quality experiments have indicated that an NLG system is capable of producing good quality texts. Our task-oriented experiment have shown the advantages of using generated manuals for performing a task and that that subjective methods of assessing text quality are not always adequate for the estimation of text usefulness.

Although FlexyCAT is a bilingual text generation tool, so far it has not been possible to assess the Russian part of it because of the lack of both original manuals in Russian and of Russian-speaking evaluators. All experiments described in this paper are pertinent to the English part of the generation; it was assumed that Russian texts have comparable quality and properties. Further experiments to assess the quality of Russian texts and their consistency with English ones are required.

Also, larger-scale experiments are desirable to evaluate the our conclusions.

This work has indicated the importance of evaluation in NLG, especially task-oriented evaluation and the necessity of broadening of scopes of NLG evaluation.

Acknowledgements

We thank Dr. Diana Bental for her help with statistical processing of experimental data.

References

Srinivas Bangalore, Anoop Sarkar, Christine Doran, and Beth Ann Hockey. 1998. Grammar and Parser Evaluation in the XTAG Project. In *Workshop on The Evaluation of Parsing Systems*, Granada, Spain.

Nathalie Colineau, Ccile Paris, Keith Vander Linden. 2002. An Evaluation of Procedural Instructional Text. In *Proceedings of INLG 2002. pp 128-135.*

1995 EAGLES, Evaluation of Natural Language Processing Systems. FINAL REPORT, EAGLES DOCUMENT EWG-PR. Version of September 1995.

Bruce Eddy and Alison Cawsey. 2002. Balancing Conciseness, Readability and Salience in Generated Text. In *Proceedings of the 3rd International Workshop on Natural Language and Information Systems*, Aix-en-Provence, France.

Antony Hartley, Donia Scott, I. Kruijff-Korbayouva, Serge Sharoff, E. Teich, Lena Sokolova, Kamenka Staykova, Danail Dochev, Martin Cmajrek, and Jiri Hana. 2000. Evaluation of the final prototype. Technical report, Brighton University, October, 12.

Leslie M. Haydon July 1995. The Complete Guide to Writing and Producing Technical Manuals. John Wiley and Sons, Ltd.

John Levine and Chris Mellish. 1995. The IDAS User Trials: Quantitative Evaluation. In *Proceedings of the 5th European Workshop on NLG*, pages 75–93. Rijks Universiteit Leiden.

Chris Mellish and Robert Dale. 1998. Evaluation in the context of natural language generation.

Nestor Miliaev, Alison Cawsey, and Greg Michaelson. 2002. Technical Documentation: An Integrated Architecture for Supporting the Author in Generation and Resource Editing. In *The Tenth International Conference on Artificial Intelligence: Methodology, Systems, Applications AIMSA 2002*, Varna, Bulgaria, September 4-6. Springer-Verlag.

Ehud Reiter and Robert Dale. 2000. *Building Applied Natural Language Generation System.* Natural Language Engineering. Cambridge University Pres.

Ehud Reiter, Roma Robertson, A. Scott Lennox, and Liesl Osman. 2001. Using a Randomised Controlled Clinical Trial to Evaluate an NLG System. In *In Proceedings of ACL-2001*, pages pp. 434–441.

William M. Newman, Michael G. Lamming 1995. Interactive System Design. Addison-Wesley Pub Co. 1st edition.

A Phrasal Generator for Describing Relational Database Queries

Michael J. Minock
mjm@cs.umu.se
The University of Umeå
Umeå, Sweden

Abstract

This paper proposes a technique to generate single sentence natural language descriptions for a wide class of relational database queries. Such a capability meets an important need in the area of cooperative information systems.

The approach to describing queries is phrasal and is restricted to tuple relational queries using positive or negatively signed sequences of existential quantifiers over conjunctions of conditions. Query containment and equivalence are decidable for this class and this property is exploited in the maintenance and use of the phrasal lexicon.

1 Introduction

Often relational database schemas are delivered as collections of oddly named tables and attributes. Users and administrators are expected to query, integrate and otherwise maintain such systems. Natural language generation has been seen as an important part of improving the understandability of relational database schemas(McKeown, 1985).

The focus here, however, is not to explain or describe database schemas, but rather to describe database queries. While at first this may seem to be of limited value, we shall see that many techniques in cooperative information systems(Gaasterland et al., 1992) require 'query' descriptions as an integral part of their communication process. The 'query' being described is not usually the user's own query, but rather some derived expression that may be written in the form of a query. A set of cooperative techniques that require such description services shall be reviewed in this paper. We view the cooperative information system that bundles these techniques, as essentially providing the services of a strategic text planner. The cooperative information system decides communication content with communication acts consisting of template sentences with embedded requests for 'query' descriptions.

Given that we seek to describe queries, we must contend with the fact that there are infinitely many syntactically correct queries over a given schema. It is critical that the generation system provide adequate *coverage* over some *well defined* portion of this space. In this paper the space of coverage mirrors a recently defined class of *schema tuple queries*(Minock, 2002). Because of the natural closure properties of this language, and its decidability for equivalence and containment, it is reasonable to assume that many cooperative techniques may generate output 'queries' within this form.

The author of the query description system is assumed to be a database administrator. Thus we must adopt a generation technique that does not require a deep understanding of linguistics. Moreover the administrator must be given a structured method of authoring so that they may declare a schema *covered*. That is the system should faithfully, if not always elegantly, describe all queries of the form that are posed over the database schema. Given these requirements we adopt a phrasal approach that couples parameterized queries with patterns. The parameterized queries are within the class of the identified form and the patterns are simply modifier, head, complement triples.

1.1 Organization of this Paper

Section 2 shall review cooperative information systems and shall illustrate how such systems play the role of strategic natural language generators. Section 3 shall give a brief introduction to the language in which queries must be expressed. Section 4 describes the phrasal lexicon and section 5 describes the generation process. Section 6 discusses this work in the context of prior work and gives future directions.

2 Cooperative Information Systems

Cooperative information systems(Gaasterland et al., 1992) seek to extend conventional database query-answer dialogues with the principles of cooperative conversation(Grice, 1975). Thus the response to the user's query is richer than simply presenting the answers that meet the conditions of their query. Often such responses are conceptual and may in fact be specified with derived 'query' expressions. We now cover specific cooperative techniques that have such 'conceptual' outputs.

2.1 Cooperative Techniques

A user may have a *query misconceptions*, meaning that the user is unaware that their query presupposes an illegal state of the database. For example assume that a state law in Ohio is that all mayors must be over 25 years old. If a user is unaware of this restriction and issues a request "list all female mayors in Ohio that are younger than 23", they should be informed that "it is impossible for people under 25 to be mayors of cities in Ohio." Such a conceptual response is in the form of a 'query'. Gal (Gal and Minker, 1988) uses integrity constraints to explain query misconceptions back to the user.

Related to query misconceptions, a user should be made aware of a *false presupposition* they have about the database state. A false presupposition is an assumption that is implicit in a user's query, though false. The system CO-OP(Kaplan, 1982) used a limited theory of cooperation to correct false presuppositions. For example assume that a user requests: "Give the cities with population over 2 million in the state of Alaska or North Dakota that have a female mayor" A traditional system would say, "none." A system that could detect false presuppositions would respond, "there are no *cities in Alaska or North Dakota with a population over 2 million*." This is a description of the minimal failing sub-query of the users original query. Efficient algorithms exist to find minimal failing sub-queries (MFS) as well maximal succeeding sub-queries (MSS)(Godfrey, 1994).

Query relaxation is useful when a query has no matching tuples. During query relaxation conditions may be loosened or alternate entity types may be queried. For example when asking for flights from Dulles airport to La Guardia with a Sunday morning departure at 10 am, a relaxed query might return a 9:32 am flight from Dulles arriving at JFK. An entity type relaxation might offer a train or a bus trip rather than airplane flight. It is important to describe the relaxed query to the user before flooding them with extensional answers.

Intensional query answering (Imielinski, 1988) (Shum and Muntz, 1987) provides a summary answer rather than the entire tuple extension satisfying the query. If you are asking for all employees who make over 100,000$, instead of listing every single manager and Joe Star engineer, it is better to report "all the managers and the engineer named 'Joe Star'." This intensional response is more informative in the case that the user does not interpret an enumeration of all the managers names to mean 'all managers'. Once again the ability to describe queries is important.

The CARMIN system (Godfrey et al., 1994) includes an integrated explanation and answer presentation system. System explanations are based on the proof path used by a PROLOG meta-interpreter. Aspects of what to include and how to coordinate these explanations are also addressed(Gaasterland and Minkler, 1991). Natural language descriptions of the user query and relaxation process are generated for the cooperative information system CoBase(Chu et al., 1996),(Minock and Chu, 1996).

2.2 Cooperative Information Systems Serving as 'Strategic' Planners

We propose a modular approach in which the cooperative information system decides 'what' to say and a query description sub-system decides 'how' to say it.

Figure 1 shows an architecture for this approach. The cooperative information system consists of sub-systems that perform misconception detection, false presupposition detection, query relaxation and intensional answer generation. The user interface consists of sub-systems that perform query formulation, query description and answer presentation. Both the interface and the cooperative information system have access to the domain database.

A user is assumed to compose their query through some type of query formulator. This may be as primitive as a text box in which to type a logical query expression, to as advanced as a full natural language understanding system. Whatever the formulator type, it may be helpful to provide the user a natural language description of the query, q, that they have formulated.

After the user has verified their query, it is passed from the user interface to the misconception detection sub-system. If the query contains a misconception, then the offending portion of the query that caused the misconception, m, is reported as being 'impossible' and execution terminates. If the query contains no misconception, then subsequent flow depends on whether the user's query returns answers. If it does not return answers, the query is passed to the false presupposition detection sub-system. This sub-system identifies a minimal failing sub-query of the query. This minimal failing sub-query, s, is described to the user as not returning answers in the current database state. The minimal failing sub-query is then generalized to an answer returning query by the query relaxation sub-system. The fact that this relaxed query, a, generates answers is communicated to the user. Now that either the original query, or the derived relaxed query is answer generating, we then check whether a suitable in-

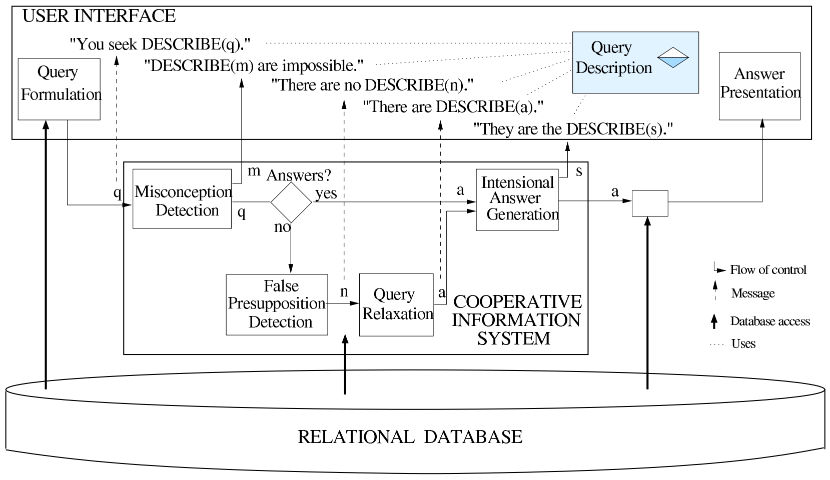


Figure 1: A cooperative information system as a strategic planner.

tensional summary may be returned as a substitute for fetching all the extensional answers to the query. The intensional answer generator either decides that no suitable summary exists, or it terminates the with a description of the summary 'query' s. If no suitable summary exists, then the extensional answers are retrieved from the database and are passed to the answer presentation sub-system.

Of course the proposed architecture leaves many issues unspecified. For example how does one pick a single minimal failing sub-query? How does one decide to relax the user's query? When is a summary appropriate in place of a full extensional answer. Still the important point to note here is that the cooperative information system makes such decisions based on semantic and pragmatic issues. These considerations determine the *quantity*,*quality* and *relation* of the content to be expressed. The tactical decision of the *manner* in which to express such content is left up to the sub-system that describes 'queries'.

3 A Class of Describable Queries

This paper now turns to the relatively pure problem of generating natural language descriptions for a broad class of relational database queries. Let us start by considering the following relational schema:

Person(name, gender, age, *city*)

City(cityName, population, *mayor*, state, country)

Knows(*knower*, *known*, opinion)

The semantics here are those of standard relational databases. The underlined attributes are the primary keys of the relations and the *italicized* attributes are foreign keys[1]. A wide variety of queries from simple to somewhat complex may be expressed over this schema. The goal here will be to describe such queries. The following queries are of interest:

1.) "Men living in 'Paris' or 'Nice'."

2.) "People living in cities with populations of over 100,000 people."

3.) "People not living in cities with populations of over 100,000 people."

4.) "People who know people living in 'Nice'

5.) "People who do not know people living in 'Nice'

6.) "People who know and like themselves"

7.) "People who know all people living in 'Nice'

The first five queries above may be described using our current approach. Query six has a reflexive reference that we are not yet able to handle. Query seven may not be expressed within the language we limit ourselves to - the language $\mathcal{L}$.

Definition 1 *(The language $\mathcal{L}$)*
$\ell \in \mathcal{L}$ if ℓ is in the form:

$$R(x) \bigwedge_{i=1}^{k} s_i \cdot (\exists \vec{y}_i)\Psi_i$$

where x is the only free variable of ℓ, $R(x)$ is the range condition for x, s_i is a positive or negative ($\neg$) sign, $\vec{y}_i$ is a finite sequence of existentially quantified variables and Ψ_i is a conjunction of range conditions, simple conditions, set conditions and join conditions.

[1] Those who are familiar with databases will note that this schema makes the rather simplistic assumption that all city names in the world are unique.

The example queries from above are shown here. See (Minock, 2002) for a more complete discussion of $\mathcal{L}$. Each query $\{x|\ell\}$ returns a set of tuples. For the first six queries, $\ell \in \mathcal{L}$.

1.) $\{x|Person(x) \wedge x.gender = \text{'male'} \wedge x.age \geq 18 \wedge$
$x.city \in \{\text{'Paris'}, \text{'Nice'}\}\}$
2.) $\{x|Person(x) \wedge (\exists y)$
$(City(y) \wedge y.population > 100000 \wedge$
$y.cityName = x.city)\}$
3.) $\{x|Person(x) \wedge \neg(\exists y)$
$(City(y) \wedge y.population > 100000 \wedge$
$y.cityName = x.city)\}$
4.) $\{x|Person(x) \wedge (\exists y)(\exists z)($
$(Know(y) \wedge Person(z) \wedge$
$\mathbf{z}.city = \text{'Nice'} \wedge$
$x.name = y.knower \wedge y.known = z.name)\}$
5.) $\{x|Person(x) \wedge \neg(\exists y)(\exists z)($
$(Know(y) \wedge Person(z) \wedge$
$\mathbf{z}.city = \text{'Nice'} \wedge$
$x.name = y.knower \wedge y.known = z.name)\}$
6.) $\{x|Person(x) \wedge (\exists y)$
$(Know(y) \wedge y.opinion = \text{'like'}$
$x.name = y.knower \wedge \mathbf{y.known = x.name})\}$

Query seven may not be expressed using $\mathcal{L}$

7.) $\{x|Person(x) \wedge (\forall z)(\exists y)$
$(Person(z) \wedge z.city = \text{'Nice'} \Rightarrow$
$Know(y) \wedge x.name = y.knower \wedge$
$y.known = z.name)\}$

Naturally all of these queries may be expressed using standard SQL.

Theorem 1 *($\mathcal{L}$ is decidable for $\subseteq, =$ and disjointness) if $q_1 \in \mathcal{L}$ and $q_2 \in \mathcal{L}$ then there exists a sound and complete inference mechanisms to decide if the three predicates:*
1.) $\{x_1|q_1\} \subseteq \{x_2|q_2\}$
2.) $\{x_1|q_1\} = \{x_2|q_2\}$
3.) $\{x_1|q_1\} \cap \{x_2|q_2\} = \emptyset$.
are necessarily true over the set of all database instances.

See (Minock, 2002) for the proof of this theorem. These properties will be used to maintain and select entries from the phrasal lexicon.

4 The Phrasal Lexicon

The approach here generates a single highly aggregated sentence of natural language that describes a query built over a formula in $\mathcal{L}$. The knowledge used to achieve this is a *phrasal lexicon*. The phrasal lexicon, denoted *PL*, consists of a set of *n entries* where each entry is a *parameterized query/pattern* pair. The *i*-th entry is $\langle\{x|\ell_i\} : p_i\rangle$ where $\{x|\ell_i\}$ is a parameterized query and p_i is a single pattern.

A parameterized query is simply a query defined using a formula in $\mathcal{L}$ in which constants may be parameters. Thus the query $\{x|Person(x) \wedge x.gender = c_1\}$ is a parameterized query where the constant c_1 could be 'male' or 'female'. The constant c_1 is said to be a parameter. We may also have set valued parameters as in: $\{x|Person(x) \wedge x.city \in C_1\}$.

A pattern is simply the three *phrases*: "[modifier] head [complement]". Phrases consist of plain text, possibly including parameters[2]. The modifier, head and complement distinction is best illustrated with example. The simple description,"Young people living in London" has "young" as a modifier, "people" as the head, and "living in London" as the complement. Thus it would be represented: "[Young] people [living in London]". The [phrase] syntax within a pattern signifies that groups of phrases may collect in such positions during aggregation. By contrast there can only be one head. Thus we may generate the aggregated pattern "[Young, employed] people [of the female gender, living in 'London']".

We now shall now cover the different types of entries within the phrasal lexicon. Special attention will be paid to insure that the entries completely cover the database schema over which queries may be posed.

4.1 Simple Entries

Let us start with the simplest type of entry. Here we specify the pattern associated with the condition free database relation *Person*:

$\langle\{x|Person(x)\}$:
"[ ] people [ ]"$\rangle$

Now we see the entry for the relation *Person*, the attribute *gender*, and the operator $=$.

$\langle\{x|Person(x) \wedge x.gender = c_1\}$:
"[ ] people [of the c_1 gender]"$\rangle$

The "[ ]" specifies that there is an an empty modifier for this pattern. The head is "people". Finally the complement phrase "[of c_1 gender]" has the parameter c_1.

There may be more than one entry with equivalent parameterized queries. For example:

$\langle\{x|Person(x) \wedge x.gender = c_1\}$:
"[c_1] people [ ]"$\rangle$

Naturally We may also have constants specified in the queries as well.

$\langle\{x|Person(x) \wedge x.gender =' male'\}$:
"[ ] **males** [ ]"$\rangle$

[2]As we shall see later, the complement phrase may also contain a recursive call to describe a sub-query.

In this final case we see that the head itself has been changed from the default for the relation. This may only occur once during a generation. The head is said to be *open* if it contains the same value as the head for the condition free entry over the relation. Otherwise it is said to be *frozen* and may not be combined with other patterns that alter the head.

Given the above entries we may generate the description of the query:

$\{x|Person(x) \wedge x.gender = \text{'male'}\}$

As "people of the male gender", "male people", or "males". The simple heuristics we employ prefer the last form over the first two.

We must fully populate the lexicon to cover all of the attribute/basic operator combinations for each relation in the schema. If there are 6 basic operators $(>, \geq, =, \neq, <, \leq)$ and two set operators $(\in, \notin)$, and each attribute/operator combination makes sense, then the minimal number of simple entries required by the schema in section 3 is: $(8 \times 4 + 1) + (8 \times 5 + 1) + (8 \times 3 + 1)$. The '$+1$' terms signify the entry for the case where no conditions are applied.

4.1.1 Aggregating Simple Entries

Now we show how simple entries combine to describe queries with more than one condition. In addition to the previous entries, assume that we also have the following entry:

$\langle\{x|Person(x) \wedge x.city \in C_1\}$:
"[ ] people [living in C_1]"$\rangle$

Now suppose that the query for the "males living in Paris or Nice" needs to be described. The query is:

$\{x|Person(x) \wedge x.gender = \text{'male'} \wedge$
$x.city \in \{\text{'Paris', 'Nice'}\}\}$

This may be rewritten as a combination of filled in parameterized queries from the phrasal lexicon.

$\{x|Person(x) \wedge x.gender = \text{'male'}\} \cap$
$\{x|Person(x) \wedge x.city \in \{\text{'Paris', 'Nice'}\}\}$

Excluding the permutations of the 'Paris' and 'Nice', the possible ways to *combine* the 3 patterns that match the first query and the 1 pattern that matches the second are:

1.) [ ] males [living in 'Paris' or 'Nice']

2.) [male] people [living in 'Paris' or 'Nice']

3.) [ ] people [of the male gender, living in 'Paris' or 'Nice']

4.) [ ] people [living in 'Paris' or 'Nice', of the male gender]

Using a simple heuristic of minimizing sentence length, the first choice is preferred. Extending the heuristic to communicate the maximum amount of information in *k* symbols, we induce the ordering presented here.

4.2 Join Entries

Now we face the issue of representing the patterns associated with join conditions within queries. First we shall consider the simple case of joining over foreign keys. This accounts for the bulk of meaningful many-to-one and one-to-many relationships. Then we shall consider the more complicated case involving many-to-many relationships[3].

4.2.1 One-to-many and many-to-one Relationships

Each attribute that can be meaningfully joined with another must be considered. For the example of section 3 this would amount to a total of 6 attribute matches. Assuming that we are only interested in equality joins, then we must account for $2 \times 2 \times 6$ entries to cover this space. The reason for the first doubling is that one must take into account direction when one describes joins. The second doubling occurs because one must also consider the negative case in which the query specifies that answers do <u>not</u> participate in such relationships.

For the join involving $person : city \longrightarrow^{+} city : cityName$ we have the entry:

$\langle\{x|Person(x) \wedge (\exists y)(City(y) \wedge$
$x.city = y.cityName \wedge \Phi\}$:
"[ ] people [that live in $GEN(\{y|City(y) \wedge \Phi\})$"]$\rangle$

The negative case is almost identical. The entry for $person : city \longrightarrow^{-} city : cityName$ is:

$\langle\{x|Person(x) \wedge \neg(\exists y)(City(y) \wedge$
$x.city = y.cityName \wedge \Phi\}$:
"[ ] people [that don't live in
$GEN(\{y|City(y) \wedge \Phi\})$"]$\rangle$

The key issue to note here is that the complement phrase has a recursive call. This will cause a completely new generation problem to be instantiated.

Assuming that we have simple entries covering *city*, we may now generate descriptions such as "[adult] [males] [that live in ([ ] cities [with populations over 100,000 people])". The material enclosed within

[3]For those familiar with Entity-Relationship modeling, the goal is to generate descriptions of an entity's participation in one-to-many and many-to-many relationships. We skip one-to-one relationships because of their simplicity. Note that we do not yet handle either reflexive (self-joining) relationships or general *n*-ary relationships, even though $\mathcal{L}$ admits such queries

parenthesis shows the solution to the recursive description problem.

4.2.2 Many-to-many relationships through joins

We now consider a case of many-to-many type relationships. These present a special, though not insurmountable difficulty. There is only one possible many-to-many relationship in the example of section 3. This is the relationship that many people may know many other people. Especially vexing is the fact that attributes may also be involved within the many-to-many relationship. For example the opinion one has about who one knows. Consider the following entry:

$\langle\{x|Person(x)\wedge$
$(\exists y)(\exists z)(Knows(y)\wedge Person(z)\wedge$
$opinion = c_1 \wedge x.personId = y.knower\wedge$
$y.known = z.personId \wedge \Phi\}$:
"[ ] people [who know and c_1 some
$GEN(\{z|Person(Z)\wedge\Phi\})$]"$\rangle$

The only way to precisely control this is to make multiple join entries for each combination of given attributes in the joining relation. Thankfully relations that bridge many-to-many relations are often of fewer attributes. In the case of the example in this paper the many-to-many relationship is taken care of with 4 entries.

4.3 Coverage and extension of the phrasal lexicon

If we may guarantee that there are a sufficient set of entries to fully cover the schema, we may declare the phrasal lexicon to be *covered* with respect to the schema. Thus based on the approach above, it takes 123 entries to cover the schema in section 3.

Naturally we may improve the phrasal lexicon by extending it to cover more specific entries. For example, the following entry will simplify some descriptions considerably:

$\langle\{x|Person(x)\wedge x.gender =$ 'male' $\wedge x.age \geq 18\}$:
"[ ] **men** [ ]"$\rangle$

5 The Generation Process

Beyond simple aggregation, we have not yet described how the entries within the phrasal lexicon are used to obtain natural language descriptions of a query. We begin with a definition of what constitutes a description and then we show a relatively efficient mechanism to actually obtain such descriptions. Finally we say a little about how specific descriptions are selected.

5.1 Descriptions Defined

A *description* of a query is the aggregate pattern generated through applying a *covering*, *non-redundant* subset of the phrasal lexicon over the query.

Before we define what a covering, non-redundant subset of the the phrasal lexicon is, we must resolve formal difficulties associated with parameters in the phrasal lexicon. Because there are a finite number of constants in the formula specifying the query (ℓ) and a finite number of parameters within the entries of the phrasal lexicon (*PL*), we may consider the expansion of the phrasal lexicon to be all combinations of constants from within ℓ substituted in place of parameters within the entries of *PL*. This generates the finite expanded lexicon PL^{ℓ} where all parameters are bound.

We now define the subsets of the expanded lexicon that *cover* the query without *redundancy*. A set of lexical entries $s \in 2^{PL^{\ell}}$ are said to *subsume* $\{x|\ell\}$ iff the set intersection of all the queries within the entries of s necessarily contain $\{x|\ell\}$ where $\top$ is substituted for Φ in every (parameterized) query. Another, more concise way of saying this, is that the extension of the entries s contains $\{x|\ell\}$. A set of lexical entries $s \in 2^{PL^{\ell}}$ is said to *cover* $\{x|\ell\}$ iff there is no other s' where $s' \supset s$ where s' subsumes $\{x|\ell\}$, and the extension of s' are properly contained within the extension of s. Finally a set $s \in 2^{PL^{\ell}}$ is *redundant* iff some $s' \subset s$ has the same extension as s.

5.2 Obtaining Descriptions

To quickly identify subsets of the phrasal lexicon that cover the user query without redundancy, we organize the phrasal lexicon into a *subsumption hierarchy*[4]. The phrasal lexicon is compiled into a subsumption hierarchy by sorting entries down into the hierarchy. Nodes correspond to parameterized queries, though entries with equivalent parameterized queries are collapsed into a single node with multiple attached patterns. When we sort the query into the subsumption hierarchy, the set of immediate parents identify the set of entries that cover the user query without redundancy.

Figure 2 illustrates the subsumption hierarchy for the entries we have considered thus far. The upper portion of the hierarchy consists of the nodes that correspond to the 123 entries we must provide if we wish to cover the given database schema. The lower portion of the hierarchy consists of nodes representing entries that are meant to make generation more precise. Figure 2 also shows two queries sorted into the hierarchy.

The only complication in the sort procedure is handling parameters and open formula of the lexical entries. The way to handle parameters, is to allow them to range over any domain value except a distinguished

[4]The notion of subsumption hierarchy here corresponds with that used in description logics. However it should be noted that the logic here differs considerably from the unary and binary predicates of typical description logics. See (Minock, 2002) for a full discussion of this.

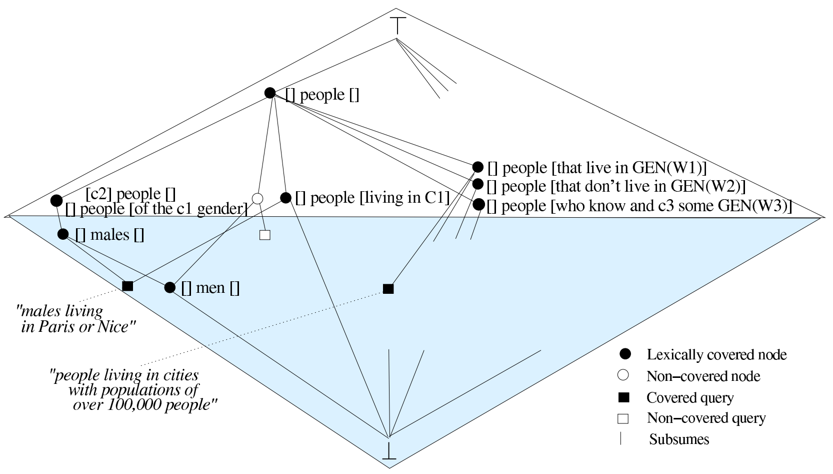

⊤
[] people []
[c2] people []
[] people [of the c1 gender]
[] people [living in C1]
[] people [that live in GEN(W1)]
[] people [that don't live in GEN(W2)]
[] people [who know and c3 some GEN(W3)]
[] males []
[] men []
"males living in Paris or Nice"
"people living in cities with populations of over 100,000 people"
⊥
● Lexically covered node
○ Non–covered node
■ Covered query
□ Non–covered query
| Subsumes

Figure 2: The phrasal lexicon organized as a subsumption hierarchy.

symbol *unkown*. This is recored by inserting the simple condition that the parameter does not equal *unknown*. Open formula Φ are simply assumed to be true and are replaced with the symbol ⊤.

Given that the parameterized queries and queries requiring description tend to be of limited size, we assume that deciding subsumption between two parameterized queries takes constant time. If the number of entries in the phrasal lexicon is n, then $O(n)$ nodes are in the subsumption hierarchy. It thus costs $O(n)$ subsumption operations to sort a query into the hierarchy.

5.3 Selecting Descriptions

Given a set of nodes that cover a query without redundancy, we may select combinations of associated patterns from this set of nodes to generate an actual description. The conditions guiding this process are that only one pattern is selected from each node and at most one pattern of the combination may over-ride the default head. A heuristic that we adopt is to prefer the shortest sentence. It is likely that a greedy technique is sufficient to enable the identification of such a minimal combination. Occasionally the whole generation process recurs when there is a Φ term within a complement. Once all the textual material is obtained from the recursive calls, a final process reorders the constituents so that the shorter phrases appear before longer phrases.

6 Discussion

It has been observed that fielded NLG systems tend to have pipelined architectures with the vast majority using some type of semantic network based representations as the common knowledge representation language(Reiter, 1994). Surface realization tends to be carried out using unification grammars(Kay, 1979)(Penman Project, 1989)(Elhadad, 1993). A common use for such grammars is enforcing number, gender, and case agreement.

The system here is also pipelined. The cooperative information system is granted the strategic decision about 'what' to express and the query description generator decides 'how' to express description requests from the cooperative information system. Thus the language being shared here is the language of query expressions, which are analogous to a fragment of first order logic. The description system interacts with the semantical system to obtain the most succinct descriptions of a 'query', but it does not pass any information back to the cooperative information system. This is in contrast to more general techniques that explicitly plan content through complex plan operators(Moore and Paris, 1989) or schemata(McKeown, 1985) using established rhetorical theories(Mann and Thompson, 1988). The specific, cooperative information strategy adopted here is less flexible, but the knowledge specification task is simplified considerably.

The choice of using a non-feature based phrasal grammar is based on the relative ease by which non-expert administrators might provide such phrasal knowledge. It is also anticipated that case and number errors will be of only minor annoyance and that clever administrators might be able to author phrases so that such errors are minimized. Currently it is assumed that a single, highly aggregated sentence, may describe a query. Certainly there is some limit to the number of query conditions that may be aggregated into a single sentence. Techniques to break up of the sentences must be entertained if we are to scale to more complex queries. Issues such a pronominalization and ellipsis are not yet addressed in this work, but will become more important as we consider how to span complex query descriptions over multiple sentences.

Though the idea of using a phrasal grammar is not new(Reiter, 1990), nor is using classification in text generation(Reiter and Mellish, 1992), the approach here is new in regard to exploiting the properties of the query formation language $\mathcal{L}$. As long as the reasoning task is able to present its results as expressions within this language (or, more liberally disjunctions of $\mathcal{L}$ expressions) then there may indeed be a high degree of modularity between the reasoning system and the generation component(Shieber, 1994).

7 Conclusions

This paper proposes a scalable and structured approach to generating natural language descriptions for a broad class of relational database queries. The simplicity of the phrasal approach enables clever database administrators to author the system without requiring specialized linguistic knowledge. The firm semantic basis of the approach lends a great deal of structure to the authoring process. Notably an administrator can declare their schema 'covered' once they have provided lexical entries for a bounded set of simple and join conditions.

8 Bibliography

References

W. Chu, H. Yang, Chiang K., M. Minock, G. Chow, and C. Larson. 1996. Cobase: A scalable and extensible cooperative information system. *Intelligent Information Systems*, 6(3):223–259.

M. Elhadad. 1993. FUF: the Universal Unifier. User Manual Version 5.2. Technical report, Computer Science, Ben Gurion University.

T. Gaasterland and J. Minkler. 1991. User needs and language generation issues in a cooperative answering system. In *ICLP Workshop on Adv. Logic Programming Tools and Formalisms for Language Proc.*

T. Gaasterland, P. Godfrey, and J. Minker. 1992. An overview of cooperative answering. *Intelligent Information Systems*, 1(2):127–157.

A. Gal and J. Minker. 1988. Informative and cooperative answers in databases using integrity constraints. In *Natural Language Understanding and Logic Programming*, pages 277–300.

P. Godfrey, J. Minker, and L. Novik. 1994. An architecture for a cooperative database system. In *Proceedings of the 1994 International Conference on Applications of Databases*.

P. Godfrey. 1994. Minimization in cooperative response to failing database queries. Technical report, University of Maryland Dept. of Computer Science, College Park, MD.

P. Grice. 1975. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics*. Academic Press.

T. Imielinski. 1988. Intelligent query answering in rule based systems. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufman Publishers.

S. Kaplan. 1982. Cooperative responces from a portable natural language query system. *Artificial Intelligence*, 19:165–187.

M. Kay. 1979. Functional grammar. Fifth Annual Meeting of the Berkeley Linguistics Society.

W. Mann and S. Thompson. 1988. Rhetorical structure theory. *TEXT*, 8(3):243–281.

K. McKeown. 1985. *Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press.

M. Minock and W. Chu. 1996. Interactive explanation for cooperative information systems. In *Proceedings of the 9th International Symposium on Methodologies for Intelligent Information Systems*.

M. Minock. 2002. Beyond query containment: a decidable language supporting syntactic query difference. Technical Report 02.21, The Univeristy of Umea, Umea, Sweden, December.

J. Moore and C. Paris. 1989. Planning text for advisory dialogue. In *Proceedings of the 27th Meeting of the Association for Computational Linguistics*.

Penman Project. 1989. Penman documentation: the Primer, the User Guide, the Reference Manual, and the Nigel manual. Technical report, USC/Information Sciences Institute.

E. Reiter and C. Mellish. 1992. Using classification to generate text. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics.*, pages 265–272.

E. Reiter. 1990. Generating appropriate natural language object descriptions. phd thesis. *Harvard*.

E. Reiter. 1994. Has a consensus NL generation architecture appeared, and is it psychologically plausible? In *Proceedings of the 7th. International Workshop on Natural Language generation*, pages 163–170.

S. Shieber. 1994. The problem of logical-form equivalence. *Computational Linguistics*, 19(1):179–190.

C. Shum and R. Muntz. 1987. Implicit representation for extensional answers. In L. Hershberg, editor, *Expert Database Systems*. Tysons Corner.

Porting to an Italian Surface Realizer: A Case Study

Alessandra Novello and **Charles B. Callaway**
ITC-irst, TCC Division
via Sommarive, 18
Povo (Trento) I-38050, Italy
{callaway, novello}@itc.it

Abstract

Multilingual generation is becoming an increasingly important aspect of implemented systems that showcase the abilities of generation systems. Most such systems require multiple grammars, one for each language which must be deployed. Yet little is known about the development costs for additional languages which are developed not from scratch, but by adapting existing resources. We ported a standard English surface realizer and grammar with wide coverage to Italian. After describing major grammatical differences, we quantitatively specify the porting process and present statistical information for the changes we found necessary to develop the new grammar.

1 Introduction

Multilingual generation systems will play increasingly important roles in showcasing the abilities of deep NLG (Paris et al., 1995; Stede, 1996; Callaway et al., 1999; Scott, 1999). These systems require an array of resources that can function regardless of the language selected, such as discourse and sentence planning rules, lexica, and pronominalization strategies. One of the most important of these resources is the grammar that a surface realizer uses to produce linearized text from a syntactic sentence plan, and multilingual systems must use a distinct grammar for each desired language.

While many multilingual systems have either developed grammars from scratch or borrowed them from other projects, relatively few projects have focused on reworking existing grammars to port them to new languages. Most such work has been connected with the KPML environment (Bateman, 1997; Aguado et al., 1998; Kruijff et al., 2000), and the newer EXPRIMO system developed at Edinburgh and based on ILEX (Oberlander et al., 1998). However, these projects have not addressed the issue of exactly how much effort is involved in converting a surface realizer for one language into another in a quantitative manner. And while (Callaway et al., 1999) presented basic data on an English to Spanish project, it was not comprehensive enough to allow future projects to accurately estimate what potential development costs might be.

A separate trend has been to justify as both useful and cost-effective the continued use of resources in investigating deep natural language generation over other, more near-term approaches such as template generation. In order to make an informed comparison, hard data is needed on the costs for developing and maintaining projects which use both formalisms. In this article, we provide such data for the grammar and morphology development of an Italian surface realizer as a first step in allowing such comparisons to be made.

During the course of work on a multilingual generation system for English and Italian, we took elements from both the original FUF/SURGE systemic-functional surface realizer for English (Elhadad, 1991; Elhadad, 1992; Robin, 1994) as well as a less-developed Spanish version (Callaway et al., 1999) of that same realizer to create a new Italian version[1]. The porting process involved changes to morphology, linearization, and the grammar, while leaving unchanged other features of the FUF system such as formatting and efficiency directives. This paper presents the results of creating the new surface realizer, including an overview of differences between the languages and a quantitative analysis of the effort and changes involved.

2 Examples of Language Differences

The differences between Italian and English are not significant compared to languages from differing families. The following areas are indicative of the types of linguistic changes necessary when generating Italian text as opposed to English. Extensive catalogues of such changes for other languages such as French also exist (Rayner et al., 1996). The various categories for Italian include:

Morphology Changes that affect the prefixes and suffixes of words for purposes of agreement, along with interactions between surface forms after they have already been syntactically specified.

- *Irregular Words*: Irregulars mainly concern lexical forms for nouns, verbs, and adjectives (which have few irregular forms in either English or Italian). Besides the three most important, regular rules for Italian noun pluralization (-o/-i, -a/-e, -e/-i), there are more than 20 other minor rules for pluralization (*e.g.*, nouns with accented endings: sing. *crisi*, pl. *crisi*) and a third category of completely irregular plurals (*e.g.*, sing. *tempio*, pl. *templi*). Furthermore, while English can express all verbs with at most five basic forms plus auxiliaries, Italian verbs can have up to 49 different irregular forms.

- *Contractions*: Italian can form contractions between a preposition and a definite article, such as *su* + *la* ⇒ *sulla* ("on" + "the"). Additionally, contractions can occur between certain proclitic pronouns and verbs beginning with a vowel or 'h' plus a vowel (*e.g.*, *l'ho vista* "I have seen her", or *c'é* "there is"). There are also rules for dropping unstressed vowels, especially after infinitives: *aver detto* rather than *avere detto*, or with enclitic pronouns: *fare+lo* = *farlo* "to do it".

Word Order Differences in relative positioning of certain syntactic categories with respect to others and co-occurrence constraints.

- *Adjectives*: Adjectives in Italian can be found in pre-nominal or post-nominal position. Some adjectives allow only one position, so a feature "pre-n" or "post-n" must be added to the lexicon. Lots of adjectives can appear in both positions, causing the distinction between their appositive and restrictive use. Since some adjectives change their meaning completely according to their position,

La vecchia strada	*La strada vecchia*
(*lit.* the old street)	(*lit.* the street old)
"The familiar street"	"The old street"

  this requires that they be listed as different lexical items. Further order constraints arise when more than one adjective determines a nominal head:

  Un nuovo cinema italiano
  (*lit.* a new cinema Italian)
  Un cinema italiano nuovo
  (*lit.* a cinema Italian new)
  **un cinema nuovo italiano*
  (*lit.* a cinema new Italian)
  "A new Italian cinema"

- *Subject Position*: Subject in Italian can occur either in preverbal or postverbal position. It generally precedes the verb, but it follows it with unaccusative and unergative structures:

  E' arrivata Laura.
  (*lit.* Is arrived(agr.) Laura.)
  "Laura arrived."

  Other verbs such as "mancare" (be lacking), "piacere" (be pleasing), and "servire" (be of

[1]The resulting grammar is freely available for research purposes at http://tcc.itc.it/

use) strongly prefer the inversion of complement order:

A Luca piace la pasta.
(*lit.* (dat-prep.) Luca likes the pasta.)
"Luca likes pasta."

The postposition of the subject is also required with interrogative WH:

Che cosa ha comprato Giorgio?
(*lit.* What thing has bought Giorgio?)
"What did Giorgio buy?"

- *Clitics*: Accusative clitics precede the finite verb, while direct objects usually follow it:

  Mary l'ha letto.
  (*lit.* Mary cl.(acc) has read.)
  "Mary has read it."

  With restructuring verbs, clitics can attach either to the main verb or to the infinitive:

  Mary vuole comprarlo.
  (*lit.* Mary wants to buy cl.(acc).)
  Mary lo vuole comprare.
  (*lit.* Mary cl.(acc) wants to buy.)
  "Mary wants to buy it."

  But clitics follow the verb when the mood is imperative:

  Lo regali a Gianni. (indicative)
  (*lit.* cl.(acc) give to Gianni.)
  "Give it to Gianni."

  Regalalo a Gianni! (imperative)
  (*lit.* give cl.(acc) to Gianni!)
  "Give it to Gianni!"

  When both dative and accusative clitic are required, the order of complements is inverted (dative precedes accusative):

  Mary me lo dice.
  (*lit.* Mary cl.(dat) cl.(acc).)
  "Mary tell it to me."

Grammar Modifications to choosing which syntactic categories are allowed in which positions and what defaults are given to individual features.

- *Secondary Clauses*: Sentences where matrix verbs govern a gerund clause, such as

  Ho visto il ragazzo uscendo dalla chiesa.
  (*lit.* have seen the boy leaving from the church.)
  "I saw the boy leaving the church."

  by default prefer to keep the subjects identical, whereas in English the object of the matrix verb generally corefers to the subject of the matrix verb by default. Thus where the boy was leaving the church in the English example above, in the Italian version it is the speaker who was leaving the church.

- *Formal/polite pronouns*: Italian uses the third person feminine address "Lei" (even when it is addressed to a male person) instead of the second person. The use of the polite form involves changes to verbs and pronouns when the mood is imperative. Indeed, Italian has imperative forms for the second singular person and second and third plural, but changes to the subjunctive for polite imperatives, eg.:

  Leggi! (imperative)
  "Read!"
  Legga! (imperative realized by a subjunctive)
  "Read!"

  Further changes arise from the use of clitics:

  Leggilo! (enclitic in familiar form)
  "Read it!"
  Lo legga! (proclitic in polite form)
  "Read it!"

- *Verb-governed pronouns*: Most notably, dative constructions in Italian are much different than those in English. Features in SURGE like "dative-shift" are not useful and are thus not referenced in the Italian Grammar.

Discourse Differences in which although a constituent is allowed grammatically, one language prefers something slightly different.

- *Zero pronominalization*: Also called *pro-drop* (Haegeman, 1994), this is the result of not mentioning a repetitive subject pronoun, as it is redundant given that verbs are inflected for a subject's number and gender (Di Eugenio, 1998).

3 Coverage of the Italian Grammar

Most symbolic generation systems use regression testing as a means of demonstrating the amount

```
"This car is expensive."
((cat clause)
 (proc ((type ascriptive)
        (mode attributive)))
 (partic ((carrier ((lex "car")
                    (cat common)

                    (distance near)))

          (attribute ((lex "expensive")
                      (cat ap)))))))
```

```
"Questa macchina e' costosa."
((cat clause)
 (proc ((type ascriptive)
        (mode attributive)))
 (partic ((carrier ((lex "macchina")
                    (cat common)
                    (gender feminine)
                    (distance near))))

          (attribute ((lex "costoso")
                      (cat ap)))))))
```

Figure 1: A simple example with almost direct feature-feature mapping

of coverage of a particular language. For example, the FUF/SURGE surface realizer includes over 500 examples of paired inputs and outputs covering a wide range of phenomena subdivided into categories such as yes/no questions, relative clauses, noun phrases, etc.

Although we did not attempt to duplicate coverage for this extensive test suite, we did obtain enough coverage to allow for the production of multiple paragraphs of simple text.

Throughout our efforts, we strove to make the input representation as similar as possible to the existing SURGE test suite. An example of this similarity is found in Figure 1, where only individual lexical items differ between the two functional descriptions. Thus the new surface realizer can be used with existing discourse and sentence planners, with only changes to the lexicon needed in a pipelined NLG architecture.

Figure 2 shows a more complex example where the structures of the sentences are so divergent that either the sentence planner must be able to generate different sentential representations or the interface to the surface realizer must be moved even higher to exclude all syntactic references. For Italian and English we have seen a higher proportion of the simple cases in our application environment, although it is highly likely that the exact proportion changes by language pair.

4 The Porting Process

Porting the SURGE grammar to Italian was accomplished in a systematic way. We first worked on the morphology of individual words in isolation. Italian words typically have much more inflection than those in English as is documented in most books on language instruction. Italian morphology is well defined, and thus we used such materials to ensure that morphologic coverage was complete and could be performed rapidly. Another quick change that could be made was to replace all lexicalized closed-class words in the grammar (such as the English "to" with infinitives).

Next, we performed basic testing of examples in the provided SURGE test suite (with Italian lexicalizations substituted and additional features like gender added) to gauge how many changes might be necessary. The results showed that morphology interactions and linear precedence were the most obvious errors that were immediately noticeable. We thus proceeded to attempt to fix these errors before concentrating on the grammar itself. After reexamining the newly regenerated sentences, we found that most were recognizably similar to the Italian equivalent "gold standards", even if they contained many errors.

We next controlled for morphological interactions between adjacent words (similar to contractions in English), adjust for accented characters not present in English, and add simple feature propagations such as gender in predicate verb and attributive noun constructions where similar such features (*e.g.* number) already existed. Finally, we began work on the more difficult differences in actual grammar, which required significantly more time than the changes mentioned above.

At this point there are two possible directions that a surface realization project can take: to continually develop the grammar as a linguistic-

```
 "The town is meant to be Trento"
((cat clause)
 (proc ((type lexical)
        (lex "mean")
        (voice passive)
        (subcat ((1 {^3 lex-roles influencer})
                 (2 {^3 lex-roles influenced})
                 (3 {^3 lex-roles soa})
                 (1 ((cat np)))
                 (2 ((cat np)
                 (3 ((cat clause)
                     (mood to-infinitive)
                     (controlled {^ oblique 1}))))))))))
 (lex-roles ((influenced ((cat common) (lex "town"))
             (soa ((proc ((type equative)))
                   (partic ((identified ((cat proper)
                                         (lex "Trento"))))))))))


 "Si ritiene che la citta' sia Trento"
((cat clause)
 (proc ((type lexical)
        (lex "ritenere")
        (subcat ((1 {^3 lex-roles believer})
                 (2 {^3 lex-roles belief})
                 (1 ((cat np)))
                 (2 ((cat clause)
                     (binder ((lex "che")))
                     (mood bound))))))
 (lex-roles ((believer ((cat personal-pronoun) (case reflexive)
                        (animate yes) (person third)))
             (belief ((cat clause)
                      (proc ((type ascriptive) (mode equative)
                             (mood subjunctive)))
                      (partic ((identified ((lex "citta") (cat common)
                                            (gender feminine)))
                               (identifier ((lex "Trento")
                                            (cat proper))))))))))
```

Figure 2: A more complex example where features are not aligned

only initiative to provide extensive coverage (the breadth approach), or to begin to flesh out particular projects and provide only the type of linguistic structures they need but in greater detail (the depth approach).

5 Quantitative Results

The overall process required approximately five person-months, split between two people: one with pre-existing knowledge of FUF/SURGE and a native English speaker, the other without knowledge of FUF/SURGE who is a native speaker of Italian. By the middle of the project, the second person was capable of making major grammatical changes unaided. Also, as the project continued and intensive knowledge of Italian was increasingly necessary, the burden of the labor shifted to the native Italian speaker. Below we detail the information gathered after this five month period, when the surface realizer was sufficiently developed to produce a paragraph of Italian in a working demo where the equivalent English paragraph was also generated from the same discourse plan.

Table 1 shows various quantitative aspects of the grammar-creation process. *Lines* refers to the number of lines of actual code devoted to different items. While not indicative of the amount or degree of changes necessary, the results show a high degree of correlation between the overall sizes.

	English Grammar		Italian Grammar		
	Lines	Constraints	Lines	Constraints	Work Time
Top Level	477	392	496	335	1%
Adverbials	1191	3167	1034	2861	2%
Clauses	461	202	432	200	8%
Conjunctions	293	126	283	100	3%
Determiners	900	688	933	636	17%
Clause Mood	511	282	485	245	2%
Noun Phrases	753	479	732	463	17%
Transitivity	749	420	743	423	5%
Verb Group	927	723	699	453	6%
Clause Voice	432	235	519	278	8%
Grammar Totals	6694	6714	6356	5994	69%
	Lines		Lines	Changed	Work Time
Irregular Verbs	210		450	100%	10%
Other Irregulars	40		80	100%	7%
Other Morphology	680		700	30%	5%
Linearization	640		640	5%	4%
	Lines	Examples	Lines	Examples	Work Time
Regression Testing	5922	∼500	680	45	5%
Totals	14186		8906		100%

Table 1: Grammar, Code, and Resources Data

This indicates that even when substantial changes were made, they effect was to replace rather than increase or decrease the size of the grammar.

A more closely related statistic is the number of actual *constraints* incorporated in the grammar. Due to the feature-based nature of the functional unifi cationformalism (Kay, 1979) underlying SURGE, it is possible to count the number of times features are expressed. This ignores the effects that comments and formatting imposed by different grammar authors have on the grammar itself as a data structure.

Work time refl ects the percentage of the five person months that were spent in certain areas of the grammar and other resources. Small percentages indicate that a grammar module was little changed from the previous grammar. Unlike the other statistics, this is an estimate, as we did not count the actual time spent in each area. Importantly, it is probably not possible to make a completely accurate estimate of time spent, as different people work at different speeds, and even a single person will work slowly or quickly on different days.

This data indicates that noun/determiner phrases and morphology required the most amount of work. We hypothesize that this data also indicates the verbal systems between English and Italian are closer than the nominal systems.[2]

To ensure the robustness and to double check the coverage, we employed the grammar as part of several multilingual projects we are currently researching. Thus the size of the regression test set is rather small compared to that of the English version, especially as we did not fi ndexamples of questions, appositions, partitives, dates, forms of address, *etc.* that are contained in the original regression set. From this work we estimate that another five to six person months would be necessary to ensure that these additional types of possi-

[2]The large differences in the *verb-group* module are not indicative of signifi cantchange; we removed many of the extensive tenses SURGE has for sentences like " Hewas about to be going to the bus." With these included, the fi lelength and number of constraints are still tightly correlated.

ble grammar inputs could also be generated.

6 Discussion

There are at least three important questions that need to be resolved in future research of this type:

- What impact is there on portability
- What type of regression testing is needed
- How does porting for deep generation compare with that for template generation

It is highly desirable that multilingual generation can take place with a minimal amount of changes to architectural modules. Because the functional unifi cationformalism is feature-based, features are a necessary aspect of the representation. But it is an open research topic whether different languages can be described with a similar set of semantic features (*c.f.* research on interlinguas) and if so, can such a representation generate a large enough set of paraphrases in each target language. Another way of looking at this is to ask whether the divergent structures in Figure 2 can be resolved so that they are identical but generate the different required syntax. If not, this problem must be pushed further up the NLG pipeline. In the texts we have generated, we have yet to fi nd an example where changes above the level of the sentence planner must be made. In practice, most syntactic features which are not part of both languages (such as the dative-shift feature in SURGE) have been safely ignored because they are not referenced or constrained in the Italian grammar.

A second aspect is the use of regression testing with a small number of examples to " test"the coverage of the grammar. Without standards, regression testing is useless as a comparison metric between surface realizers because (1) it is not clear how many examples are necessary, (2) there is no recognized set of levels of coverage other than " notcomplete", and (3) it is not clear how complex a set of examples needs to be (*e.g.*, does every possible combination of intermixed features or syntactic constructions need to be attempted).

Finally, it is important to be able to compare with a cost/benefi tanalysis the claimed advantages of lingustic-based (deep) generation over those of string-based (template) generation. Related to this particular project are 4 aspects of this problem:

- What other NLG infrastructure was there to begin with? The Italian grammar we developed was used in a separate project which already had extensive NLG infrastructure. For both template and deep generation systems, most of the development effort goes into producing a working system for a single language. But it is unknown what the costs of additional languages are for each approach.
- How domain-independent is each approach? Intuitively, template approaches are highly domain-specifi c while deep generation is more domain independent. But to what extent has never been quantitatively demonstrated.
- How much effort is required to integrate with other NLG elements? For example, future work may fi ndthat pronominalization or revision algorithms have different effects depending on language. If hidden interaction complexities arise, they may have a large infl uenceon cost/benefi tratios.
- What is the break-even point where development and maintenance costs for template approaches outweigh those for deep generation? In this project, would another 5 months of effort on the grammar result in a domain-independent surface realizer? We believe so, but detailed evidence must be collected on a template realizer and deep generation realizer working with an identical NLG pipelined system on an identical project and domain to be certain.

7 Conclusion

Multilingual generation is an increasingly important tool to demonstrate the widely-believed but little-substantiated intuition that natural language generation can provide effective and effi cientsystems whose development costs outweigh those of other methodologies such as template generation. But there has been little published evidence on what these costs may be, without which it is impossible to make an educated comparison.

We have thus presented a quantitative analysis of the effort required to build a grammar by reusing existing resources, a summary of the changes required, and estimates of how much work was devoted to varying aspects. This type of data is a necessary precursor to making future comparisons between differing methodologies on the basis of system development cost rather than traditional approaches which evaluate the text produced in a working system.

Finally, the material result of this project has been a functioning Italian generation grammar, which we plan to make available to the NLG community as an open source, freely available common resource.

8 Acknowledgements

This work was funded by the PEACH and TICCA projects, funded by the Autonomous Province of Trento. We would like to thank the anonymous reviewers, who gave us particularly lengthy and insightful comments.

References

G. Aguado, A. Bañón, J. Bateman, S. Bernardos, M. Fernández, A. Gómez-Pérez, E. Nieto, A. Olalla, R. Plaza, and A. Sánchez. 1998. ONTOGENERATION: Reusing domain and linguistic ontologies for spanish text generation. In *ECAI Workshop on Problem-Solving Methods and Ontologies*, Brighton, UK.

John A. Bateman. 1997. Enabling technology for multilingual natural language generation: The KPML development environment. *Journal of Natural Language Engineering*, 3(1):15–55.

C. Callaway, B. Daniel, and J. Lester. 1999. Multilingual natural language generation for 3D learning environments. In *Proceedings of the 1999 Argentine Symposium on Artificial Intelligence*, pages 177–190, Buenos Aires, Argentina.

Barbara Di Eugenio. 1998. Centering in Italian. In Marilyn A. Walker, Aravind K. Joshi, and Ellen F. Prince, editors, *Centering in Discourse*. Oxford University Press, Cambridge, MA.

Michael Elhadad. 1991. FUF: The universal unifi er user manual version 5.0. Technical Report CUCS-038-91, Department of Computer Science, Columbia University.

Michael Elhadad. 1992. *Using Argumentation to Control Lexical Choice: A Functional Unification Implementation*. Ph.D. thesis, Columbia University.

L. Haegeman. 1994. *Introduction to Government & Binding Theory*. Blackwell Publishers Ltd., Oxford, UK.

M. Kay. 1979. Functional grammar. In *Proceedings of the Berkeley Linguistic Society*.

Geert-Jan Kruijff, Elke Teich, John Bateman, Ivana Kruijff-Korbayová, Hana Skoumalová, Serge Sharoff, Lena Sokolova, Tony Hartley, Kamenka Staykova, and Jiří Hana. 2000. Multilinguality in a text generation system for 3 Slavic languages. In *COLING–2000: Proceedings of the 18th International Conference on Computational Linguistics*, Saarbruecken, Germany.

J. Oberlander, M. O'Donnell, C. Mellish, and A. Knott. 1998. Conversation in the musuem: Experiments in dynamic hypermedia with the intelligent labelling explorer. *The New Review of Hypermedia and Multimedia*, 4.

Cécile L. Paris, Keith Vander Linden, Markus Fischer, Anthony Hartley, Lyn Pemberton, Richard Power, and Donia Scott. 1995. A support tool for writing multilingual instructions. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1398–1404, Montréal, Canada.

M. Rayner, D. Carter, and Pierrette Bouillon. 1996. Adapting the core language engine to French and Spanish. In *Proceedings of NLP-IA-96*, Moncton, New Brunswick.

Jacques Robin. 1994. *Revision-Based Generation of Natural Language Summaries Providing Historical Background*. Ph.D. thesis, Columbia University, December.

Donia R. Scott. 1999. The multilingual generation game: Authoring fl uent texts in unfamiliar languages. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden.

Manfred Stede. 1996. *Lexical Semantics and Knowledge Representation in Multilingual Sentence Generation*. Ph.D. thesis, University of Toronto, Toronto, Ontario.

Incremental Generation by Incremental Parsing: Tactical Generation in Dynamic Syntax

Matthew Purver
Department of Computer Science
King's College London
Strand, London WC2R 2LS, UK
matthew.purver@kcl.ac.uk

Masayuki Otsuka
Department of Philosophy
King's College London
Strand, London WC2R 2LS, UK
masayuki.otsuka@kcl.ac.uk

Abstract

The paper shows how an incremental tactical generator can be constructed based on the incremental parsing framework described in Dynamic Syntax (DS)(Kempson et al., 2001), without adding a generator-specific vocabulary or intermediate levels of representation. The resulting generator is defined purely in terms of the parsing process, together with a notion of tree subsumption. This is shown to have various advantages including easy self-monitoring and psycholinguistic plausibility. A simple Prolog implementation is described, together with various possible improvements in efficiency.

1 Introduction

In this paper we give a description of a prototype tactical generator based on the DS approach. DS defines a formalism that allows the articulation of natural-language grammars that reflect left-to-right processing: natural-language strings are paired with decorated semantic trees by a process of monotonic growth over sequences of partial trees associated with the processing of each word. As such it incorporates a prototype parser as its central subpart. The generation process described here performs the reverse operation, producing possible output strings from a defined semantic tree. The generation method is defined entirely in terms of parsing, providing a tight parsing-generation correspondence allowing for self-monitoring without a separate parsing process, and a simple reflection of certain facts about natural language use in dialogue. We also describe a Prolog implementation of a DS system which incorporates both parser and generator, and discuss various possible improvements in efficiency.

Firstly we give some background on the DS formalism in section 2. We then describe our approach to generation in section 3 and the implementation in section 4. We then explain the computational and psycholinguistic advantages in section 5, and draw conclusions and outline further work in section 6.

2 Dynamic Syntax

DS is a parsing-directed grammar formalism which claims that parsing is the central mechanism of language processing. DS defines parsing as a process of establishing mappings from an initial tree to complete trees using general *computational rules* (roughly corresponding to syntactic rules) and specific *lexical actions* projected by lexical items in the input string. The resulting tree structures represent semantic interpretations for a given input string, and are described by the language **DU** based on a modal tree logic LOFT (Blackburn and Meyer-Viol, 1994).

Trees are either complete or partial. *Partial* trees have: (a) node addresses that are partially specified by underspecified modalities; or (b) node decorations that are underspecified by the use of requirements and meta-variables. Such partial specifications are always introduced with imposed requirements, which jointly constitute a set of constraints on possible tree growth. Trees are *complete* iff all such requirements are satisfied, leaving no remaining partiality or underspecification.

For example, in "left-dislocation", for which an initial constituent node is introduced into the tree related to the root node using an underspecified modality, this "unfixed" node bears the requirement that it has to be updated to a fully specified relation; and this requirement is met by later merging of this node with some local node which is in a fully specified relation to the root node. Similarly, if underspecified decorations are projected, they must be updated to fully specified values in the subsequent states (e.g. a pronoun projects a meta-variable which has to be substituted with a proper term).

The parsing model is defined to reflect the idea that parsing is a process of progressive and monotonic extensions of tree structures given the sequence of words as input. It is goal-directed, in the sense that all requirements must be satisfied for wellformedness (grammaticality).

Technically, a set of parse paths is a partially ordered set of possible trees where the initial tree is the lowest bound, and the goal trees are the highest bounds. The partial ordering $\leq_{ACT}$ on the set indicates a monotonic extension relation on pairs of trees, bearing a label of the name of the computational/lexical action performed in the transition.

The search for a parse path can be viewed as the search for a successful composition of actions which yields complete trees. The set of sequences of permissible actions is obtained by interweaving the linearly ordered set of lexical actions projectable by the input string into a finite number of partially ordered Kleene*-iterated computational actions, e.g. $ACT(\text{John sneezed}) = \langle C^* < john < C^* < sneezed < C^* \rangle$.

Generally, the initial tree has only one node bearing the requirement to establish a formula of type t, represented as $?Ty(t)$. The input string is scanned from left to right, with each lexical item projecting a lexical action to update the trees. Computational rules are transition functions on trees, which are conditional in the sense that if the input tree satisfies the condition of some computational rule, it yields the output tree updating the input tree. The parse is successful when the parser produces at least one complete tree having the root node decorated with a formula of type t, after using computational rules and all the lexical actions from the input string. In this case, the input string is called grammatical and the complete trees represent interpretations for the string. In other words, in DS, grammaticality of string is defined in terms of parsability.

DS specifies the theoretical parser, but, being a grammar formalism, it has no specification as to how a parser performs, i.e. no algorithm is defined giving the concrete parser. A parse state can be thought of as a pair of a string and a set of possible trees (semantic interpretations for the string). Without any parsing strategy, we may define the parser to generate all the possible trees using the computational rules (C-possible trees) for the input parser state, then scan the input string to apply the lexical action to all of the trees to update the parser state. This routine is repeated until the parser finishes scanning the input string.

3 Generation

As DS identifies grammaticality with parsability, rather than using concepts such as syntactic constituents or heads, standard approaches to generation such as head-driven methods (Shieber et al., 1990) cannot be applied. Instead, generation must be defined in terms of parsing.

3.1 Connecting Parsing and Generation

Despite informal observations made in psycholinguistics that production and comprehension systems have much in common (Garrett, 1982; Frazier, 1982), parsing and generation have generally been treated as quite separate research enterprises, both in the psycholinguistic and computational linguistic communities.

Nevertheless, following Shieber (1988), attempts have been made to define the two in terms of a broadly common architecture and using shared reversible grammars (see e.g. (Erbach, 1991; Neumann, 1994; Gardent and Thater, 2001)).

Although the contribution made by the present system is modest in only addressing the level of tactical generation, it nevertheless contributes to this co-articulation of parsing and generation by defining a generation system which purports to reflect the process of left-to-right incremental production, by using incremental parsing as the basic building block.

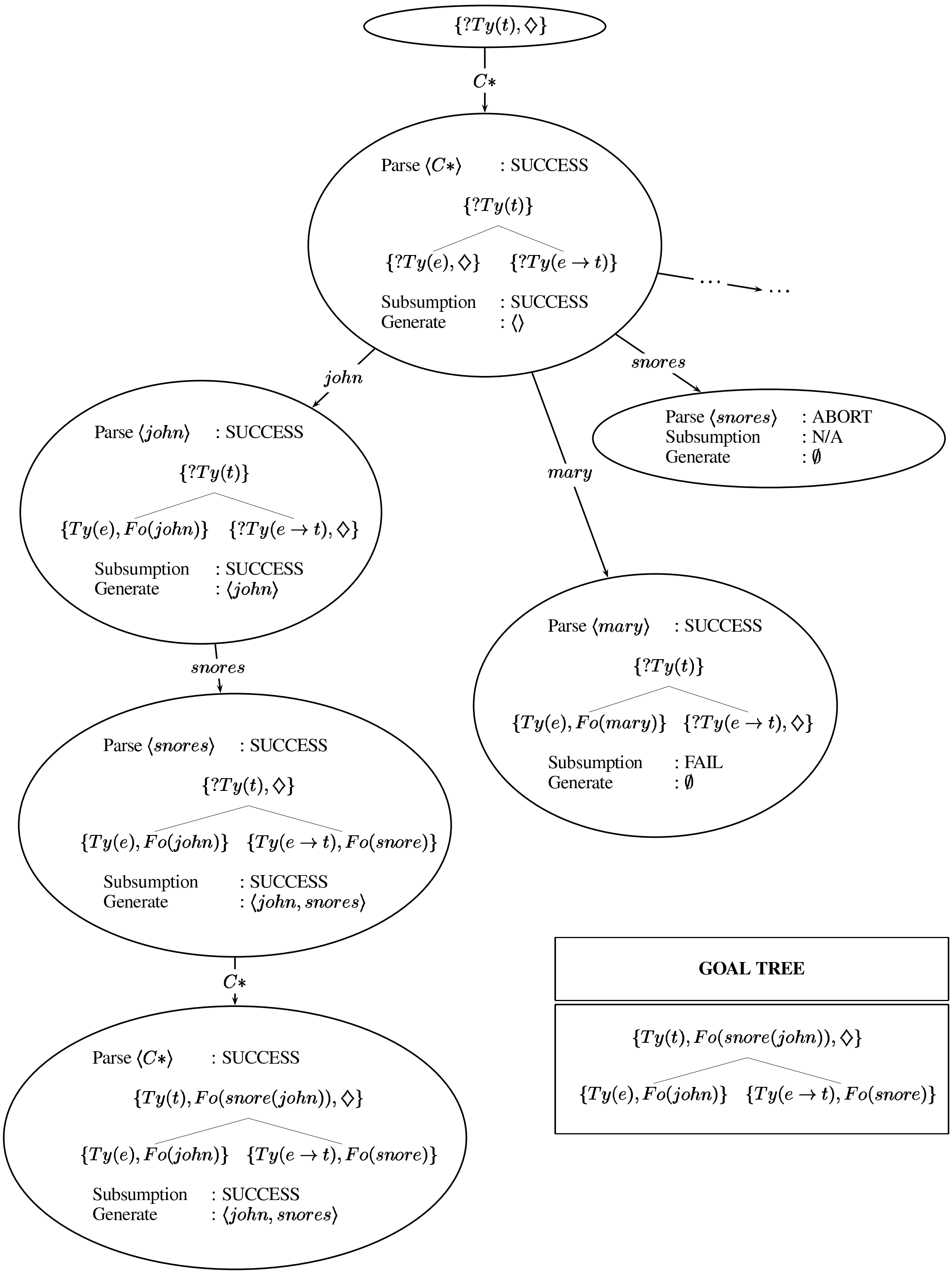


Figure 1: Generating *“john snores”*

3.2 Generation as Parsing

Generation can be achieved using very little beyond the standard DS notion of parsability. We assume a fully specified goal tree as input[1]: given this, the generator incrementally produces a set of corresponding strings by following standard parsing routines and using the goal tree as a check. In other words, for a naive generator, all that is required is a notion of tree subsumption: this allows any candidate partial tree produced in the generation process to be checked against the goal tree to determine whether it is a sensible candidate (by checking for tree mergeability without inconsistency in node relations and decorations).

At any point in the parsing process, a DS parser state can be seen as a pair of a partial string (the input so far) and a set of associated partial trees. With generation, we view the generator state as a set of these parser states (a set of pairs of 'possibly acceptable' partial strings and their associated sets of partial trees).

At each stage of generation, each pair is extended by tentatively extending the partial string by adding any word from the lexicon. The associated set of possible (partial) trees is produced using the standard parser – it may of course be empty if the word under consideration cannot be grammatically added to the partial string – and only those which are subsumed by the goal tree are kept. Any strings associated with empty tree sets are then rejected.

Generation is complete when the process can be continued no further (any string extension results in an empty parser state), and the set of output strings is taken as those for which the associated tree set contains a member identical to the goal tree – see figure 1.

Lexical selection is implicit: any word in the lexicon which is not associated with a node (or formula decorating a node) in the goal tree will produce empty tree sets (as the trees produced will not be subsumed by the goal tree), and therefore cannot produce acceptable extended partial strings.[2]

4 Implementation

The formal definition of DS species constraints of possible forms and extensions of tree structure and node decorations in terms of axioms (in LOFT) and algebraic definitions, and update actions (input/output relations of grammar rules) are defined in a way that they respect the constraints. This leaves room for implementation as to how the parser performs. This section describes the parser algorithm and possible strategies to improve efficiency.

4.1 Overview

A prototype DS system has been implemented in Prolog[3] The number of lexical entry types is currently small (i.e. the "grammar" is only small), and limited to English, but some relatively complex constructions such as left-dislocation and relative clauses can be processed.

The implementation remains close to the logical DS formalism. A tree node is represented as a pair of a node name and a set[4] of labels; a tree is represented as a pair of a set of nodes and a pointer (a node name). Labels can be requirements `?REQ`, directed requirements `?([DIR],REQ)` or features `+FEAT`.

There are some representational differences: mother-daughter relations are not expressed directly as labels but are implicit in the node naming scheme (e.g. node 0 has daughters 00, 01); LINKs are represented as 2-daughters; and unfixed nodes are *-daughters.

Computational actions can now be defined in terms of list manipulation. Prolog backtracking is used to allow any number of actions to be applied to any parser state (where possible).

Lexical actions are defined similarly and given as templates for individual parts-of-speech and

[1]In assuming a source tree, we are currently ignoring some issues that might be required in a dialogue system, such as concept generation, and translation from some flat meaning representation into a decorated source tree (not a trivial issue, as it is here that we will meet Shieber (1988)'s logical equivalence problem). We are therefore treating a generator as a module which supports that part of the generation process called sugaring (Ranta, 1994) or linguistic realisation (Reiter and Dale, 1997) – translation from an unambiguously structured object in a meaning representation language to a natural-language string.

[2]This may not be the most efficient way of rejecting unsuitable words – see section 4.2.3 below.

[3]The system can be accessed at `http://st228.dcs.kcl.ac.uk:8080/ds.`

[4]We represent sets as Prolog lists: no use is made of list order.

verb subcategorisation frames. The templates can then be interfaced to a standard computational lexicon (we are currently using one derived from the OALD (Hornby, 1974)).

Parsing is now easily defined declaratively: the initial parser state is one in which the only possible tree is a single root node with a $?Ty(t)$ requirement, and none of the input string has been consumed; a final parser state is one in which a complete (all requirements discharged) tree is available and the entire input string has been consumed; and possible intermediate states are produced from other states by any number of computational actions, plus lexical actions defined by the consumption of the next word from the input string.

4.2 Efficiency

4.2.1 Parsing

As described above, the current parser is highly unconstrained (it is defined declaratively and liberal use is made of Prolog backtracking). As generation is performed by parsing, any efficiency in parsing will be reflected in generation. Parsing efficiency can be improved by considering certain computational actions[5] as required whenever they can be applied (thus reducing the number of possible partial trees and the need for backtracking).

Further improvements in parsing may be possible by using e.g. probabilistic methods to constrain search, but this is outside the scope of the current work.

4.2.2 Generation

The naive generation process described in section 3 is not as inefficient as it might initially appear. Statically speaking, the basic concept is to generate all possible strings and check whether parsing them gives a sensible tree – therefore one might expect N^W possible paths given a lexicon of N words and typical string length W. However, the incremental nature means that this is not the case: unsuitable search paths are eliminated on a stage-by-stage basis (by the parser for ungrammatical paths, and by the subsumption check for paths incompatible with the goal tree).[6] The implementation of this naive version generates simple sentences in a few seconds with a small test lexicon.

However, we briefly explore efficiency improvements by lexical selection and search method.

4.2.3 Lexical Selection

Lexical selection uses the decorations of the input goal tree to produce a limited set of lexical items which can be used by the generator, consisting of a set L of words which correspond to logical formula decorations (e.g. N, V, Adj, Det) and a set F of functional words (e.g. relativiser, complementiser). This reduces the search space considerably, with the number of paths at most $(L + F)^W$ and in fact significantly less.

Members of L are chosen on the basis of goal tree node search followed by lexical lookup: only those words corresponding to $Fo(X)$ semantic formula decorations are admitted. Some members of F can be chosen on the basis of tree features (e.g. *LINK* structures for relativisers), although any defined in the grammar to have truly vacuous semantics must just be selected by default. For pronouns, node features can also be used (to admit only relevant gender & number), and similarly for verbs to control tense. This modification has been implemented and results in significant speed increase (although no formal corpus evaluation has yet been performed as our grammar is still small, even more complex sentences including relative clauses can now be generated in a few seconds with a large 40,000 word lexicon to search through).

Heuristics could be used to govern lexical lookup, based on e.g. recency of words in the current dialogue context (ensuring more recently used words are used in preference), and prior probabilities (ensuring more common words are used in preference to rarer ones).

4.2.4 Search Method

The incremental nature of the generation process allows us to apply various search methods,

[5]Currently *thinning*, *elimination*, and *star-adjunction*.

[6]In fact, in the best case there will be $N \times W$ paths, although this only applies for an unambiguous lexicon and grammar that gives a strict one-to-one correspondence between strings and trees.

and stop the process once any suitable string has been found (rather than continuing until all search paths are terminated). Using a depth-first approach with this method will therefore cause only one string to be produced; whereas using a breadth-first approach will cause only the set of all strings of the shortest possible length to be produced. Both approaches have been successfully implemented.

This provides scope for probabilistic (or other heuristic) methods to increase efficiency in the future by constraining the search using techniques well known in areas such as speech recognition (e.g. beam search). However, a prerequisite for this will be a theory of heuristically constrained DS parsing.

5 Computational and Psycholinguistic Features

The view of generation as incremental parsing appears to have certain advantages from both computational and psycholinguistic viewpoints. For computational dialogue systems, it ensures that self-monitoring information is available, and displays the property of incrementality. Psycholinguistically, as well as conforming to the general principle that production and comprehension are tightly coupled, it allows us to to explain certain phenomena characteristic of dialogue (where standard models of generation cannot), and therefore promises to meet the challenge set out by Pickering and Garrod (2003) that linguistic systems be evaluated by their success in reflecting such phenomena. This includes alignment between dialogue participants at many levels (lexical, syntactic and semantic), and the ability of participants to collaborate on and complete each other's sentences.

5.1 Self-Monitoring

Self-monitoring (the step-by-step monitoring of the generation process by an associated parse routine) has long been taken to be essential by psycholinguists (de Smedt and Kempen, 1990, for example). As discussed by Neumann and van Noord (1994), it can also be useful in a computational dialogue system, as it allows generation to be controlled from the point of view of the expected hearer. As well as checking parsability, it can be used e.g. to prevent ambiguous utterances (if a self-monitoring process can spot that a string under consideration during generation is ambiguous, that string can be excluded on the basis that it will be less clear for the hearer than other unambiguous alternatives).

With standard generation approaches, self-monitoring must be carried out in parallel by a parsing module which communicates with the generation module in some suitable manner. With our approach, however, self-monitoring comes built-in, as parsing is the building block for generation: all information that would be produced by a separate parsing process is already available and there is no need for a separate module.

The basic requirement for parsability by a hearer is of course guaranteed, as strings are produced by incremental parsing. Further control can be added easily by including the desired check at the subsumption-checking stage: if desired, rejection of ambiguous strings could be added by checking for partial trees that are *not* subsumed by the goal tree, and rejecting any parser paths which produce such trees (rather than just removing such trees from the path) – as long as alternative unambiguous paths are available.

5.2 Incrementality

The sense in which our generation process (and the DS parsing process) is *incremental* differs from that standardly used in both generation and psycholinguistic literature. In the generation literature (see e.g. (Erbach, 1991; Stone and Doran, 1997)), incrementality is taken to refer to the ability to produce substrings corresponding to (incomplete) sub-parts of the semantic representation, but without any requirement that this reflect left-to-right processing. This concept is useful for dialogue system architecture, in that processing bottlenecks in the semantic production module can be worked around by passing any complete sub-parts to an incremental generation module as they become available. In the psycholinguistic literature, "strong" incrementality (Sturt and Crocker, 1996; de Smedt and Kempen, 1990) is taken to require the projection of fixed tree relations as early and as close to word-by-word processing as possible[7] – and this progressive update enforces revision and backtracking.

Our use of the term is closer to the psycholinguistic concept in reflecting left-to-right parsing, with all relevant processing being complete after the addition of each word.[8] It is this monotonicity that allows unsuitable generation paths to be ruled out on a word-by-word basis (see section 4.2.2 above), and that allows us to select lexical items based just on a subsumption check, allowing selection to be economical compared to lexicalist approaches such as (Gardent and Thater, 2001)[9] where syntactic variation is encoded in distinct lexical entries, multiplying the number of generation search paths by the number of distinct structures licensed by the lexical item.

However, our approach can also be considered incremental in the computational sense: substrings can be generated from suitable goal subtrees with no change to the algorithm (the root node of the generation tree is merely given a node name which is not fully specified, and requirements which correspond to the root node of the goal sub-tree).[10]

5.3 Alignment

Our view of generation simplifies an explanation of the multi-level alignment described by Pickering and Garrod (2003). Standard generation models have little trouble explaining how lexical choice can be co-ordinated between dialogue participants (word preference can be adjusted based on recency) – but syntactic mirroring as observed by e.g. Branigan et al. (2000) is more problematic (grammar rule preferences must be adjusted, which is non-trivial).

In DS, alignment of syntactic and semantic structure comes by definition, given the reduction of syntax to the progressive projection of semantically transparent structure. Lexical alignment can be explained as a preference for repeated recent lexical items, bypassing the general lexicon search. Apparent alignment of syntactic structure over and above semantic alignment, as in repetition of double object constructions as opposed to a switch to the prepositional phrase equivalent, can be explained by exactly the same mechanism: since syntactic and semantic preferences are encoded in the lexical actions associated directly with words (rather than in any general grammar rules), so adjustment of syntactic and semantic constructions reduces directly to lexical choice.

5.4 Collaboration / Completion

The phenomenon of shared utterances, in which participants are able to switch roles and complete one another's utterances at any point in a sentence, is problematic for standard models of generation, as Pickering and Garrod (2003) observe.[11]

The treatment of generation as *incremental* parsing allows a simple explanation. Speaker and hearer are building the same partial trees in parallel, using the same parsing process and directly corresponding (if not perhaps identical) lexical entries: the only significant difference is that the speaker has knowledge of the goal tree that is being generated. If the hearer can guess or deduce the goal tree,[12] (s)he is in just as good a position to complete the utterance as the original speaker.

6 Conclusions and Further Work

6.1 Conclusions

The paper sets out a formal definition of a generation process for DS, in terms of the parsing process, and describes a naive computational implementation together with some possible improvements. In closing, we note that this (a) parsing-oriented, and (b) incremental view of generation

[7]Sturt and Crocker reflecton how different aspects of language processing are more or less incremental, with syntactic processing involving word-by-word update of the syntactic tree, semantic processing, eg pronoun resolution, allowing some delay.

[8]However, it differs from the concept of strong incrementality, by allowing the articulation of partial trees, in particular trees in which not all relations are uniquely determined.

[9]We are not claiming the overall approach is more economical.

[10]Though this approach remains to be fully worked out, some first steps in this direction have been taken in considering head-final languages such as Japanese.

[11]In particular, any shift prior to the sentence head is problematic for head-driven and LTAG approaches.

[12]We have nothing to say here about how this deduction is performed: of course it will depend on world and domain knowledge, the participants' shared situation and experience, and so on.

promises to provide a basis for an explanation of certain psycholinguistic observations about dialogue such as co-ordination and collaboration between speakers.

6.2 Further Work

The current system is a prototype which we intend to extend in two main directions. Firstly, efficiency can be improved by employing computational tactics such as probabilistic/heuristic parsing and generation, and possibly more goal-directed methods. Secondly, we intend to incorporate the parsing/generation model into a full dialogue system, to test out in detail the extent to which the apparent potential for modelling dialogue phenomena such as alignment, collaboration etc. can actually be fulfilled. This will require a detailed model of semantic tree creation and manipulation.

7 Acknowledgements

The authors would like to thank Ruth Kempson, Matthew Stone and two anonymous EACL reviewers for very helpful comments.

References

Patrick Blackburn and Wilfried Meyer-Viol. 1994. Linguistics, logic and finite trees. *Bulletin of the IGPL*, 2:3–31.

Holly Branigan, Martin Pickering, and Alexandra Cleland. 2000. Syntactic co-ordination in dialogue. *Cognition*, 75:13–25.

Koenrad de Smedt and Gerard Kempen. 1990. Incremental sentence production, self-correction and coordination. In G. Kempen, editor, *Natural Language Generation*, pages 365–376. Martinus Nijhoff, Dordrecht.

Gregor Erbach. 1991. A bottom-up algorithm for parsing and generation. CLAUS report, Universität des Saarlandes, Saarbrücken, February.

Lyn Frazier. 1982. Shared components of production and perception. In M. Arbib et al., editor, *Neural Models of Language Processes*, chapter 11, pages 225–236. Academic Press, New York.

Claire Gardent and Stefan Thater. 2001. Generating with a grammar based on tree descriptions: a constraint-based approach. In *Proceedings of the 39th Annual Meeting of the ACL*. Association for Computational Linguistics.

Merrill Garrett. 1982. Remarks on the relation between language production and language comprehension systems. In M. Arbib et al., editor, *Neural Models of Language Processes*, chapter 10, pages 209–224. Academic Press, New York.

Albert S. Hornby. 1974. *Oxford Advanced Learner's Dictionary of Current English*. Oxford University Press, third edition. With the assistance of Anthony P. Cowie and J. Windsor Lewis.

Ruth Kempson, Wilfried Meyer-Viol, and Dov Gabbay. 2001. *Dynamic Syntax: The Flow of Language Understanding*. Blackwell.

Günter Neumann and Gertjan van Noord. 1994. Reversibility and self-monitoring in natural language generation. In T. Strzalkowski, editor, *Reversible Grammars in Natural Language Processing*. Kluwer Academic Publishers.

Günter Neumann. 1994. *A Uniform Computational Model for Natural Language Parsing and Generation*. Ph.D. thesis, Universität des Saarlandes, Saarbrücken.

Martin Pickering and Simon Garrod. 2003. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, forthcoming.

Aarne Ranta. 1994. *Type-Theoretical Grammar*. Oxford University Press.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. In K. van Deemter and M. Stone, editors, *Formal Issues in Natural Language Generation*. CORSO C05.

Stuart Shieber, Gertjan van Noord, Robert Moore, and Fernando Pereira. 1990. A semantic head-driven generation algorithm for unification-based formalisms. *Computational Linguistics*, 16(1).

Stuart Shieber. 1988. A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 614–619. COLING.

Matthew Stone and Christine Doran. 1997. Sentence planning as description using tree-adjoining grammar. In P. Cohen and W. Wahlster, editors, *Proceedings of the 35th Annual Meeting of the ACL*, pages 198–205. Association for Computational Linguistics.

Patrick Sturt and Matthew Crocker. 1996. Monotonic syntactic processing: a cross-linguistic study of attachment and reanalysis. *Language and Cognitive Processes*, 11:448–494.

Acquiring and Using Limited User Models in NLG

Ehud Reiter, Somayajulu Sripada, and Sandra Williams
Department of Computer Science
University of Aberdeen
{ereiter,ssripada,swilliam}@csd.abdn.ac.uk

Abstract

It is a truism of NLG that good knowledge of the reader can improve the quality of generated texts, and many NLG systems have been developed that exploit detailed user models when generating texts. Unfortunately, it is very difficult in practice to obtain detailed information about users. In this paper we describe our experiences in acquiring and using limited user models for NLG in four different systems, each of which took a different approach to this issue. One general conclusion is that it is useful if imperfect user models are understandable to users or domain experts, and indeed perhaps can be directly edited by them; this agrees with recent thinking about user models in other applications such as intelligent tutoring systems (Kay, 2001).

1 Introduction

It has long been recognised that NLG systems should in principle generate texts that are targeted towards individual readers, and should use detailed models of the readers when doing so. The content of generated texts should be tailored to the reader's tasks and existing knowledge; for example, a weather forecast for a pilot landing an airplane should focus on wind and visibility at the destination airport, while a weather forecast for a farmer planting crops at a farm next to the airport should focus on temperature and precipitation. The expression (microplanning) of a generated texts should be tailored to the user's linguistic abilities and preferences; for example, a smoking-cessation letter sent to someone with an age 10 literacy level should use short sentences and simple words, while a smoking-cessation letter sent to a doctor with excellent literacy could use complex sentences and specialised medical terminology. And the realisation (for example, grammar) of a text could be tailored to a user's dialect, although this is perhaps more debatable. In other words, people are very different, and texts intended for individuals will be more effective if they can be targeted towards that individual.

In accordance with this accepted wisdom, many NLG systems and models allow detailed user models to be specified. For example, plan-based content determination (Appelt, 1985; Moore and Paris, 1993) is based on detailed models of user tasks and goals, and dialect or even ideolect grammars can be specified for realisation engines such as SURGE (Elhadad and Robin, 1997) and KPML (Bateman, 1997). Zukerman and Litman (2001) review how user models have been used in a variety of NLG and NLU systems.

Unfortunately, we are not aware of any NLG systems which actually use detailed user models with non-trivial numbers of users, probably because of the difficulty of acquiring detailed user models. Such models can of course be handcrafted for demonstration purposes for a single user performing a single task, but we are not aware of any successful systems based on detailed user models which work for a non-trivial number of real users.

The reality of NLG today is that any system with a non-trivial number of users has imperfect information about its users. It may know something about them, but its knowledge is far from complete. This raises an important question for NLG - what is the best way to acquire and use limited and imperfect information about users? Little has been published about this topic in the NLG literature; Zukerman and Litman's (2001) review, for example, says little about this topic, other than suggesting that perhaps user models can be built up during the course of a dialogue with the user.

We have struggled with this question in the course of building several NLG systems – IDAS, STOP, SUMTIME-MOUSAM, and GIRL – and used a different approach in each of these systems. In this paper we summarise the approaches we have taken and how well they seemed to work. We certainly do not have any definitive answers, but we hope our paper will at least clarify the issue. Also, perhaps one general lesson from our work is that it is helpful if imperfect user models are understandable to users or at least to domain experts. Under some circumstances this could allow people to edit and thus directly control their model; even if this is not possible, users are likely to be more helpful in the model acquisition process if they understand how the model is going to be used.

These observations fit in with recent thinking in the general user-modelling community. Fischer (2001) acknowledges that user modelling has been less successful than originally hoped, and suggests that in part this could be due to the difficulty of creating effective user models, and problems in dealing with models that are incorrect, incomplete, and/or out-of-date. Kay (2001) suggests that user models should be *scrutable* (understandable and modifiable) to the user, because (among other things) this allows users to understand what the system is doing and to correct mistakes.

Note that while research has been done on planning under uncertainty, the focus of such work is contexts where the uncertainty is well understood; for example, an object is in one of two locations (Collins and Pryor, 1995), or a diagnostic test has a well-understood false positive and false negative rate (Haddawy et al., 1995). Under these conditions it is possible to produce plans that are optimal under some cost or effectiveness criteria. However, in our experience the level of uncertainty in user models for NLG greatly exceeds what can be handled by such approaches.

2 IDAS: User in Control

One approach to the imperfect user knowledge problem is to generate a text using whatever user knowledge is available (which may not be much), and then allow the user to request additional information, clarifications, and so forth. This of course is the approach used by human speakers in dialogues, and indeed by many computer dialogue systems.

It was also used in IDAS (Reiter et al., 1995), an NLG system developed in the early 1990s which dynamically generated hypertext technical documentation from an AI knowledge base. IDAS had a facility which allowed detailed user models to be used during the generation process; but in fact no detailed models of real users were created for IDAS. Instead, users in experimental trials of IDAS would use its hypertext links to obtain more information if they needed it (Levine and Mellish, 1995).

For example, consider the question of how much detail should be in an instructional text on how to change a flat tire on a bicycle. Should such a text use high-level instructions such as *remove the front wheel*, or should it use more detailed instructions such as *lift the front wheel's quick-release lever*? The original IDAS vision was to make this choice on the basis of a detailed user model which stated which high-level actions the user already knew how to perform, and which needed to be broken down into substeps. However, in practice it was not possible to acquire such detailed information about IDAS users. Instead, IDAS produced a guess at an appropriate instruction sequence, often based on a coarse expert/novice distinction between users, and then allowed the user to click on a text if he or she wanted more information. For example, IDAS might initially produce the high-level text *remove the front wheel*, and the user could expand this into substeps (such as *lift* *the front wheel's quick-release lever*) by clicking on the original instruction.

Of course this strategy only makes sense if the user understands what questions he can ask. This requires both a good user interface and also an intuitive and easily understandable 'question space' (using IDAS's terminology) of what questions the system can answer.

Letting the user specify what he or she wants to know is not always possible; for example, it is not possible in a system which generates paper letters such as STOP, and may not be realistic in a system used by people with limited computer confidence such as GIRL. It also may sometimes be risky, for example if a user thought he knew how to remove a wheel but in fact did not. But it certainly seems to be an effective strategy in many situations, because users usually have excellent knowledge of their own goals, tasks, and expertise, much better than any computer system.

3 STOP: Ask User for Key Information

Another response to the difficulty of getting perfect user knowledge is to try to determine which knowledge about the user is most important, and then design a questionnaire or GUI to explicitly acquire this knowledge. In such cases we may need to impose a size or time-to-complete constraint on the questionnaire or GUI, based on what we think is realistic for the target user group.

This approach was used in STOP (Reiter et al., 2003), which generated personalised smoking-cessation letters. We devised a 4-page multiple-choice questionnaire for smokers based on what previous research suggested would be the most important information for the letter-tailoring process. The STOP software then used this questionnaire (which was completed on paper, and scanned into a database) as its primary information source when generating tailored smoking-cessation letters; some information was also obtained from the smoker's medical record. The user questionnaire data only effected content decisions in STOP; in principle it would have been desirable to also take user information into account when making microplanning and realisation (expression) decisions, but this was not done.

One problem we encountered was that in retrospect the information elicited by the questionnaire was perhaps not the most important information needed by the tailoring process. For example, although we asked users about their smoking habits, beliefs, and concerns, we did not directly ask them what information they would like to see in the generated letter (for example, medical information about the effects of smoking vs. practical how-to-stop advice); in hindsight we probably should have asked for this information. But this is not a flaw with the technique, it is a flaw in how we applied it.

A perhaps more basic problem is that we were limited to acquiring small amounts of well-structured information. We could not acquire a lot of information (since smokers would not spend more than 10-15 minutes on a questionnaire), and we could not acquire unstructured information such as free-text explanations of interests and goals (since such texts could not be interpreted and understood by our software). For example, one key issue in smoking advice is why previous attempts to quit have failed. Our questionnaire had seven check boxes for standard reasons such as stress or weight gain. It did not elicit detailed information which turned out to be very important to individual smokers, such as one person's frustration with hypnosis techniques, and another's promising attempt to quit being derailed by stress caused by the death of a relative.

Also, the questionnaire approach of course only works if the user understands and can answer the questions. For example, detailed medical information about the smoker's health, such as the condition of his lungs, would have been useful but in general we could not expect smokers to have this information. Another example is whether the smoker is addicted to nicotine (again important information for selecting appropriate cessation advice). Many smokers have incorrect beliefs about whether they are addicted, so instead of directly asking this question, STOP inferred addiction status from a set of questions devised by Fagerström and colleagues (Heatherton et al., 1991), such as whether the smoker smoked within 30 minutes of waking up.

In other words, in IDAS we believed that users themselves had the best knowledge of relevant information such as their tasks and goals. In STOP, however, we believed that users might not have good self-knowledge of some important information, such as addiction status.

In summary, a questionnaire can work well if we need a small amount of well-structured information, and we believe that users have (and will provide) this information. Otherwise, we should consider other approaches.

4 SumTime-Mousam: Domain Expert Creates a Model

Another approach to creating user models that usefully approximate reality is to get a domain expert (or 'knowledge engineer') to build the model. That is, the domain expert meets with users and discusses their needs and constraints, and from this develops a user model for a software system.

This approach was used in SUMTIME-MOUSAM (Sripada et al., 2002), which generates weather forecasts for offshore oil rigs; these are essentially summaries of the output of a numerical weather simulation, where the summarisation is controlled by a model of what is important to the user. Such forecasts are used by oil company staff to make specific decisions, for example on how to unload supply boats and when to schedule diving operations. If we had perfect knowledge about what decisions needed to be made and what the constraints on these decisions were (for example, what sea conditions were too rough for the particular diving equipment currently at a rig), then we could generate perfectly tailored forecasts using plan-based techniques. Unfortunately, this information is not available.

Instead, we included in SUMTIME-MOUSAM some parameters which made sense to expert meteorologists (such as what changes in wind speed were significant in different contexts), and let an expert set the parameters appropriately for different users. The expert had previously discussed with the end user what the user's tasks and needs were, and based the parameter settings on these discussions and on his expertise. These parameters essentially constituted a 'user model' which was intended to apply to an entire rig, and cover all types of operations.

We have not yet evaluated the usefulness of the generated forecasts with end users, so we do not know how effective this strategy is. However, we can make some observations. First of all, building such models requires making a tradeoff between the range of situations they cover and their effectiveness. A model which covered all possible decisions would insist that all data from the numerical simulation be communicated, and no data be summarised; this would negate the usefulness of textual summaries. On the other hand, a model that was tailored just to the most typical decisions would generate texts that were well suited to those situations, but not to others.

For example, in general light winds (less than 15 knots) have minimal impact on rig operations, so when the wind is light there is no need to report (for instance) a small change in wind direction, such as N to NNE. However, there are some unusual operations, such as flaring gas, when small changes in wind direction are relevant even with light winds. If we had perfect user knowledge we would report such changes if and only if the user was flaring gas or otherwise doing an operation for which this information was important. However, since we do not know what operations are planned for a particular day, we have to choose between either never reporting such changes (which somewhat improves forecast quality for most days by eliminating unnecessary information), or always reporting such changes (which greatly improves forecast quality on those rare days when the information is important).

One solution to this problem would be to allow users to explicitly specify information about their planned tasks and known constraints (such as whether they planned to flare gas), via a software package which was installed on the rigs. We have not seriously investigated this because of the substantial cost of developing, installing, and maintaining such a system, plus the cost of training users so that they entered the correct information at the correct time. In other words, SUMTIME-MOUSAM as it currently exists fits smoothly into current operational procedures and does not require users to install new hardware or software or change the way they work; this would not be true of a system which required users to enter information about their tasks and constraints.

Another issue with SUMTIME-MOUSAM's approach is that the parameters and model must be understandable to the domain expert. The first versions of SUMTIME-MOUSAM used fairly simple and hence understandable algorithms for data analysis and text generation, but the most recent version of the system uses more sophisticated algorithms which are less easy for someone who is not a computer scientist to understand. Hence it is harder for the domain expert to build models for the most recent version of the system than for previous versions.

Finally, economics means that domain experts cannot continually create new models, the models they creates must be usable for a period of time. This means that the usage of the texts must be fairly stable.

In summary, asking domain experts to build models that approximate how a group of users will use texts can work if the texts are used in ways that are predictable, limited, and stable; and if the user model is understandable to the domain expert. We expect this may be true in other NLG applications in addition to weather forecasts (financial summaries?), and encourage other researchers to consider this approach when it seems appropriate.

5 GIRL: Obtain Model by Testing Users

The final approach we have tried is building a model of a user's skills by testing his performance on a set of tasks, using an independently developed assessment test.

We are using this approach in a new system, GIRL (Williams, 2002), which generates reports on how well a student has done in a computer-based literacy assessment. From a research perspective, GIRL's focus is on making microplanning choices (aggregation, word choice, etc.) that are appropriate for the recipient's reading ability. For example, an aggregation decision that leads to a 30-word sentence is acceptable for a good reader but not for a poor reader. This requires knowing how well the recipient can read, and GIRL obtains this information from the literacy assessment which the student has completed. The assessment was independently developed by NFER-Nelson as a skill-based reading, writing and listening test for adult literacy learners. The results give a hierarchical model of derived literacy skills, with results of individual questions at the leaf nodes, results of skills tests at the next level, overall reading, overall writing and overall listening levels at the next and the overall literacy level at the root of the tree. Part of the experimental work in GIRL is determining which information from this model is most useful in guiding expression choices. The ultimate vision in GIRL is to create user models for readers which include this key information, and then use the reader model to control microplanning and perhaps also content and realisation choices in a text-generation system.

We cannot say much about the success or failure of this approach yet, as GIRL is still being developed and has not yet been evaluated. But certainly it seems likely that we cannot expect to obtain a large amount of information using this strategy; like the STOP strategy of asking a user to fill out a questionnaire, the strategy of testing the user is feasible if a small amount of information is needed, but not otherwise.

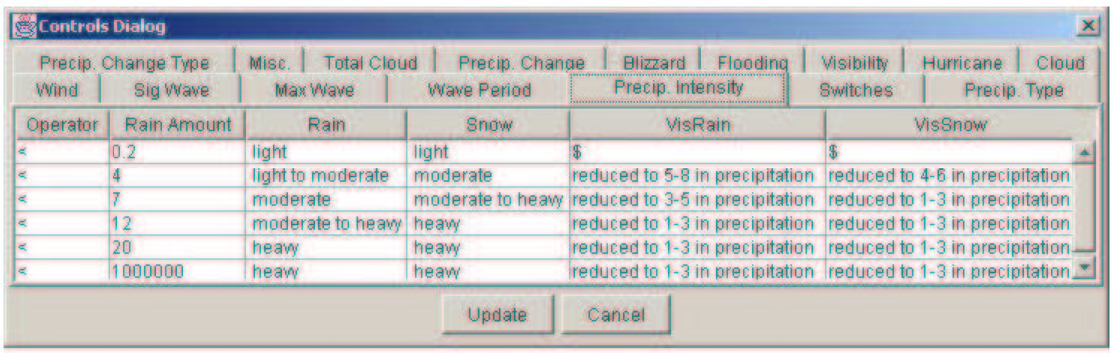
It is possible that in the long term, a lot of observational data about users will be available, which perhaps can be analysed and 'mined' for user information (Fitzgibbon and Reiter, 2002). For example, if we knew what web pages a person had read, we could probably make a plausible guess about his or her literacy level. This is an interesting possibility which should be kept in mind for the future, even if it is not realistic now.

6 Discussion

6.1 Understandability of User Models

One recurring theme in our work is that user models should be understandable to users or domain experts. If we had perfect user models, users perhaps would not care how they worked; but since we have imperfect user models, it is very helpful if users can understand them, either to edit them or to understand what the system is doing and how it might fail.

Kay (2001, page 118) discusses giving users access to and control over models about themselves in intelligent tutoring systems, and argues that this



Operator	Rain Amount	Rain	Snow	VisRain	VisSnow
<	0.2	light	light	$	$
<	4	light to moderate	moderate	reduced to 5-8 in precipitation	reduced to 4-6 in precipitation
<	7	moderate	moderate to heavy	reduced to 3-5 in precipitation	reduced to 1-3 in precipitation
<	12	moderate to heavy	heavy	reduced to 1-3 in precipitation	reduced to 1-3 in precipitation
<	20	heavy	heavy	reduced to 1-3 in precipitation	reduced to 1-3 in precipitation
<	1000000	heavy	heavy	reduced to 1-3 in precipitation	reduced to 1-3 in precipitation

Figure 1: Precipitation Intensity Options Box from SUMTIME-MOUSAM

is desirable because:

- Users have a legal and ethical right to access and control about themselves.
- Users can assess the correctness of the model, and repair mistakes.
- Users who have access to their user models will have a better idea of what the system is doing.
- Developers will be more accountable if models can be inspected by users, and will place more importance on making models understandable.
- Users may learn useful information about themselves from their user models.

Although the last reason may be specific to tutoring systems, we believe that the first four rationales give by Kay for giving users control over their models are likely to apply to NLG systems that use user models as well.

Many commercial software packages allow users to specify their preferences and needs via an options box, and we have in fact done this to a limited extent in SUMTIME-MOUSAM. A simple SUMTIME-MOUSAM options box is shown in Figure 1; this box allows the domain expert to specify how numerical precipitation figures (in mm/hr) should be translated into linguistic descriptors such as *heavy*. This is user-dependent because different readers interpret *heavy* in different ways (Reiter and Sripada, 2002). Among other things, the interpretation depends on location; a precipitation amount that would be considered *heavy* in a dry environment such as the Middle East might not be considered *heavy* in a wet environment such as Scotland.

Perhaps the key technical challenge to building such options boxes or other model-editing facilities is making choices understandable to users and domain experts. In particular we cannot expect such people to have expertise in linguistics; this is why the descriptors in Figure 1 are specified as actual adjectival phrases instead of as conceptual or semantic representations. We also cannot expect such people to have expertise in computer science; this is an issue in some of the other SUMTIME-MOUSAM options boxes, where the expert is expected to specify parameters that control a linear segmentation algorithm (Sripada et al., 2002). Finally, if the options box is intended to be filled out by users instead of domain experts, we should not expect an unrealistic amount of domain knowledge. For example the options box in Figure 1 may not be understandable to end users, because they may not understand precipitation expressed as mm/hr.

In short, we need to be able to present user model choices in an understandable way to people who do not have expertise in linguistics or computer science; how best to do this is an important topic for future research. This problem is likely to become even more acute if we try to learn user models from large amounts of observational data, as many learning algorithms do not produce results which are easy for people to understand.

6.2 Knowledge Source: Users or Domain (Communication) Experts

An important related issue is who supplies the information for a user model. In IDAS, STOP and GIRL the users themselves supply the basic information (which the system may make inferences from), but in SUMTIME-MOUSAM the information comes from a forecaster, that is a domain expert who knows the users. The forecaster is also a *domain communication* expert (Kittredge et al., 1991); that is, he is an expert in how to communicate information about the weather to users, as well as an expert in meteorology itself.

The disadvantage of using a domain (communication) expert is that he is supplying information secondhand, he does not know the users as well as the users know themselves. If we consider Figure 1, for example, presumably the users themselves have a better idea of what *heavy* means to them than the domain expert does. But on the other hand, the advantage of using a domain expert is that he will have more domain knowledge (for example, understand precipitation expressed as mm/hr), more understanding of how the NLG system works and how it is controlled by the user model, and also more domain communication knowledge (for example, know how *heavy* was defined for other users). Because he understands the domain, the user, and the system, the domain expert can be a good person to create a model that 'bridges the gap' between the user and the system, and specifies to the system, in terms it can understand, what is important to the user and how best to communicate with the user linguistically.

The other reason for acquiring user models from domain experts in SUMTIME-MOUSAM was that SUMTIME-MOUSAM weather forecasts are read by many people in an oil rig, not just a single individual. Hence the user model needs to be an approximation to a group of people, not a detailed description of a single person. In other words, the user model should record how the recipients on average interpret *heavy*, not how one individual interprets this word. In SUMTIME-MOUSAM we believed that it might in fact be easiest for an outside expert to create such a model, especially if the expert had experience in doing this for other user groups.

6.3 Cost of Mistakes

Another general issue that needs to be considered is the impact of getting the user model wrong. For example, what is the impact of careless mistakes in STOP questionnaires, of users failing to give the SUMTIME-MOUSAM domain expert complete information about all of their tasks and lexical interpretations, or of a GIRL subject doing poorly on a test because she hadn't had much sleep the previous night? Do such mistakes marginally decrease the utility of generated texts, or do they make generated texts completely useless? NLG systems that are not robust in the face of such mistakes may not be useful in real-world applications.

This may be an especially serious problem for systems such as STOP and GIRL which make inferences about the user based on the information he or she explicitly specifies. For example, if a smoker does not realise that STOP uses the 'do you smoke within 30 minutes of waking up' question to infer addiction level, and instead believes that this question is unimportant, then he or she may ignore it or respond very quickly without thinking about the question.

This suggests that at minimum we need to carefully design the user model acquisition tool (STOP questionnaire, SUMTIME-MOUSAM option boxes, GIRL literacy test) to minimise careless errors. It may also be sensible to check user models for plausibility and consistency (as STOP in fact does). Finally, this once again emphasises the desirability of users understanding what is in the user model and how the system uses it.

A related issue is how to update and maintain user models. People of course change over time, and user models should change with them. We are not aware of any previous research on how user models for NLG systems should be updated to reflect changes in the user's expertise or preferences. Probably the simplest approach here is simply to let users or domain experts directly edit and update a user model, which is possible in SUMTIME-MOUSAM.

7 Conclusion

One of the great promises of NLG is that it can tailor texts for individual users, but realising this promise requires that we know a lot about users, and in practice it can be difficult to obtain detailed information about the expertise, background, tasks, goals, and so forth of individual users. We have tried various approaches to acquiring information about users for NLG systems, including letting users implicitly specify models (IDAS), explicitly asking users to enter a model (STOP), explicitly asking a domain expert to construct a model (SUMTIME-MOUSAM), and implicitly inferring a model from a standard assessment test (GIRL). None of these approaches seem generally applicable, but all probably work better if users can easily understand their models, so that they can edit their model or at least better understand what the NLG system is doing.

Acknowledgements

Many thanks to Chris Mellish, Ingrid Zukerman and the anonymous reviewers for their helpful comments. This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC), under a PhD Studentship and grants GR/F36750/01, GR/L48812 and GR/M76681; and by the Scottish Office Department of Health under grant K/OPR/2/2/D318.

References

Doug Appelt. 1985. *Planning English Referring Expressions*. Cambridge University Press, New York.

John Bateman. 1997. Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, 3:15–55.

Gregg Collins and Louise Pryor. 1995. Planning under uncertainty: Some key issues. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1567–1573.

Michael Elhadad and Jacques Robin. 1997. SURGE: A comprehensive plug-in syntactic realisation component for text generation. Technical report, Computer Science Dept, Ben-Gurion University, Beer Sheva, Israel.

Gerhard Fischer. 2001. User modelling in human-computer interaction. *User Modeling and User-Adapted Interaction*, 11:65–86.

Andrew Fitzgibbon and Ehud Reiter. 2002. Memories for life: Managing information over a human lifetime. Technical Report AUCS/TR0207, Department of Computing Science, University of Aberdeen, UK.

Peter Haddawy, AnHai Doan, and Richard Goodwin. 1995. Efficient decision-theoretic planning: Techniques and empirical analysis. In *Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence*, pages 229–236.

Todd Heatherton, Lynn Kozlowski, Richard Frecker, and Karl-Olav Fagerström. 1991. The Fagerström test for nicotine dependence: A revision of the Fagerström tolerance questionnaire. *British Journal of Addiction*, 86:1119–1127.

Judy Kay. 2001. Learner control. *User Modeling and User-Adapted Interaction*, 11:111–127.

Richard Kittredge, Tanya Korelsky, and Owen Rambow. 1991. On the need for domain communication language. *Computational Intelligence*, 7(4):305–314.

John Levine and Chris Mellish. 1995. The IDAS user trials: Quantitative evaluation of an applied natural language generation system. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 75–93, Leiden, The Netherlands.

Johanna Moore and Cecile Paris. 1993. Planning text for advisory dialogues. *Computational Linguistics*, 19:651–694.

Ehud Reiter and Somayajulu Sripada. 2002. Human variation and lexical choice. *Computational Linguistics*, 28:545–553.

Ehud Reiter, Chris Mellish, and John Levine. 1995. Automatic generation of technical documentation. *Applied Artificial Intelligence*, 9(3):259–287.

Ehud Reiter, Roma Robertson, and Liesl Osman. 2003. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*. in press.

Somayajulu Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. 2002. Segmenting time series for weather forecasting. In *Applications and Innovations in Intelligent Systems X*, pages 105–118. Springer-Verlag.

Sandra Williams. 2002. Natural language generation of discourse connectives for different reading levels. In *Proceedings of the 5th Annual CLUK Research Colloquium*.

Ingrid Zukerman and Diane Litman. 2001. Natural language processing and user modeling: Synergies and limitations. *User Modeling and User-Adapted Interaction*, 11:129–158.

Generation of Video Documentaries from Discourse Structures

Cesare Rocchi
ITC Irst
Trento Italy
`rocchi@itc.it`

Massimo Zancanaro
ITC Irst
Trento Italy
`zancana@itc.it`

Abstract

Recent interests in the use of multimedia presentations and multimodal interfaces have raised the need for the automatic generation of graphics and especially temporal media. This paper presents an engine to build video documentaries from annotated audio commentaries. The engine, taking into consideration the discourse structure of the commentary, plans the segmentation in shots as well as the camera movements and decides the transition effects among shots. The output is a complete script of a "video presentation", with instructions for synchronizing images and movements with the playing of the audio commentary. The language of cinematography and a set of strategies similar to those used in documentaries are the basic resources to plan the animation. Strategies encompass constraints and conventions normally used in building video documentaries.

1 Introduction

In the last decade there has been an increasing interest in the generation of multimedia presentations and a growing tendency towards the use of multi-modal interfaces (Wahlster et al., 1993; Maybury, 1993). These interests have raised the need for automatic generation not only of natural language, but also graphics and especially temporal media (André, 2000).

In this paper, an engine to build video sequences of images starting from an audio commentary is described. The input for the engine is a representation of (possibly automatically) generated verbal commentary. The engine, taking into consideration the discourse structure of the commentary, retrieves the most appropriate set of images from an annotated database, plans the segmentation in shots as well as the camera movements and finally decides the transition effects among shots. The output of the engine is a complete script of a "video presentation", with instructions for synchronizing images and movements with the playing of the audio commentary.

The language of cinematography (Metz, 1974), including shot segmentation, camera movements and transition effects, is the basic resource to plan the animation and to synchronize the visual and the verbal parts of the presentation. In generating animations, a set of strategies similar to those used in documentaries are employed. Two broad classes of strategies have been identified. The first class encompasses constraints imposed by the grammar of cinematography, while the second deals with conventions normally used in guiding camera movements in the production of documentaries.

After a short discussion on related works, relevant concepts and terminology of cinematography are introduced in section 3. Section 4 briefly summarizes the Rhetorical Structure Theory (RST) for the analysis of discourse structure. In section 5 we present some of the heuristics that we have borrowed from the field of cinematography. In section 6 we illustrate the architecture of the engine and its parts. In section 7 we give some examples of how the engine works. Finally, in section 8, we outline conclusions and future work.

2 Related Work

One of the first case studies of the generation of "motion presentations" is the work of (Karp and Feiner, 1993). Their system generates scripts for animation using top-down hierarchical planning techniques. (Christianson et al., 1996) presents a successful attempt to encode several of the principles of cinematography in the *Declarative camera control language*.

Similar systems are BETTY (Butz, 1994) and CATHI (Butz, 1997). BETTY is an animation planner, which generates scripts for animated presentations. The CATHY system generates on-line descriptions of 3D animated clips for the illustration of technical devices, in the context of a coordinated multimedia document.

Animated presentations have been successfully employed also in multimodal frameworks for the generation of explanations (Daniel et al., 1999) and in learning environments (Bares and Lester, 1997).

The novelty of our approach lies in the use of rhetorical structure of the accompanying audio commentary in planning the video. In particular, knowledge of rhetorical structure is extremely useful in taking decisions related to the punctuation of the video, in order to reflect the rhythm of the audio commentary and its communicative goals. In our view, the verbal part of the documentary always drives the generation of the visual part.

3 Relevant concepts and terminology

According to Metz (1974), cinematic representation is not like a human language, which is defined by a set of grammatical rules. It is nevertheless guided by a set of generally accepted conventions. These guidelines may be used for developing multimedia presentations that can be best perceived by the viewer. Following, we briefly summarize the basic terminology of cinematography.

3.1 Shot and camera movements

The shot is the basic unit of a video sequence. In the field of cinematography a shot is defined as a continuous view from single camera without interruption. Since we only deal with still images, we define a shot as a *sequence of camera movements applied to the same image*.

The basic camera movements are *pan*, from "panorama", a rotation of the camera along the x-axis, *tilt* a rotation along the y-axis and *dolly*, a movement along the z-axis.

3.2 Transition effects

Transitions among shots are considered as the punctuation symbols of cinematography; they affect the rhythm of the discourse and the message conveyed by the video. The main effects are *cut* - the first frame of the shot to be displayed immediately replaces the last frame of the shot currently on display; *fade* - a shot is gradually replaced by (fade out) or gradually replaces (fade in) a black screen or another shot and *cross fade* (or dissolve) which is the composition of a fade out on the displayed shot and a fade in applied to the shot to be shown.

4 Rhetorical Structure Theory

Rhetorical Structure Theory (Mann and Thompson, 1987) allows the analysis of discourse structure in terms of dependency trees, with each node of the tree being a text span. Each branch of the tree represents a relationship between two nodes. One node is called the nucleus and the other is called the satellite. The information in the satellite relates to that found in the nucleus in that it expresses an idea related to what is said in the nucleus. For example, a *background* relation holds when a satellite provides a context to the information expressed in the nucleus. Figure 1 shows an example of a portion of a rhetorical tree. The second paragraph provides details with respect to the content expressed in the first paragraph. This additional information acts as a sort of reinforcement for what has been previously said in the first paragraph and consequently facilitates the absorption of information. In the original formulation by Mann and Thompson the theory posited twenty

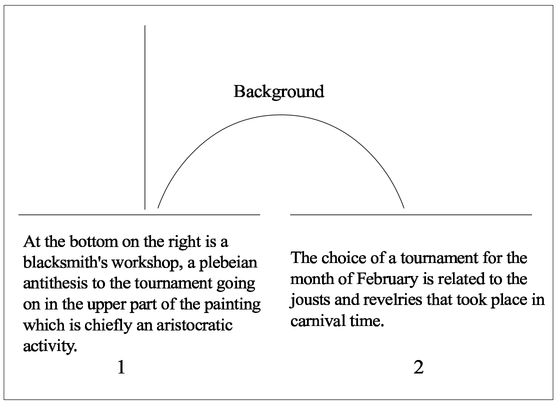

Background
At the bottom on the right is a blacksmith's workshop, a plebeian antithesis to the tournament going on in the upper part of the painting which is chiefly an aristocratic activity.
1
The choice of a tournament for the month of February is related to the jousts and revelries that took place in carnival time.
2

Figure 1: An example of Rhetorical Tree.

different rhetorical relations. From this original repository we borrowed a set of relations (elaboration, background, sequence and circumstance), which are commonly used in descriptive text, like those we have analyzed (see Section 6.1).

5 Heuristics and constraints of cinematography

Directors and film critics have identified several heuristics for making good movies. In designing a shot, it is important to consider the message that it has to convey and the (semantic) relations with the previous and following messages. Camera movements can be used to signal some of these semantic relations. For example, according to Arijon (1976), panning and tilting can be used to reveal spatial relations among objects and to move the watcher's attention from one center of interest to another; dollying can be employed to focus the attention on a particular zone or object previously displayed. For example, if an object is currently displayed and the following message deepens one aspect of it, a zoom on that aspect can be chosen.

Besides rules for movement selection, cinematographers have also identified a set of constraints on possible camera movement combinations, in order to ensure a pleasant presentation. In particular, each camera movement has to be "consistent" with respect to the previous movements. The watcher, looking at a movie in which camera moves to one side and then to the opposite one, can misunderstand the underlying message and experience some difficulties in following the stream of the presentation. For example, if the previous move is a pan towards the right the following effect cannot be a pan towards the left neither along the same path nor along similar paths. In general when a camera movement is chosen it constrains the choice of the following movements.

Another important feature of a movie is *cohesion*. A video sequence has to be a *continuum*, an uninterrupted stream in which each piece is connected to the others and is part of a whole. To achieve cohesion in designing the visual part of a presentation it is worth considering the relations among the new information to be delivered and those already given (*discourse history*) and to provide rhetorical strategies to build the presentation. The combination of rules and constraints encode some of the basic "principles of cinematography", which have been identified with the help of an expert director of documentaries (see also Arijon 1976).

Rules and constraints are the core on which the system relies. They encode the rhetorical strategies that are the basic resource for: (i) selecting appropriate images, (ii) designing the presentation structure, (iii) completing each shot, (iv) synchronizing the visual part with the audio commentary and avoiding the "seasickness" effect. Rules are formalized in a context sensitive "presentation grammar", are fired by a forward chaining mechanism and are relative to: (i) rhetorical relations among the text spans; (ii) the geometric properties of images selected from the information repository and (iii) the topics matching among segments and images. An example of rule is given in Figure 2. The rule applies when a segment has a relation of type background or circumstance; in that case the segment is assigned to a new shot.

Constraints are conditions that forbid particular combinations of camera movements and are tested

```
(defrule split (segment)
  (conditions
    (or (has-relation segment background)
        (has-relation segment circumstance)))
  (actions
    (init-shot shot)
    (add-segment segment shot)))
```

Figure 2: An example of rule for shot initialization.

```
(defconstraint zoom-in
  (var mv (get-previous-movement))
  (var mv2 (get-previous-movement mv))
  (and
      (not (equal mv zoom-out))
      (not (equal mv2 zoom-out))))
```

Figure 3: An example of constraint.

according to the type of movement proposed by the engine and the sequence of past movements. An example of constraint is shown in Figure 3. Potentially each camera movement can lead to an inconsistent sequence. To select a zoom-in movement it is worth considering whether the previous move or the penultimate one is a zoom-out; if not, then a zoom-in applies.

6 The Video Planner Engine

The engine is structured as in Figure 4. When a video for a given commentary is requested, the engine analyses the discourse structure of the commentary and selects an appropriate set of images to be presented. The generation chain consists of four phases:

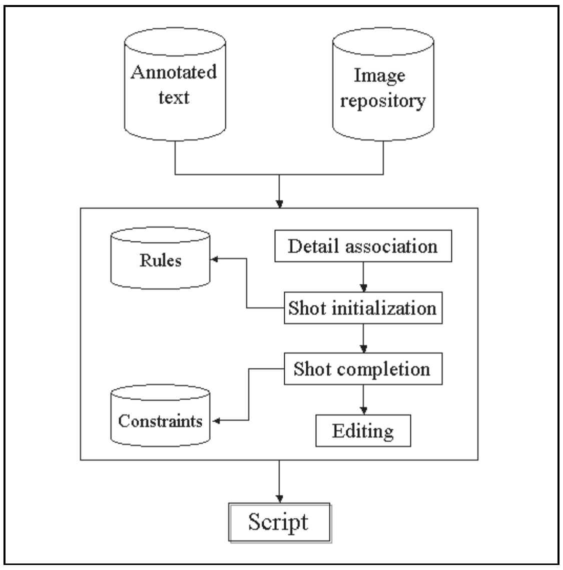


Figure 4: The system architecture.

Detail Association: a detail is associated with each segment of the commentary;

Shot initialization and structure planning: a candidate structure for the final presentation is elaborated, taking into consideration the rhetorical structure of the commentary;

Shot Completion: camera movements between details are planned. Constraints are considered in order to avoid "inconsistencies";

Editing: transitions among effects are selected according to the rhetorical structure of the commentary.

The output is a complete script for the video and the audio channels encoded in a renderer-independent markup language (see Figure 5).

```
<movie id="january">
  <shots>
    <shot id="shot603" image="det01">
      <video-track>
        <pause duration="2"/>
      </video-track>
      <audio-track>
        <play audio="january.wav"/>
      </audio-track>
    </shot>
    <shot id="shot605" image="det01">
      <video-track>
        <pause duration="1"/>
        <zoom duration="4" scale="4"/>
        <pause duration="2"/>
      </video-track>
      <audio-track>
        <audio-pause duration="3"/>
        <play audio="snowball-fight.wav"/>
        <audio-pause duration="1"/>
        <play audio="castle.wav"/>
      </audio-track>
    </shot>
  </shots>
  <editing>
    <display shot="shot603"/>
    <crossfade shot="shot605" duration="1"/>
  </editing>
</movie>
```

Figure 5: Example of script.

6.1 Resources

The video engine requires access to information about the structure of the data and a certain amount of knowledge about the domain.

As domain of application we have chosen the *Cycle of the Months* of Torre Aquila at the Buonconsiglio Castle in the city of Trento (Italy). This fresco is composed of eleven panels (each one representing a month) painted during the 1400s and illustrates the activities of aristocrats and peasants throughout the year.

As a case study we have collected a set of text that have been annotated by means of RST (see below). The nature of these texts, taken from a guide

```
<segment id="01" parent="root" relname="none"
topic="tournament" audio="castle.wav"
duration="3" >
At the bottom on the right is a blacksmith's
workshop, a plebeian antithesis to the
tournament going on in the upper part of the
painting which is chiefly an aristocratic
activity.  </segment>
<segment id="02" parent="01"
relname="elaboration" topics="castle"
audio="windows.wav" duration="2" />
The differences between the various styles of
construction have been reproduced extremely
carefully.
</segment>
```

Figure 6: Enriched RST annotation of a text.

of Torre Aquila, is descriptive and the prevailing rhetorical relations are elaboration, sequence, circumstance and background. At the moment we have favoured a sentence-by-sentence segmentation and the average size of the resulting trees ranges from seven to ten nodes.

The domain knowledge is encoded in a simple taxonomy, that is a set of keywords - called *topics* - representing entities, such as characters and animals, and processes, such as hunting and leisure activities. At this phase of the work, only one relation between topics is defined, the *member-of* relation, that denotes that a topic belongs to a particular class. For instance, the topic *fox_hunting* is in a *member-of* relation with the topic *hunting*, which means that *fox_hunting* is a form of *hunting*. At the moment, even if simple, knowledge representation is rich enough to accomplish our purposes.

The main input of the engine is a textual representation of the commentary annotated according to its rhetorical structure (see Figure 6). Additionally, the main concept of each segment is specified as well as the duration in milliseconds of the segment when played (although in Figure 6 the transcription of the commentary is shown, it is never used). Finally, the engine employs a database of images. For each images, the relevant details depicted have to be specified both in terms of their bounding boxes and in terms of the topics they represent. For example, Figure 7 illustrates the details for the panel of the month of January, annotated as in Figure 8. This picture consists of three main details: the snowball fight at the bottom (1), the castle at the top on the right (2) and the hunting scene (3), beside the castle. Within each detail it is possible to identify further details, as in the case of the castle, which contains the detail of windows (a).

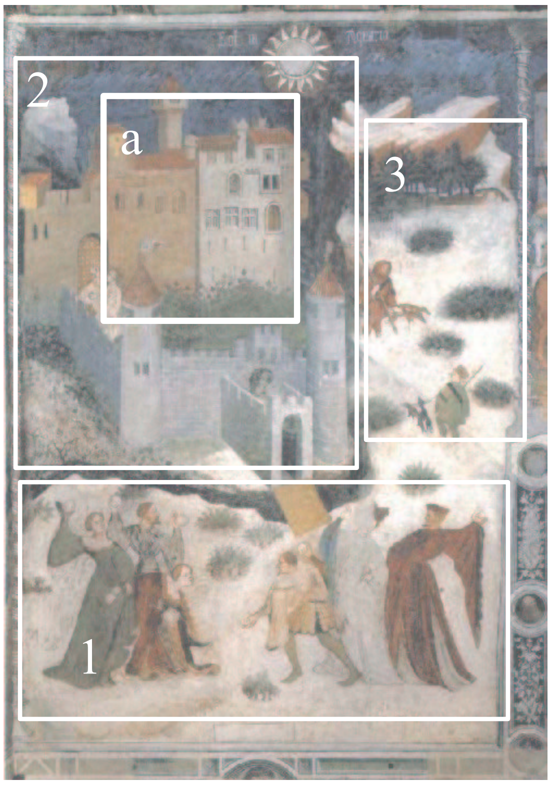


Figure 7: Details for the picture of January.

6.2 Phase 1: Detail association

In this phase the system assigns one or more details to each segment of the commentary. This operation is performed by searching the image repository for details with the same topic of the segment.

```
<db month="january">
  <image id="jan_img" source="january_full.jpg"
  height="713" width="500"/>
  ...
  <detail id="01" topic="january" parent="root"
    img="jan_img"  coords="0,0,500,713"/>
  <detail id="02" topic="snowball-fight"
    parent="01" img="january_img"
    coords="20,430,460,650"/>
  <detail id="03" topic="castle" parent="01"
    img="january_img" coords="12,50,330,430"/>
  <detail id="03a" topic="window1"  parent="03"
    img="january_img" coords="190,55,315,300"/>
  <detail id="04" topic="hunters" parent="01"
    img="january_img" coords="300,105,475,400"/>
</db>
```

Figure 8: Annotation of the image in figure 7.

6.3 Phase 2: Shot initialization

In this phase, shots are initialized taking into consideration the rhetorical structure of the commentary. At the moment the nucleus/satellite distinction is not taken into account. The result of phase 2 is a candidate structure for the final presentation. The processing is guided by a set of rules, which are fired when particular configurations of rhetorical relations are matched (see Figure 2). For example a relation of type elaboration or sequence signals a smooth transition from the current topic to new information that is strictly related to it; it is thus preferable to aggregate segments in the same shot and to exploit camera movements. Background and circumstance tend to highlight the introduction of new information that provides a context in which the following or the previous messages can be interpreted. They tend to break the flow of the discourse. It is thus preferable to split the segments in two different shots so that, in the next phase, it is possible to exploit proper transition effects in order to emphasize that change of rhythm. There are cases in which the structure planned in this phase is revised during successive stages of computation. For example, to avoid the ”seasickness” effect the system can apply constraints and then modify the previously planned structure by adding new shots (see examples in section 7).

6.4 Phase 3: Shot completion

This is the phase in which the engine incrementally completes each shot by illustrating each of its segments. In performing this task the engine traces the camera movements already planned. When a candidate move is proposed the system verifies whether it is suitable or not according to the list of past camera movements and the constraints imposed over that type of movement. Constraints encode the cinematographer's expertise in selecting and applying camera movements in order to obtain "well-formed" shots. For instance, when a panning movement is proposed where the previous movement is also a panning, the system has to check if the resulting sequence is suitable. Simple constraints include:

- When the previous movement is a dolly-out
  a dolly-in cannot be applied;
- When the previous movement is a dolly-in
  a dolly-out cannot be the subsequent movement;
- When a panning or a tilting is along a similar path and in the opposite direction of the previous movement
  that panning or tilting cannot be applied.

Constraints encode *schemes* of forbidden movements and when one of them is not satisfied the proposed move is rejected. In this case the engine initializes a new shot, declares the previous one completed and associates the remaining segments to the new shot.

6.5 Phase 4: Movie Editing

This is the phase in which the engine chooses the "punctuation" of the presentation. Movie editing is achieved by selecting appropriate transitions among shots. In order to reflect the rhythm of the discourse, the choice of transition effects is guided by the rhetorical structure of the commentary. The system retrieves the last segment of the shot displayed and the first segment of the shot to be presented and plans the transition according to the following rules:

- If two segments are linked by a relation of type elaboration
  a short cross fade applies;
- If two segments are linked by a relation of type background or circumstance
  a long cross fade applies.
- If two segments are linked by a relation of type sequence
  a cut applies.
- If a relation of type enumeration holds among two or more segments
  a rapid sequence of cut applies.

These rules have been selected according to the observations about the usual employment of transition effects in the field of cinematography (Arijon, 1976). Fade effects are fit for smooth transition, when there is a topic shift or when the center

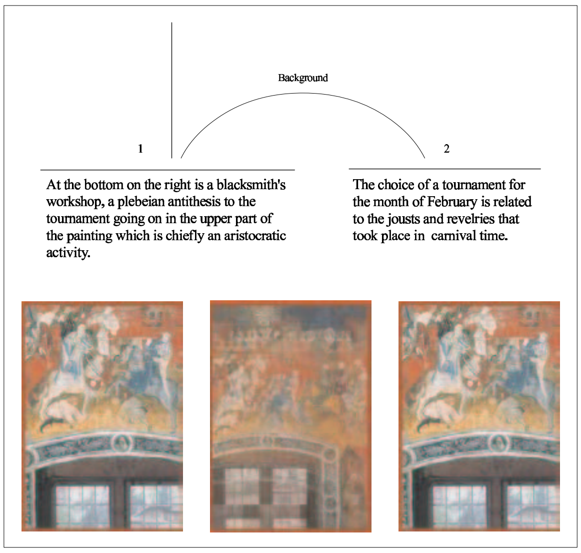

Background
1
2
At the bottom on the right is a blacksmith's workshop, a plebeian antithesis to the tournament going on in the upper part of the painting which is chiefly an aristocratic activity.
The choice of a tournament for the month of February is related to the jousts and revelries that took place in carnival time.

Figure 9: The "Tournament" example.

of interest changes but the new topic is related to the old one, as in the case of elaboration or background. Cut is more appropriate for abrupt and rapid changes, to emphasize the introduction of a new concept, as in the case of sequence. A special case holds when the verbal commentary enumerates a set of subjects or different aspects of the same object; in those cases a rapid sequence of cuts can be used to visually enumerate the elements described.

7 Examples

The first example concerns the rhythm of the discourse (Figure 9). Since the topic of both segments is the same, the text could be visually represented by displaying the same image during the playing of both the first and the second audio commentary. In this case a cross fade effect helps the user to understand that background information is going to be provided. In fact, the second segment provides contextual information to support the user in understanding the information presented in the first paragraph. The first image is thus presented while the audio of the first segment is played; then, when the audio switches to the second segment, the image is enlarged to cover the entire panel and finally refocused on the detail once the audio has stopped. By adopting this strategy the system generates a movie that reflects the discourse structure of the text and the rhythm of the discourse, supporting the same communicative goals of the verbal part of the presentation.

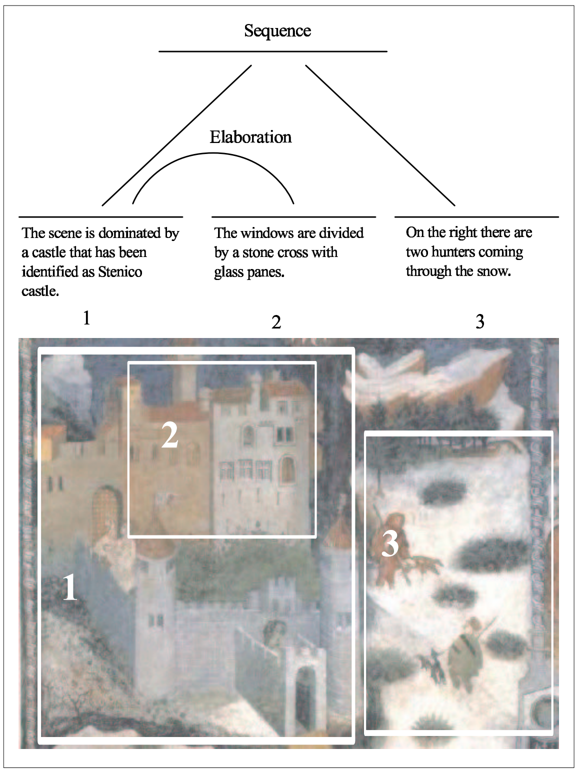

Sequence
Elaboration
The scene is dominated by a castle that has been identified as Stenico castle.
The windows are divided by a stone cross with glass panes.
On the right there are two hunters coming through the snow.
1
2
3
1
2
3

Figure 10: The "Castle" example.

The second example concerns the application of constraints in order to avoid an inconsistent sequence of camera movements (Figure 10). The text first describes the castle on the left. In this case the system, after a brief pause on the whole scene, selects a dolly-in movement, magnifying the detail of the castle (1). Then a second dolly-in is applied to focus on the castle's windows (2). Finally, in order to focus on the hunting scene (3) the camera should dolly out and then move towards right, but this combination is forbidden by the constraint on dolly-out. In this case the engine revises the structure of the movie. It declares completed the current shot, initializes a new shot and associates the remaining segments with it.

8 Conclusions and future work

In this paper we have presented an engine to generate video sequences starting from an audio commentary. First, we have identified a set of cinematic techniques that are the basic resources to plan the presentation. Second, we have shown how the resources (knowledge on the rhetorical structure of the commentary, knowledge about the domain and the repository of images) are annotated. Third, we have illustrated the architecture of the engine and the four steps of computation. Finally we have presented some examples, which show how the system employs rules and constraints to generate engaging presentations.

At the moment the system relies on a set of fifteen rules and ten constraints. Improvements are envisaged in particular to take into consideration the time needed to complete the movements (in this moment we assume a constant speed of the camera in movements) and more elaborated strategies to replan forbidden sequences of camera movements.

We have noted that the annotation of the resources (especially text) is time-consuming. In the future, in order to speed-up this task, we intend to investigate the possibility of a (semi-)automatic annotation of the discourse structure.

The application of the video clips in a mobile museum guide is currently under study (Zancanaro et al., 2003) and we are now experimenting with the techniques described here to automatically produce user-tailored videos.

Acknowledgments

This work has been supported by the PEACH and TICCA projects, funded by the Autonomous Province of Trento.

References

Elisabeth André. 2000. The Generation of Multimedia Documents. In *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text*, pages 305–327. Marcel Dekker Inc., New York.

Daniel Arijon. 1976. *Grammar of the Film Language*. Silman-James Press, Los Angeles, CA.

William. H. Bares and James. C. Lester. 1997. Realtime Generation of Customized 3d Animated Explanations for Knowledge-based Learning Environments. In *AAAI97 Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 347–354, Rhode Island, July 27-31.

Anreas Butz. 1994. BETTY: Planning and Generating Animations for the Visualization of Movements and Spatial Relations. In *Proceedings of Advanced Visual Interfaces*, Bary Italy.

Anreas Butz. 1997. Anymation with CATHY. In *Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, volume 1, pages 957–962, Providence, Rhode Island, July 27-31.

David B. Christianson, Sean E. Anderson, Li We He, David Salesin, Daniel S. Weld, and Michael F. Cohen. 1996. Declarative Camera Control for Automatic Cinematography. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference*, volume 1, pages 148–155, Portland, Oregon, August 4-8.

Brent H. Daniel, Charles B. Callaway, William H. Bares, and James C. Lester. 1999. Student-sensitive Multimodal Explanation Generation for 3d Learning Environments. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, volume 1, pages 114–120, Orlando, Florida, July 18-22.

Peter Karp and Steve Feiner. 1993. Automated Presentation Planning of Animation using Task Decomposition with Heuristic Reasoning. In *Proceedings of Graphics Interface*, pages 118–127.

William C. Mann and Sandra Thompson. 1987. Rhetorical Structure Theory: a Theory of Text Organization. In *The Structure of Discourse*. Ablex Publishing Corporation.

Mark T. Maybury. 1993. *Intelligent Multimedia Interfaces*. AAAI Press.

Christian Metz. 1974. *Film Language: a Semiotics of the Cinema*. Oxford University Press.

Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jrgen Profitilich, and Thomas Rist. 1993. Plan-based Integration of Natural Language and Graphics Generation. *Artificial Intelligence*, 63:387–427.

Preserving Discourse Structure when Simplifying Text

Advaith Siddharthan
Natural Language and Information Processing Group
Computer Laboratory, University of Cambridge
as372@cl.cam.ac.uk

Abstract

Text simplification involves restructuring sentences by replacing particular syntactic constructs (like embedded clauses and appositives). The aim is to make the text easier to read for some target group (like aphasics and people with low reading ages) or easier to process by some program (like a parser or machine translation system). However, sentence-level syntactic restructuring can wreak havoc with the discourse structure of a text, actually making it harder to comprehend, and possibly even altering its meaning. In this paper, we present and evaluate techniques for detecting and correcting disruptions in discourse structure caused by syntactic restructuring. In particular, we look at the issues of preserving the rhetorical relationships between the original clauses and phrases and preserving the anaphoric link structure of the text.

1 Introduction

Syntactic restructuring involves replacing particular syntactic constructs (like embedded clauses and appositives) in sentences. The aim is usually to reduce their grammatical complexity to make the text either easier to read for some target group (like aphasics and people with low reading ages) or easier to process by some program (like parsers or machine translation systems). When we constrain the restructuring operations to preserve the meaning and information content of the original text, we call the process *text simplification*. Syntactic transforms for dis-embedding relative clauses were first suggested as a preprocessing step for parsers (Chandrasekar et al., 1996; Chandrasekar and Srinivas, 1997) as they reduce sentence length and hence improve parser throughput. They were later used as part of a text simplification project aimed at making newspaper text accessible to aphasics (Carroll et al., 1999; Devlin, 1999). We illustrate syntactic simplification with an example. The sentence (1) a. contains two relative clauses and one conjoined verb phrase. Our text simplification system can simplify (1) a. to (1) b.

(1) a. Also contributing to the firmness in copper, the analyst noted, was a report by Chicago purchasing agents, which precedes the full purchasing agents report that is due out today and gives an indication of what the full report might hold.

b. Also contributing to the firmness in copper, the analyst noted, was a report by Chicago purchasing agents. The Chicago report precedes a full purchasing agents report. The full report is due out today. The Chicago report gives an indication of what the full report might hold.

A broad coverage text simplification system is expected to be useful to people with language disabilities like aphasia (Carroll et al., 1999; Devlin, 1999), adults learning English (by aiding the construction of texts that are of the desired linguistic complexity, while being relevant to adults), non-native English speakers surfing a predominantly English internet and users of limited channel devices (software that displays text in short sentences that fit on small screens could improve the usability of these devices).

Further, text simplification is useful as a preprocessing tool to improve the performance of other applications like parsing and machine translation (where performance deteriorates rapidly with

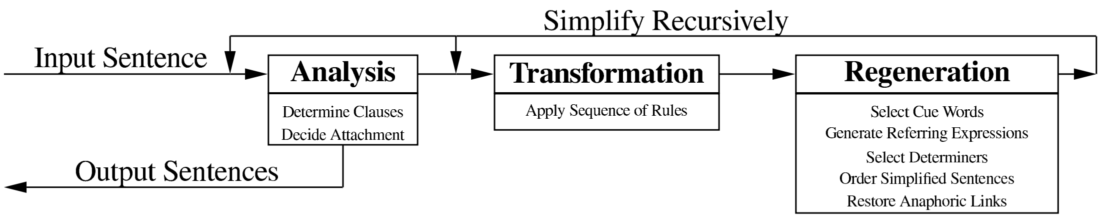


Figure 1: An Architecture for Syntactic Simplification

sentence length) and text summarisation systems based on sentence extraction (as simplified sentences contain smaller units of information).

Previous research on text simplification has not considered the discourse level issues that arise from applying syntactic transforms at the sentence level. Chandrasekar and Srinivas (1997), for example, use an architecture with two stages—*analysis* and *transformation*. There are various discourse level issues that arise when carrying out sentence-level syntactic restructuring of the sort illustrated by example 1. Not considering these discourse implications could result in the resultant text losing coherence, thus making it harder to read, or alter the intended meaning; in either case, making the text harder to comprehend. Our architecture (figure 1) therefore uses a third stage—*regeneration*, that we describe in this paper.

In section 2, we describe how to preserve the rhetorical relations (Mann and Thompson, 1988) that existed between clauses and phrases in the original text and ensure that we do not introduce spurious relations and conversational implicatures.

Applying syntactic transforms on text containing pronouns can cause further discourse level problems. In section 3, we discuss how syntactic transforms can result in discourse referents getting introduced in different orders, with different grammatical relations, and how this could make it hard for a reader (or program) to correctly resolve pronouns further in the text.

In section 4 we conduct a corpus evaluation of the techniques described in sections 2 and 3.

2 Preserving Rhetorical Relations

In this section, we discuss how generation issues like cue-word selection, referring expression generation, determiner choice and sentence ordering can be resolved so as to minimise disruption in the text's rhetorical structure. Then, in section 3, we show that the process of preserving rhetorical structure can unavoidably result in the destruction of the anaphoric link structure of a document and provide techniques to restore this structure.

2.1 Using Cue Words

Subordinating conjunctions connect clauses and make one of the clauses subordinate. Subordinating conjunctions also act as cue words that define the relationship between the conjoined clauses. When separating out the conjoined clauses, we can preserve the rhetorical relation between them by introducing a new cue word like *however* or *then*:

(2) a. **Though** all these politicians avow their respect for genuine cases, *it's the tritest lip service.*

b. All these politicians avow their respect for genuine cases. **However**, it's the tritest lip service.

(3) a. Kenya was the scene of a major terrorist attack on August 7 1998, ***when*** *a car bomb blast outside the US embassy in Nairobi killed 219 people.*

b. Kenya was the scene of a major terrorist attack on August 7 1998. A car bomb blast outside the US embassy in Nairobi killed 219 people **then**.

The table below gives a list of conjunctions (coordinating, subordinating and correlative) and the corresponding cue word that our algorithm introduces:

Conjunctions	**Cue Word**
although, though, whereas, but	however
or, or else	otherwise
even though	still
if, if...then	suppose...then
when	then
not only...but also	also
because, since, as	hence
and	

Our algorithm does not separate out conjoined clauses in cases where there is no appropriate cue word; for example, when the conjunction is *unless*. The placement of cue words is sentence initial, apart from *then* that is placed sentence finally (a sentence initial *then* would erroneously trigger a *chronological sequence* relation). The ordering of simplified sentences is described in section 2.4.

2.2 Generating Referring Expressions

In the examples above, the extracted clause had a subject and could be made into a stand alone sentence trivially. However, when splitting a sentence into two by dis-embedding a relative clause, we need to provide the clause with a subject. The referent noun phrase hence gets duplicated, occurring once in each simplified sentence. This phenomenon also occurs when separating out conjoined verb phrases and extracting appositives. We now need to generate a referring expression the second time, as duplicating the whole noun phrase can make the text stilted and cause unwanted conversational implicatures. For example, contrast:

(4) a. 'The pace of life was slower in those days,' says 51-year-old Cathy Tinsall, *who had five children.*

b. 'The pace of life was slower in those days,' says 51-year-old Cathy Tinsall. Cathy Tinsall had five children.

c. 'The pace of life was slower in those days,' says 51-year-old Cathy Tinsall. 51-year-old Cathy Tinsall had five children.

(4) c., apart from sounding stilted, emphasises Cathy Tinsall's age. This might, for example, inadvertently suggest to the reader that the relationship between her age and her having five children is important.

Existing referring expression generation algorithms (Reiter and Dale, 1992; Dale and Haddock, 1991) can't cope with open domains like newspaper text as they assume a classification of adjectives which is possible only for very restricted domains. We have proposed a new algorithm (Siddharthan and Copestake, 2002) that relies on WordNet synonym and antonym sets and gives equivalent results on the examples cited in the literature and improved results in other cases that prior approaches cannot handle. This algorithm is suitable for open domains like newspaper text and has been evaluated on the text-simplification task using Wall Street Journal data with promising results (summarised in section 4).

2.3 Determiner Choice

In example 4, the relative clause attached to a proper noun. However, in general, we have to decide on what determiners to use. This decision depends on the rhetorical relation between the extracted clause or phrase and its referent NP.

In the non-restrictive case (for either appositives or relative clauses), the rhetorical relation is that of *elaboration*. This relation continues to hold when we make the clause into the second sentence:

(5) a. A former ceremonial officer, who was at the heart of Whitehall's patronage machinery, said there should be a review of the honours list.

b. A former ceremonial officer said there should be a review of the honours list. *This* officer was at the heart of Whitehall's patronage machinery.

For extracting non-restrictive constructs, we only need to ensure that the referring expression contains a definite determiner. The determiner *this* is stronger than *the* and can only be used if there is no future reference that uses the determiner *the*.

When simplifying restrictive clauses, the relationship between the clause and the referent noun phrase is that of *specification*; that is, identifying a member (or some members) from a larger set. To preserve this, we require an indefinite determiner (*a* or *some*) in the noun phrase that the clause attaches to. This has the effect of introducing the member(s) of the larger set into the discourse:

(6) a. The man who had brought it in for an estimate then returned to collect it.

b. *A* man had brought it in for an estimate. *This* man then returned to collect it.

The indefinite article is not introduced if the NP contains a numerical attribute (eg. ***...two conversions** which turned out to be crucial.*). The referring expression contains a definite determiner as usual. The algorithm for selecting determiners is:

Algorithm *select_determiner*

```
1. IF restrictive clause THEN
      IF head noun is not a proper noun AND NP does
      not contain a numerical attribute THEN
         introduce indefinite determiner (a or some)
         in NP in the first sentence

2. IF no future references to the NP THEN
      introduce this or these in referring expression
   ELSE introduce the in referring exp.
```

2.4 Sentence Order

In general, the clause order should be preserved in the transformed sentences. However, there are a few exceptions. In the following example, the lack of a suitable cue word for the *reason* relation forces us to change the clause ordering and use the cue word for the *consequence* relation.

(7) a. The "unengageable" element of the welfare population is rising ***because*** *the city is playing reclassification games.*

b. The city is playing reclassification games. Hence the "unengageable" element of the welfare population is rising.

We also need to reverse the clause ordering when extracting non-restrictive clauses that attach to noun phrases in the subject position; the *elaboration* clause or phrase has to come second. This is illustrated in example 5 above. Also, the *elaboration* relation tends to get lost if the second sentence is separated from the noun phrase being elaborated by too much text. This can happen if the first sentence is very long, or if it contains another construct to be simplified. Consider:

(8) a. The agency, ***which*** *is funded through insurance premiums from employers* , insures pension benefits for some 30 million private-sector workers who take part in single-employer pension plans.

b. The agency is funded through insurance premiums from employers. The agency insures pension benefits for some 30 million private-sector workers. These workers take part in single-employer pension plans.

c. The agency insures pension benefits for some 30 million private-sector workers. These workers take part in single-employer pension plans. The agency is funded through insurance premiums from employers.

It is obvious that the ordering (8) b. is less disruptive than (8) c. In such cases, using sentence order to preserve rhetorical relations is counterproductive and we make the extracted clause the first sentence.

Algorithm *sentence_order*

```
1. order = "preserve"

2. IF cue word introduction changes clause order THEN
     order = "reverse"

3. IF non-restrictive clause THEN
     IF referent NP is a subject THEN
       order = "reverse"

4. IF length(sent_1)-length(sent_2) > threshold THEN
     order = "reverse"

5. IF sent_2 can be simplified further THEN
     order = "reverse"

6. RETURN order
```

3 Preserving Anaphoric Structure

Syntactic restructuring that involves splitting sentences or changing their voice can change the grammatical function of NPs and alter the order in which they are introduced into the discourse. This can affect the reader's ability to correctly resolve pronouns further in the text. If we cannot ensure that the most salient (Lappin and Leass, 1994; Kennedy and Boguraev, 1996) entities before simplification remain the most salient after simplification, we have to consider the possibility of broken anaphoric links. We do this in section 3.2.

When syntactic restructuring reverses the original clause order, this disruption in the anaphoric link structure can become evident in the restructured sentences themselves. We illustrate this in section 3.1.

In both cases, our approach is the same; we make use of a pronoun-resolution algorithm in deciding what to generate. The discussion in this section is based on *salience* and our implementation uses a shallow version of the Lappin and Leass (1994) algorithm. It is worth pointing out in advance that in the examples that follow, we use the term *salience* to mean "salience, as calculated by our algorithm", which may differ slightly from other calculations that use differently weighted features.

We use a three sentence discourse window containing the sentence to be simplified and the two previous sentences and calculate the salience of entities at the end of this window. We then simplify the required sentence, splitting it into two or changing its voice from passive to active. We then check that any pronouns in this sentence resolve to the same antecedents in the original and simplified text. If not, we need to replace them with referring expressions. This process needs to continue till the relative salience of entities in the original and restructured text is the same, at which point we know that the resolution of future pronouns will not be affected by our restructuring. We now illustrate the process with examples.

3.1 Problems with reversing Clause Order

Consider:

(9) a. Incredulity is an increasingly lost art.

b. It requires a certain self-confidence to go on holding the line that Elvis Presley [1] isn't in an underground recording studio somewhere.

c. David Beckham[2] is prone to provoking revisionist hints because the virtues he[2] represents are rare not only in the general population but especially so in football.

When we restructure sentence (9) c. into (9) c'. below, we need to check that the pronouns continue to refer to the same antecedents.

(9) c'. The virtues **he**[1] represents are rare not only in the general population but especially so in football. Hence, David Beckham is prone to provoking revisionist hints.

Our salience-based pronoun resolution system resolves *he* to *David Beckham* in the original text, but incorrectly to *Elvis Presley* in the restructured text. We therefore need to replace *he* by *David Beckham* (its antecedent in the original text). We then check whether the *David Beckham* in the second sentence would, if replaced by the pronoun *he*, still be interpreted correctly. Our pronoun resolution system tells us it will. Hence we can safely simplify sentence (9) c. to (9) c". below:

(9) c". The virtues David Beckham[2] represents are rare not only in the general population but especially so in football. Hence, he[2] is prone to provoking revisionist hints.

Algorithm *fix_restructured_sentence*

```
1. FOR every pronoun in restructured sentences DO 2-3
2. resolve pronoun in original and restructured text.
3. IF they are not the same THEN
   (a) replace pronoun in restructured text with referring expression for antecedent in original text
   (b) IF that antecedent NP has ended up in second sentence THEN
        i. temporarily replace that NP with pronoun
       ii. check that it resolves correctly.
      iii. IF it does THEN
              make the replacement in $i$ permanent
           ELSE
              withdraw replacement in $i$
```

We still need to ensure that future anaphoric links are not affected. This is described next.

3.2 Fixing future Anaphoric Links

We now describe how we can tell when future anaphoric links will be affected, and how we can fix disrupted links.

3.2.1 Transforms that preserve Relative Salience

In example 9 above, the five most salient classes at the end of sentence (9) c. in the original text are:

David Beckham, revisionist hints, virtues, general population, football

The five most salient classes at the end of sentence (9) c". in the restructured text are:

David Beckham, revisionist hints, virtues, general population, football

We find that the relative salience of entities is preserved. This tells us that the reader will be able to resolve future pronouns correctly.

For another example, consider:

(10) a. The Supreme Court agreed to decide whether the federal Pension Benefit Guaranty Corp. may require LTV Corp. to reassume funding responsibility for a $2.3 billion shortfall in the company's pension plans .

b. The high court's decision may affect the stability of many large corporate pension plans that have relied on the availability of pension insurance provided by the federal insurance agency.

c. The agency[1], which is funded through insurance premiums from employers , insures pension benefits for some 30 million private-sector workers who take part in single-employer pension plans.

At the end of sentence (10) c., the top 5 salience classes are (in order):

agency, pension benefits, 30 million private-sector workers, part, single-employer pension plans

When we split sentence (10) c. the first time, we choose to order the simplified sentences as (10) c'. (sentence (10) c. is the same as sentence 8 that was dealt with in section 2.4).

(10) c'. The agency[1] is funded through insurance premiums from employers. The agency[1] insures pension benefits for some 30 million private-sector workers who take part in single-employer pension plans.

When sentence (10) c. is replaced by (10) c'. the top 5 salience classes are:

agency, pension benefits, 30 million private-sector workers, part, single-employer pension plans

Again, we find that the relative salience of entities is preserved by this transform and hence future anaphoric links will not be disturbed. As an illustration of this, consider the sentence following the simplified sentence (10) c'. in the original text:

(10) d. It[1] recently reported assets of $2.4 billion and liabilities of $4 billion.

Our anaphora resolution algorithm resolves the pronoun (*it*) in sentence (10) d. identically (to *agency*) for the simplified and original texts, suggesting that we can safely leave it as it is.

3.2.2 Transforms that alter Relative Salience

If a clause attaches to a non-subject NP, the discourse structure is invariably disturbed. Consider:

(11) a. Back then, scientists[1] had no way of ferreting out specific genes, but under a microscope they[1] could see the 23 pairs of chromosomes in the cells that contain the genes.

b. Occasionally, gross chromosome damage was visible.

c. Dr. Knudson[2] found that some children with the eye cancer had inherited a damaged copy of chromosome No. 13 from a parent[3], who had necessarily had the disease.

At the end of sentence (11) c., the top 5 salience classes are:

Dr. Knudson, children, damaged copy, parent, eye cancer

When we split the last sentence, we have the choice of ordering the simplified sentences as either of (11) c'. or (11) c".

(11) c'. A parent[3] had necessarily had the disease. Dr. Knudson[2] found that some children with the eye cancer had inherited a damaged copy of chromosome No. 13 from the parent.

c". Dr. Knudson[2] found that some children with the eye cancer had inherited a damaged copy of chromosome No. 13 from a parent. The parent[3] had necessarily had the disease.

When sentence (11) c. is replaced by (11) c'., the top 5 salience classes are:

Dr. Knudson, children, damaged copy, parent, eye cancer

When sentence (11) c. is replaced by (11) c"., the top 5 salience classes are:

parent, disease, Dr. Knudson, children, damaged copy

There is now a conflict between preserving the discourse structure in terms of anaphoric links and preserving the discourse structure in terms of rhetorical relations. The non-restrictive relative clause has an *elaboration* relationship with the referent NP. To maintain this *elaboration* relationship after simplification, the dis-embedded clause needs to be the second sentence, as in (11) c". However, this ordering significantly disrupts the relative salience of different entities that is more or less preserved by the ordering (11) c'. This conflict between picking the ordering that preserves anaphoric links and the ordering that preserves rhetorical structure is unavoidable as the simplification process places a noun phrase that was originally in a non-subject position in a subject position, hence boosting its salience. Our solution is to select the ordering that preserves rhetorical structure ((11) c".) and detect and then fix broken anaphoric links as described next.

We detect and fix broken anaphoric links as follows. We consider each sentence following the simplified sentence. For each pronoun we encounter, we use our anaphora resolution procedure to find its antecedent in both the original and simplified texts. If the antecedents differ, we replace the pronoun by a referring expression for its correct antecedent (determined using the original text). The salience scores are then recomputed. This process continues until the relative salience of entities in the original and simplified text are the same again.

Now consider the sentence that follows the simplified sentence (11) c.

(11) d. Under a microscope he$^{original:2,\ simplified:3}$ could actually see that a bit of chromosome 13 was missing.

Our anaphora resolution algorithm resolves the pronoun *he* in sentence (11) d. to *Dr. Knudson* in the original text, but incorrectly to *parent* in the simplified text. To preserve the meaning of the original text, we need to replace the pronoun in the simplified text with a new referring expression for its antecedent in the original text. Thus we, replace (11) d. with (11) d'. below:

(11) d'. Under a microscope Dr. Knudson[2] could actually see that a bit of chromosome 13 was missing.

Now, we find that at the end of this sentence, the five most salient classes are the similar for the original text:

Dr. Knudson, microscope, bit, chromosome, children

and the simplified text:

Dr. Knudson, microscope, bit, chromosome, parent

This tells us that future anaphoric link will not be disrupted by our simplification process.

This process of fixing anaphoric links looks quite daunting. However, in practice, as salience decreases rapidly at sentence boundaries, we rarely (in only 2% of the cases; refer to section 4 on evaluation) have to consider more than just the one sentence succeeding the transformed one. Hence the loop in step 2 below is rarely executed more than once.

Algorithm *fix_future_links*

```
1. IF relative salience of entities in original and transformed text is same, THEN $flag$ = 0 ELSE $flag$ = 1
2. WHILE $flag$ DO steps 3 and 6
3. FOR every pronoun in next sentence DO 4-5
4. resolve pronoun in original and transformed text.
5. IF they are not the same THEN
       replace pronoun in transformed text with referring expression for antecedent in original text
6. IF relative salience of entities in original and transformed text is same, THEN $flag$ = 0 ELSE $flag$ = 1
```

4 Evaluation

For many of the algorithms presented in this paper, evaluation is difficult. It is hard to quantify the effects of text restructuring on a text's discourse-level structure. The effects of many regeneration decisions (eg. cue word selection and sentence ordering) on the regenerated text are largely stylistic, which makes evaluation subjective.

The referring expression generator gives *correct* results on ˜81%, *acceptable* results on ˜12% and *wrong* results on ˜7% of cases, when evaluated on WSJ data (Siddharthan and Copestake, 2002). A generated referring expression was labelled as *correct* if it was optimal and factually accurate, as *acceptable* if the generated expression was accurate but suboptimal and as *wrong* if the generated expression was nonsensical or ambiguous with a distractor. The mistakes mainly arose due to multi-word expressions being incorrectly analysed as multiple attributes to generate, for example, *the care products* from *personal care products.*

That evaluation considered only examples where there were one or more distractors in context. However, in over 90% of the cases for which we need to generate referring expressions, the contrast set of distractors is empty, which means the error rate for our application is less than 1%.

For a preliminary evaluation of the other regeneration components, we used a corpus of newspaper columns and news reports, travelogues, medical articles and literary extracts and manually examined the output of our text simplification algorithm on the first 250 embedded clauses.

Our method for selecting determiners gave wrong results on ˜4% of examples. The following examples show the output of our algorithm for two sentences. In example 12, the adjectival pronoun *his* would have been preferable to *this* in the referring expression. In example 13, the referring expression should have had the indefinite determiner *a*.

(12) a. Puckett played in 10 All-Star games during his career, which was cut short by glaucoma.

b. Puckett played in 10 All-Star games during his career. **This** career was cut short by glaucoma.

(13) a. Petroleum companies were also popular because of expectations of a weaker dollar, which cuts crude-oil prices.

b. Petroleum companies were also popular because of expectations of a weaker dollar. **This** weaker dollar cuts crude-oil prices.

The algorithms on preserving anaphoric links can be evaluated more objectively. 20% of the cases contained pronouns in the sentence to be simplified. Assuming that salience based anaphora resolution algorithms perform with an accuracy of ˜0.65 on open domains (Barbu and Mitkov, 2001; Preiss, 2002), algorithm *fix_restructured_sentence* can be expected to have an error rate of around $20 \times 0.35\% = \tilde{\ }7\%$. In practice, our algorithm made mistakes in only ˜2.5% of the cases. This is because our anaphora resolution algorithm performs with an accuracy of ˜0.80 on our corpus, and because intra-sentential pronouns are relatively easy to resolve.

The loop in algorithm *fix_future_links* needed to be executed only once in 98% of the cases. In the remaining 2% cases, the loop needed to be executed twice. 15% of the cases contained anaphora in the sentence following the simplified sentence. Assuming again that salience based anaphora resolution algorithms perform with an accuracy of ˜0.65 on open domains, algorithm *fix_future_links* can be expected to have an error rate of around $15 \times 0.35\% = \tilde{\ }5\%$. Experimentally, using our anaphora resolution on this corpus, we report an error rate of ˜4%.

5 Conclusions and Future Work

In this paper, we have motivated the need for a regeneration component in text simplification systems by showing how naive syntactic restructuring of text can significantly disturb discourse structure. We have presented and evaluated techniques for detecting and correcting these disruptions in discourse structure. In particular, we have examined the issues of preserving the rhetorical relationships between the original clauses and phrases and preserving the text's anaphoric link structure. We believe that the techniques we have described to analyse the simplified discourse might prove useful to other NLP applications that involve transforming text; in particular, summarisation and translation.

We have tried to evaluate our algorithms intrinsically. Future work includes an extrinsic evaluation of these algorithms, using comprehension tests on subjects. This would be more useful than intrinsic evaluations in judging the benefits of text restructuring to target groups like aphasics.

Acknowledgement

This work was partially supported by the EC Fifth framework project Deep Thought IST-2001-37836.

References

Catalina Barbu and Ruslan Mitkov. 2001. Evaluation tool for rule-based anaphora resolution methods. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001), Toulouse, France*, pages 34–41.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying English text for language impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Bergen, Norway.*

Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10:183–190.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING '96)*, Copenhagen, Denmark.

Robert Dale and N. Haddock. 1991. Generating referring expressions involving relations. In *Proceedings of EACL-91, Berlin*, pages 161–166.

Siobhan Devlin. 1999. Simplifying natural language for aphasic readers. Technical report, Ph.D. thesis, University of Sunderland, UK.

Christopher Kennedy and Branimir Boguraev. 1996. Anaphora in a wider context: Tracking discourse referents. In *European Conference on Artificial Intelligence*, pages 582–586. John Wiley and Sons, Ltd, London/New York.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Towards a functional theory of text organization. *Text*, 8(3):243–281.

Judita Preiss. 2002. Choosing a parser for anaphora resolution. In *Proceedings of the 4th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2002), Lisbon, Portugal*, pages 175–180.

Ehud Reiter and Robert Dale. 1992. A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th International Conference on Computational Linguistics, Nantes, France*, pages 232–238.

Advaith Siddharthan and Ann Copestake. 2002. Generating anaphora for simplifying text. In *Proceedings of the 4th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2002), Lisbon, Portugal*, pages 199–204.

Deriving the Communicative Structure in Applied NLG

Leo Wanner
Intelligent Systems Institute
University of Stuttgart
wanner@informatik.uni-stuttgart.de

Bernd Bohnet
Intelligent Systems Institute
University of Stuttgart
bohnet@informatik.uni-stuttgart.de

Mark Giereth
Intelligent Systems Institute
University of Stuttgart
giereth@informatik.uni-stuttgart.de

Abstract

Information structure is decisive for constraining linguistic options during sentence planning. Nonetheless, it is only recently that it became a topic on the agenda of the mainstream text generation research. We investigate how certain parameters of the information (or communicative) structure developed in *Meaning-Text Linguistics* can be derived in applied text generation from the domain and discourse data, and how these parameters guide the process of sentence generation.

1 Introduction

One of the notorious problems NLG faces since its early days is the purposeful choice of one of the linguistic options available to express a given meaning. It is well known that a rich information structure constraints sentence structures, and thus, to a major extent, also the process of sentence generation (Prince, 1978; Vallduví, 1995; Choi, 1996; Mel'čuk, 2001). Existing proposals for the derivation of the information structure in the context of NLG draw mainly on contextual (extra-linguistic) information (Klabunde and Jansche, 1998; Geldorf, 2000) or on the communicative intent of the speaker (Stone et al., 2001; Creswell, 2002). Occasionally, recourse is made to semantic coherence relations (Creswell, 2002). We believe that a detailed information structure can be sufficiently determined only when the following sources are taken into account: (i) domain-specific communicative constraints (*domain communication knowledge* in (Rambow, 1990)), (ii) a detailed discourse structure as provided, e.g., by RST-based text planners, and (iii) the communicative intent of the speaker.

In what follows, we describe the derivation of the information structure in applied text generation from the above sources. As information structure, we use the *Communicative Structure* (henceforth CS) defined in the *Meaning-Text Theory* (MTT), which has the advantage of being detailed and rigorously defined; see (Mel'čuk, 2001). The derivation of the CS and its processing is currently being implemented in a text generator that is also based on MTT (Mel'čuk, 1988). The application domain under study is the ozone concentration domain in the province of Baden-Württemberg, Germany. Note, however, that the proposed approach is fully applicable to all data-oriented domains (such as stock market, flood surveyance, weather forecast, etc.).

2 Communicative Structure in MTT

Hardly any other notion in linguistics received such a heterogeneous presentation across the different theories as the information (= communicative) structure. But it cannot be our goal to present here a constrastive overview of the different interpretations. Rather, we concentrate on a brief presentation of MTT's CS as described in (Mel'čuk, 2001).

MTT's CS is defined on a semantic structure S_{sem} and consists of eight different *dimensions* or *tuples of contrastive information parameters*. Six of them call for consideration in NLG:[1]

1. Thematicity (*Rheme* vs. *Theme*),
2. Giveness (*Given* vs. *New*),
3. Focalization (*Focalized* vs. *Non-Focalized*),
4. Perspective (*Foregrounded* vs. *Backgrounded*),
5. Presupposedness (*Presupposed* vs. *Asserted*),
6. Unitariness (*Unitary* vs. *Articulated*).

[1]The seventh, *Emphasis* is immediately relevant to speech generation, and the eighth, *Locutionality* to combined gesture-language generation.

In what follows, we restrict their introduction to short definitions and a minimal number of examples; the interested reader is asked to consult (Mel'čuk, 2001). Note also that the definitions reflect the generative point of view, not the analytical one. Therefore, they do not define the parameters in terms of surface clues to be used to identify the former in a sentence, but, rather, in terms of the intentions of the Speaker.

Rheme vs. Theme. *Rheme* is that part of S_{sem} that the Speaker intends to present as being communicated. *Theme* is that part of S_{sem} that the Speaker intends to present as something about which Rheme is stated. (Theme is also sometimes referred to as *topic*, *starting point*, or *old*; Rheme—as *comment*, *focus*, and *new*.)

Depending on its POS and the interrelation with other parameters, a thematized element may be realized as the Subject of a clause, be fronted or be proleptized. The Rheme/Theme-dimension constraints thus lexicalization, syntactic choice and word order. Cf. an example:

1. $\boxed{\textit{The typical function of an interrogative clause}}_{Th}$ $\boxed{\textit{is to ask a question}}_{Rh}$
2. *In a wh-interrogative,* $\boxed{\textit{the Theme}}_{Th}$ $\boxed{\textit{consists of the wh-element.}}_{Rh}$

Given vs. New. *Given* is the part of S_{sem} that the Speaker intends to present as being in the Addressee's current consciousness or easily accessible by the Addressee. *New* is that part of S_{sem} that the Speaker intends to present as being new to the Addressee.

Since most often the Speaker assumes that the information being stated is new to the Addressee, while the information about which it is stated is present in the Addressee's consciousness, Rheme/Theme and Given/New are often conflated (this is why Theme/Rheme is sometimes called *old/new*; see above). However, it does not need to be the case that they coincide; consider, e.g.: $\boxed{\textit{A farmer from Sommerset}}_{Th/New}$ $\boxed{\textit{has found a Roman chamberpot}}_{Rh/New}$.

Given elements are usually expressed by anaphora. As suggested by Gundel (1988), MTT distinguishes four degrees of giveness: (1) unique identifiability, (2) familiarity, (3) activatedness, and (4) focality. Each of the degrees licenses primarily the choice of specific anaphoric references: (1) of the definite article, (2) of the deictic THAT, (3) of the deictic THIS, and (4) of a personal pronoun (HE, SHE, ...). That is, the Given/New dimension constraints primarily morphosyntactic options, but also lexical and syntactic ones. Cf. (from Halliday, 1994):

1. *There was* $\boxed{\textit{a little guinea pig,}}_{New}$
2. $\boxed{\text{which,}}_{Given}$ $\boxed{\text{being little,}}_{Given}$, $\boxed{\text{was not big.}}_{New}$

Focalized vs. Non-Focalized. *Focalized* is that part of S_{sem} that the Speaker intends to present as being focus of attention, i.e., logically prominent for him.

Focalization presents a configuration of entities as excluding other logical options: "exactly X, and not something else". The linguistic means to express focalization (or *focus*) include first of all *dislocation* or *detachment* and various types of *clefting*; cf.:

1. *To* $\boxed{\textit{my daughter}}_{Foc}$ *the uncle sent a doll, to his son he sent a toy car.*
2. *It was* $\boxed{\textit{the uncle}}_{Foc}$ *who sent my daughter that doll*

Foregrounded vs. Backgrounded. *Foregrounded* is that part of S_{sem} that the Speaker intends to present as being psychologically prominent for him. *Backgrounded* is that part of S_{sem} that the Speaker intends to present as being psychologicall secondary for him. Some parts of S_{sem} may be neither foregrounded nor backgrounded.

The main linguistic means for the realization of foregrounded elements is raising; for the realization of backgrounded elements—parenthetical constructions and downing; cf. (from the web):

1. *I changed my girl's oil yesterday and washed the car for* $\boxed{\textit{her}}_{Foregr}$ (in contrast to *... and washed her car,* which is neutral)
2. *The prisoner* $\boxed{\textit{(who was a skillful climber)}}_{Backgr}$ *climbed over the fence and escaped.*

Presupposed vs. Asserted. Mel'čuk (2001) distinguishes between two types of presupposition: "pragmatic" presupposition and "linguistic" presupposition, focusing on the latter one.

Linguistically presupposed is that part of S_{sem} that the Speaker intends to present as taken for granted. (If the whole structure is negated or questioned, the presupposed fragment remained affirmed.) The part that is not presupposed is *Asserted.*

Linguistically presupposed elements can be realized only as attributive (modifying or appositive) constructions; cf.:

1. *The car,* $\boxed{\textit{which was an old Renault}}_{Pres}$*, broke down soon after we left the town.*
2. Germ. $\boxed{\textit{Bei Ute}}_{Pres}$ *liegt ein Toter* $\boxed{\textit{unterm Sofa}}_{Pres}$ lit. 'At Ute's, lies a dead man under the couch'. vs. *Unter Utes Sofa liegt ein Toter* 'Under Ute's couch lies a dead man'.[2]

In German, Presupposedness also constraints word order (see below).

"Pragmatic" presupposition as used in generation, e.g., in (Stone et al., 2001), encloses all elements that are expected to be familiar to the reader (either from his world knowledge, from the context, or from the text). For generation, both types of presupposition are needed.

Unitary vs. Articulated. *Unitary* is the part of S_{sem} that the Speaker intends to present as being looked at as one (opaque) single entity. *Articulated* is the part of S_{sem} which the Speaker intends to present as being a configuration of semantic entities.

Fragments of S_{sem} that are marked as Unitary are preferably expressed by single lexemes; those that are marked as Articulated, are preferrably expressed such that each element in the semantic structure receives an own lexical item; cf.:

1. *The* $\boxed{\textit{compiler}}_{Unitary}$ *is very efficient.*
2. $\boxed{\textit{The program for compiling user written code}}_{Artic.}$ *is very efficient.*

The unitary/articulated dimension is especially important in German where regular compound production allows for a unitary realization of a broad range of configurations of semantic units.

3 The Starting Point

In this section, we introduce the resources that serve us as a basis for the derivation of the above communicative dimensions, i.e., that we presuppose as being available.[3]

[2]Only the first variant implies (presupposes) that Ute's couch is indeed at Ute's.

[3]This is not to say, of course, that we do not deal with them at all. Rather, their processing is not our topic here.

3.1 Data and Discourse Structure

We presuppose that an applied text generator starts from data stored, e.g., in a data base. In our application, these data are measuring data that are exported from a DB into an XML-document.

An "expert system" module evaluates these data, compiles a set of communicative goals that are to be achieved, and chooses the data that are relevant to these goals.

From the communicative goals, a text plan with RST-like discourse relations (Mann and Thompson, 1987) is derived. Besides RST-relations, we use Halliday's (1994) *expansion relations* ENHANCEMENT and EXTENSION and their more fine-grained variants. Our use of discourse relations differs from the use in traditional RST in two respects:
(i) specifying a discourse relation between the discourse units DU_1 and DU_2, we also specify which elements in DU_1 and DU_2 are involved in the relation; thus, the CONTRAST-relation between (a) and (b) in

(a) *It was John who sent my daughter the doll.* (b) *Mary never sends her anything*

the "hubs" of the nuclei are *John* and *Mary*, respectively (not, e.g., *doll* and *nothing*).
(ii) several relations may hold between DU_1 and DU_2 (see also (Moore and Pollack, 1992) on the need for multi-level analysis of RST-relations).

As mentioned above, we presuppose that a text plan has already been compiled when we start the compilation of the CS.

3.2 Domain Communication Data

Originally defined for the semantic level, MTT's communicative dimensions can also be used at the conceptual, i.e. "prelinguistic", level. For some of them, initial settings are already available before generation starts. They are predetermined by the domain, by the design of the interface via which the reader communicates with the generator, and by the actions the user takes during the session.

Data on Thematicity In applied generation, the global theme of the discourse (i.e., the discourse topic) is either known—if the generator is specialized on one text type—, or the reader determines it by choosing a specific topic via the generator interface. The theme and rheme of the first message in the text are either directly related to the discourse theme or can be derived from actions of the reader.

In our application, the discourse theme is determined by the goal-directed action of going to the web page of the generator: *Ozone in the province of Baden-Württemberg, Germany*. The initial discourse theme is thus 'ozone'. The page contains a map of Baden-Württemberg with stations that measure ozone being marked by a dot. By clicking onto a station on the map, the user determines the name of the station as the secondary theme. The information on the current concentration at the station in question constitutes the corresponding secondary rheme.

Note that on his first visit to the page, the reader has no explicit information that the texts will be about ozone CONCENTRATION. Therefore, discourse theme is 'ozone' only. After the first text has been generated, the discourse theme is extended to 'ozone concentration' for all subsequent messages.

Data on Giveness Some of the information units can be considered as given to (or known by) the reader before any text is generated or even planned; some others as unknown or new. Cf. Table 1 for the distribution of the given/new-parameter in our domain for the most important data:

Table 1: Giveness of entities

Entity	given	new
substance (ozone)	x	
values (concentrations)		x
measuring unit ($\mu g/m^3$)	x	
times (measured at)		x
locations (measuring stations)	x	
names of applicable thresholds	x	
values of applicable thresholds		x

Data on Focalization An entity e is a candidate for focalization, e.g., if:

– a specific (prominent) property or event can be assigned to several entities, and it is assigned to e,

– e belongs to the global discourse rheme or to a preceding local initial rheme.

Due to these conditions, in our domain, e.g., current ozone concentration ($O_3^{t_{curr}}$) can be focalized if it is either the highest or the lowest concentration in the region; also, t_{ref} (= the time whose concentration is contrasted to the current concentration). Air quality experts suggest that it is not adequate to focalize any information in the first message of the text.

Data on Perspective In an informative discourse, data that are in one way or the other unusual or are supposed to somehow influence the reader can be foregrounded, i.e., "marked as being psychologically primary". Data or their evaluation that are "normal" from the perspective of the reader may be backgrounded, i.e., "marked as psychologically secondary". Backgrounded may be also data that are the premise of an evaluation of data that is foregrounded (as, e.g., in *Um 18 Uhr hat die Ozonkonzentration in Stuttgart mit 217 $\mu g/m^3$ den höchsten Wert des Tages erreicht* the *mit*-Konstruktion backgrounds the actual concentration).

In our domain, sentence constructions with foregrounded elements have been judged "too dramatic". Consider (b) in the following example:

(a) *An der Messstation Esslingen wurde um 18 Uhr eine Ozonkonzentration von 217 $\mu g/m^3$ gemessen* 'At the measuring station Esslingen, at 18 o'clock, an ozone concentration of 217 $\mu g/m^3$ has been measured' ... (b) *Diese Konzentration war zu der Zeit in der Region Mittlerer Neckar der Höchstwert* 'This concentration was at this time the highest-value in the region Mittlerer Neckar.'

The modifier *höchste* in *höchste Konzentration* is raised to the sentential complement to build the compound *Höchstwert* and thus focalized (recall that the main syntactic means to realize foregrounded elements is raising). A more appropriate variant contains no foregrounded elements; cf.:

(a) ... (b) *Dieser Wert war die höchste zu dieser Zeit in der Region Mittlerer Neckar gemessene Konzentration.* 'This value was the highest concentration measured at this time in the region Mittlerer Neckar'.

Therefore, we use the 'background' parameter only.

Data on Presupposedness Certain elements of the discourse that are very prominent in the reader's mind can and should be omitted—either from the start or after their first mention. That is, they are pragmatically presupposed. In our domain, this is the name of the measuring station for which the reader asked for information.

In our domain, we can further identify some linguistically presupposed elements before generation starts: with the user's action of clicking on a measuring station, we can presuppose (i) the concept of 'ozone concentration', the location (i.e. measuring station), and the time at which the measure has taken place. The ozone concentration is asserted.

Note that the presuposedness of time prevents the shift of the time circumstantial to the final position in the clause (which is *per se* allowed in German):[4]

#*An der Messstation Heilbronn lag die Ozonkonzentration bei 182 $\mu g/m^3$ um 18 Uhr* lit. 'At the measuring station Heilbronn, the ozone concentration was at 182 $\mu g/m^3$ at 18 o'clock'.

Data on Unitariness. Often, a domain prescribes a unitary or an articulated realization of specific information elements. Thus, articles on a computer science issue written for professionals would hardly use *program for compiling user written code* to refer to a compiler, while in a paper for laymen, it would make sense to introduce the term "compiler" by an articulated lexicalization of the concept.

In our domain, the following information units are unitary by definition: (1) location + name, (2) time + time instance, (3) substance + 'concentration'.

4 Deriving the Communicative Structure

With the initial domain data, domain communication data, and the text plan at hand, the instantiation of the communicative dimensions can be derived for each message to be generated. In this section, we illustrate how the parameters for the first four dimensions from above can be dynamically determined by a set of communicative rules. The parameters of the other two dimensions are determined analogously.

To facilitate the presentation, let us first introduce some notations and conventions: (1) $\mathcal{M}$ stands for 'message under construction', and $\mathcal{M}^-$, for 'one of the preceding messages' (an index may be additionally given if more than one of the preceding messages is considered). (2) $DU_{\mathcal{M}^-}$ and $DU_{\mathcal{M}}$ stand for 'discourse unit containing message $\mathcal{M}^-$ or $\mathcal{M}$, respectively'. (3) Pairs of the type 'ozone concentration–Y $\mu g/m^3$', 'age–4 months', 'time–5pm', etc. will be referred to as 'token–value' (t–v).

[4] '#' marks communicatively inadequate utterances.

Theme/Rheme. To determine Theme and Rheme of the message in question, we draw upon all of the above types of data. Consider examples for the use of each. The use of the domain communication data is most obvious:

If $\mathcal{M}$ is the first to be generated then

$$Th_{\mathcal{M}} := Th_{discourse} \cup Th_{init,secondary}$$
$$Rh_{\mathcal{M}} := Rh_{init,secondary}$$

Discourse relations are often decisive when the thematic structure of one of the subsequent messages is determined. Consider:

(a) *An der Station Stuttgart wurden um 18 Uhr 180 $\mu g/m^3$ Ozon gemessen* 'At the station Stuttgart at 18 o'clock, 180 $\mu g/m^3$ have been measured'. (b) *Um 17 Uhr lag der Wert noch bei 120 $\mu g/m^3$* 'At 17 o'clock, the value still was at 120 $\mu g/m^3$'.

(a) *Sven Hannawald sprang in Bischofshofen 132.5m weit* lit. 'Sven Hannawald jumped in Bischofshofen 132.5m far.' (b) *In Garmisch waren es nur noch 124m.* lit. 'In Garmisch, they were only just 124m'.

In both examples, between (a) and (b) a CONTRAST relation holds; more precisely, between the values of a token (in the first, the token is 'ozone concentration', in the second, 'length') with respect to a circumstantial ('time' in the first and 'location' in the second). In both (a)s, the token belongs to Theme and value and the circumstantial to Rheme. In both (b)s, the token and the circumstantial are Theme, and the value is Rheme. This is a regular pattern. We can thus formulate the following rule:

If between $DU_{\mathcal{M}^-}$ and $DU_{\mathcal{M}}$ a CONTRAST-relation holds and
1. it contrasts the values $v^- \in \mathcal{M}^-$ and $v \in \mathcal{M}$ of the token t with respect to the circumstantial c,
2. $t \in Th_{\mathcal{M}^-}$, $v^- \in Rh_{\mathcal{M}^-}$, $c \in Rh_{\mathcal{M}^-}$
Then, $Th_{\mathcal{M}} \leftarrow t$; $Th_{\mathcal{M}} \leftarrow c$; $Rh_{\mathcal{M}} \leftarrow v$

Consider now

(a) *Die Ozonkonzentration lag um 18 Uhr bei 198 $\mu g/m^3$* 'The ozone concentration was at 18h at 198 $\mu g/m^3$'. (b) *Das war der höchste Wert in der Region Mittlerer Neckar* 'This was the highest value in the region Mittlerer Neckar'.

(a) *Sven Hannawald sprang in Bischofshofen 132.5m weit* lit. 'Sven Hannawald jumped in Bischofshofen 132.5m far.'

(b) *Das war der weiteste Sprung des Tages* 'This was the longest jump of the day'.

Between the (a) and (b) the EVALUATION relation holds. Again, we can detect a stable theme pattern: the value of a token introduced, i.e. rhematized, in (a), is evaluated in (b) and is thus part of theme in (b). Cf. the corresponding rule:

> If between $DU_{\mathcal{M}^-}$ and $DU_{\mathcal{M}}$ an EVALUATION-relation holds and
> 1. the value $v^- \in \mathcal{M}^-$ of a token $t \in \mathcal{M}^-$ is evaluated by the entity e in $\mathcal{M}$,
> 2. $t \in Th_{\mathcal{M}^-}$ and $v^- \in Rh_{\mathcal{M}^-}$
>
> Then, $Th_{\mathcal{M}} \leftarrow v^{-1}$; $Rh_{\mathcal{M}} \leftarrow e$.

Other relations, such as ELABORATION, ENHANCEMENT, and JUSTIFICATION are equally used for the derivation of thematic patterns.

The use of factual domain data along with discourse relations can be illustrated by the following example:

(a) *217 $\mu g/m^3$ ist relativ viel* lit. '217 $\mu g/m^3$ is relatively much', (b) *wenn auch der Alarmschwellenwert von 240 $\mu g/m^3$ noch nicht erreicht ist* 'although the alarm threshold of 240 $\mu g/m^3$ has not yet been reached'.

Here, between (a) and (b) a CONCESSION relation holds. In (a), *217 $\mu g/m^3$* is the Theme, in (b) *der Alarmschwellenwert von 240 $\mu g/m^3$*. That is, we have the general pattern 'X is Y, but not yet Z'. This pattern is captured by the following rule:

> If between $DU_{\mathcal{M}^-}$ and $DU_{\mathcal{M}}$ a CONCESSION-relation holds and
> 1. $\mathcal{M}^-$ contains an attribute assignment 'v^- IS a',
> 2. $\mathcal{M}$ is a statement that $v^- <$ threshold τ,
> 3. $v^- \in Th_{\mathcal{M}}$ and $v^- >$ threshold α
>
> Then, $Th_{\mathcal{M}} \leftarrow \tau$; $Rh_{\mathcal{M}} \leftarrow \tau > v^-$

Given/New. The task of the Given/New-rules is on the one hand to change the giveness status of entities that have been mentioned in the current message for the first time from 'New' to 'Given' and, on the other hand, to assign a giveness degree to 'Given' entities.

To all entities that are marked as 'Given' in the initial given/new-table, we assign the giveness degree 1.

The degree of giveness of an entity with respect to $\mathcal{M}$ (i.e. the message planned) depends on the distance of this entity from $\mathcal{M}$ (measured in number of words or messages). This is well-known from the approaches to the generation of referring expressions (Dale and Reiter, 1995; Horacek, 1995). Which degree is assigned to the entities immediately at $\mathcal{M}$ and how quickly (or whether) the degree is decremented with the increasing distance depends on the domain and on the nature of each individual entity. In our domain, only two degrees are used: 1 and 4. The concept 'ozone concentration' is assigned the degree 4 at the point of its mention; at message distance 2, the degree is set to 1. All other given entities receive a constant degree 1.

Degree 4 licenses the use of a personal pronoun and the deictic pronoun DIESE(R) 'this', and degree 1 licenses the use of the definite article.

Focalized. Criteria for focalization tend to be more idiosyncratic than the criteria for thematization. Nonetheless, since focalization usually happens in context, focalization rules draw on both discourse relations and ideational domain data.

A typical focalization is illustrated by the following discourses:

(a) An der Messstation Heilbronn wurden heute 86 $\mu g/m^3$ gemessen lit. 'At the measuring station Heilbronn, today 86 $\mu g/m^3$ have been measured' ...(b) *86 $\mu g/m^3$, das war der niedrigste Wert der Region Mittlerer Neckar* '86 $\mu g/m^3$, this was the lowest value of the Mittlerer Neckar region'.

(a) *Hannawald sprang heute 132m* 'Hannawald jumped today 132m' ...(b) *132m, das war der längste Sprung des Tages* '132m, this was the longest jump of the day'.

Between (a) and (b), which appear in the discourse at a certain distance from each other, an EVALUATION-relation holds: (b) evaluates (a)'s rheme v^-. The evaluation in (b) consists of the statement 'v^- is highest/lowest in the given range'. The following rule captures this pattern:

> If between $DU_{\mathcal{M}^-}$ and $DU_{\mathcal{M}}$ an EVALUATION-relation holds and
> 1. the value $v^- \in \mathcal{M}^-$ of a token $t \in \mathcal{M}^-$ is evaluated by the entity e in $\mathcal{M}$,
> 2. $v^- \in Rh_{\mathcal{M}^-}$;
> 3. e states that v^- is the highest/lowest in a given range
>
> Then, $Focus_{\mathcal{M}} \leftarrow v^-$.

Similar rules can be defined for such cases as illustrated by the following examples (focalized entities are underlined):

(a) *An der Messstation Heilbronn wurden um 17 Uhr 47 $\mu g/m^3$ gemessen* 'At the measuring station Heilbronn, at 17 o'clock, 47 $\mu g/m^3$ have been measured'. (b) *Gegenüber <u>16 Uhr</u>, als der Wert bei 20 $\mu g/m^3$ lag, hat sich also die*

Konzentration mehr als verdoppelt. lit. 'Compared to 16 o'clock, when the value was about 20 $\mu g/m^3$, the concentration thus more than doubled'.

where a CONTRAST relation holds between (a) and (b).

(a) ...(b) *Was die* *anderen Messstationen der Region Mittlerer Neckar betrifft, so lagen dort die Werte zwischen 51* $\mu g/m^3$ *in Esslingen und 67* $\mu g/m^3$ *in Plochingen* 'As far as the other stations in the region Mittlerer Neckar are concerned, the values there were between 51 $\mu g/m^3$ in Esslingen and 67 $\mu g/m^3$ in Plochingen',

where an ENHANCEMENT-relation holds between (a) and (b).

Backgrounded. As mentioned above, in accordance with the characteristics of our domain, we do not mark any information as 'Foregrounded'. We background only in the case of the following pattern:

If between $DU_{\mathcal{M}^-}$ and $DU_{\mathcal{M}}$ an EVALUATION-relation holds and
1. the value $v^- \in \mathcal{M}^-$ of a token $t \in \mathcal{M}^-$ is evaluated by the entity e in $\mathcal{M}$,
2. v^- is unusually high,
3. $v^- \in Rh_{\mathcal{M}^-}$,
4. e states that v^- is the highest in a given range
Then $Background_{\mathcal{M}^-} \leftarrow v^-$.

(compare the similarity with the focalization pattern above).

The backgrounded element is "downed" such that $\mathcal{M}^-$ and $\mathcal{M}$ are realized in one clause; cf.:

An der Messstation Heilbronn wurde um 18 Uhr mit 198 $\mu g/m^3$ *der höchste Wert des Tages erreicht* lit. 'At the measuring station Heilbronn, at 18 o'clock with 198 $\mu g/m^3$ the highest value of the day has been reached.

(*198* $\mu g/m^3$ is downed by the use of a *mit* 'with'-PP).

5 Processing Comm. Structure

MTT is a multistratal theory. The most abstract stratum (or level) we use is the conceptual stratum; the most concrete stratum that is relevant for generation is the surface-morphological level, which can be considered as a chain of inflected wordforms. The representations at each level can, somewhat simplified, be assumed as consisting of two structures: the basic (propositional) structure and the CS, which is defined on the basic structure and thus partitions the basic structure in terms of communicative dimensions.

Generation in the sense of MTT consists of a series of mappings between representations of adjacent levels, starting from the conceptual representation that is annotated with communicative dimensions and going up until the surface-morphological representation is reached; for an implementation, see (Bohnet and Wanner, 2001).

In Section 2, we have already indicated the linguistic means by which the individual communicative parameters are realized. During the transition from level E_i to level E_{i+1}, a communicative parameter is either realized by the appropriate linguistic means available at E_{i+1} or is mapped onto the CS of E_{i+1}, i.e., propagated to E_{i+1} in order to be realized on one of the higher levels. Both the realization and the propagation are specified in terms of communicative rules, which make part of the grammar rules. The communicative rules are discussed at length in a longer version of this paper.

6 Related Work

Among the first to apply the information structure to text generation were C. Matthiessen (1985), K. McKeown (1985), and L. Iordanskaja (1992). Especially Iordanskaja discusses in detail how Thematization influences the order of the messages in a text plan, and, to a certain extent, also aggregation.

More recently, Humphreys (1995) investigated how the speaker's (communicative) intentions guide the choice of such "non-canonical" sentence patterns in English as clefting and dislocation (which we considered as realizations of *focalized* elements). As Humphreys, Stone *et al.* (2001) relate in the SPUD-system sentence planning options to communicative intentions of the speaker, which are in their case captured by *Assertion*, *Presupposition* and *Pragmatics* (while Humphreys develops explicit speaker and hearer models). Note that Assertion and Presupposition in SPUD are "pragmatic notions" (see Section 2). Creswell (2002) extends Stone *et al.*'s approach by three types of more fine-grained communicative goals: attention marking, discourse relation, and focus disambiguation. As examples of discourse relations Creswell cites NARRATIVE and PARALLEL. However, it is not clear how many and which discourse relations are covered. But, obviously, Creswell's proposal is similar to ours.

7 Summary and Future Work

We presented how discourse structure relations and domain communication data can be used to compile a CS, which guides then sentence planning and realization. The described model is an extension of the model underlying the *AutoText UIS* generator, which has been developed in cooperation with the Ministry of Environment and Traffic, Baden-Württemberg (Bohnet and *et al.*, 2001) and which is in action since summer 2001. The extended model is currently under implementation. However, it still reveals several limitations. Thus, we work so far with a subset of RST-like relations in the air quality domain restricted to ozone. It is planned to extend the generation to other substances in this domain—which implies a broader coverage of discourse relations, and thus, also a broader coverage of the interrelation between discourse relations and communicative dimensions. However, the air quality domain alone is certainly still too restricted for a full scale coverage of the phenomena related to CS. Therefore, we plan to examine two other application domains: flood surveyance and weather forecast. In parallel, we continue to work on the extension of our sentence grammar module.

References

B. Bohnet, L. Wanner *et al.* 2001. Autotext-UIS: Automatische Produktion von Ozonkurzberichten im Umweltinformationssystem Baden-Württemberg. In *Proceedings of the Workshop Hypermedia und Umweltschutz*, Ulm.

B. Bohnet and L. Wanner. 2001. On Using a Parallel Graph Rewriting Grammar Formalism in Generation. In *Proceedings of the 8th European Workshop on NLG at the ACL*, Toulouse, France.

H.-W. Choi. 1996. *Optimizing Structure in Context: Scrambling and Information Structure*. Ph.D. thesis, Stanford University, Stanford.

C. Creswell. 2002. Syntactic Form and Discourse Function in NLG. In *Proceedings of the 2nd International Conference on NLG, Student Session*.

R. Dale and E. Reiter. 1995. Computational Interpretation of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science*, 19(2):233–263.

S. Geldorf. 2000. From Context to Sentence Form. In *Proceedings of the 1st International Conference on NLG, Student Session*.

J.K. Gundel. 1988. "Universals of Topic-Comment Structure". In M. Hammond, E. Moravčik, and J. Wirth, editors, *Studies in Syntactic Typology*, pages 209–239. Benjamins, Amsterdam.

M.A.K. Halliday. 1994. *An Introduction to Functional Grammar*. Edward Arnold, London.

H. Horacek. 1995. More on Generating Referring Expressions. In *Proceedings of the 5th European Workshop on NLG*, pages 43–58.

K. Humphreys. 1995. *Formalising Pragmatic Information for Natural Language Generation*. Ph.D. thesis, University of Edinburgh.

L. Iordanskaja. 1992. "Communicative Structure and its Use during Text Generation". *Int. Forum on Information and Documentation*, 17(2):15–27.

R. Klabunde and M. Jansche. 1998. Abductive Reasoning for Syntactic Realization. In *Proceedings of the International Workshop NLG*, Niagara-on-the-Lake, ON, Canada.

W.C. Mann and S.A. Thompson. 1987. Rhetorical Structure Theory: A theory of text organization. In L. Polanyi, editor, *The Structure of Discourse*. Ablex, Norwood, New Jersey.

C.M.I.M. Matthiessen. 1985. The Systemic Framework in Text Generation: Nigel. In J.D. Benson and W.S. Greaves, editors, *Systemic Perspectives on Discourse, Volume 1*. Ablex, Norwood, New Jersey.

K. McKeown. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. CUP, Cambridge.

I.A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. SUNY Press, Albany.

I. A. Mel'čuk. 2001. *Communicative Organization in Natural Language (The Semantic-Communicative Structure of Sentences)*. Benjamins, Amsterdam.

J. Moore and M. Pollack. 1992. A problem for RST: The Need for Multi-Level Discourse Analysis. *Computational Linguistics*, 18(4):537–544.

E. Prince. 1978. A Comparison of WH-Clefts and IT-Clefts in Discourse. *Language*, 54(4):883–906.

O. Rambow. 1990. Communication Domain Knowledge. In *Proceedings of the 5th. International Workshop on NLG.*, Dawson, PA.

M. Stone, C. Doran, B. Webber, T. Bleam, and M. Palmer. 2001. Microplanning with Communicative Intentions: The SPUD system. Rutgers.

E. Vallduví. 1995. Information packaging: A survey. Report, Center for Cognitive Science and HCRC, University of Edinburgh.

Adapting Chart Realization to CCG

Michael White and Jason Baldridge
School of Informatics
University of Edinburgh
{mwhite,jmb}@inf.ed.ac.uk

Abstract

We describe a bottom-up chart realization algorithm adapted for use with Combinatory Categorial Grammar (CCG), and show how it can be used to efficiently realize a wide range of coordination phenomena, including argument cluster coordination and gapping. The algorithm has been implemented as an extension to the OpenNLP open source CCG parser. As an avenue for future exploration, we also suggest how the realizer could be used to simplify the treatment of aggregation in conjunction with higher level content planning components.

1 Introduction

In this paper, we describe our initial efforts to develop a practical, open source realizer for Combinatory Categorial Grammar (CCG, Steedman (2000b)). While CCG provides theoretically attractive accounts of numerous linguistic phenomena — including unique accounts of coordination and intonation, which is of particular interest to builders of dialog systems[1] — its adoption by the NLG community has been hindered by the lack of a practical realizer. As a first step towards making such a realizer available, we have implemented a bottom-up chart realization algorithm (Carroll et al., 1999) adapted for use with CCG. The implementation builds upon the Java-based OpenNLP CCG parser[2] described in Baldridge (2002).

The paper is organized as follows. We provide the rationale for our algorithm choice in §2. In §3 and §4, we provide background for the realization algorithm and the algorithm itself. In §5, we show how the realizer handles a wide range of coordination phenomena. In §6, we provide initial evidence that the realizer can be reasonably efficient in practice. In §7, we discuss related work and conclude with a discussion of future directions.

2 Rationale

Since our chart realization algorithm may not be the most efficient algorithm one might consider implementing, we provide the following rationale for our choice:

Completeness The simple nature of our algorithm makes it relatively straightforward to achieve completeness;[3] in contrast, it would be more difficult to do so for all combinatory rules with Hoffman's (1995) adaptation of semantic head driven realization for CCG.

Parser reuse Since the algorithm is entirely bottom-up, it can directly reuse the parsing-oriented optimizations of the CCG rules described in Baldridge (2002).

[1]We are primarily targeting the realizer for use in dialog systems, and intend to use it in the IST project COMIC (COnversational Multimodal Interaction with Computers), `http://www.mpi.nl/comic/`.

[2]`http://opennlp.sourceforge.net/`

[3]That is, to ensure that all derivations licensed by the grammar can be reversed.

LF order independence The algorithm does not rely on the order of conjuncts in the input logical form, and thus handles this oft-discussed aspect of the logical form equivalence problem (Shieber, 1993).

Anytime search The use of an agenda makes it easy to control the search for possible realizations, and thus to run the algorithm in anytime mode.[4]

3 Background

3.1 Combinatory Categorial Grammar

We provide here a brief overview of CCG; see Steedman (2000b) for an extensive introduction.

A given CCG grammar is defined almost entirely in terms of the entries of the lexicon, which are (possibly complex) categories bearing standard feature information (such as tense, agreement, etc.) and subcategorization information. Some (simplified) lexical entries are given below:

(1) a. *man* $\vdash$ n
    b. *that* $\vdash$ $(\mathsf{n}\backslash\mathsf{n})/(\mathsf{s}_{vform=fin}/\mathsf{np})$
    c. *Bob* $\vdash$ np
    d. *saw* $\vdash$ $(\mathsf{s}_{tense=past,vform=fin}\backslash\mathsf{np})/\mathsf{np}$

CCG has a small set of rules which can be used to combine categories in derivations. The two most basic rules are forward (>) and backward (<) function application:

$$(>)\quad X/Y\ \ Y \ \Rightarrow\ X$$
$$(<)\quad Y\ \ X\backslash Y \ \Rightarrow\ X$$

CCG also employs further rules based on the composition (**B**), type-raising (**T**), and substitution (**S**) combinators of combinatory logic. Each combinator gives rise to several directionally-distinct rules; for example, there are forward and backward rules for both composition and type-raising:

$$(>\mathbf{B})\quad X/Y\ \ Y/Z \ \Rightarrow\ X/Z$$
$$(<\mathbf{B})\quad Y\backslash Z\ \ X\backslash Y \ \Rightarrow\ X\backslash Z$$
$$(>\mathbf{T})\quad X \ \Rightarrow\ Y/(Y\backslash X)$$
$$(<\mathbf{T})\quad X \ \Rightarrow\ Y\backslash(Y/X)$$

These rules are crucial for building the "non-standard" constituents for which CCG is well-known, and which are essential for CCG's handling of coordination, extraction, intonation, and other phenomena. For example, CCG's rules and the categories given in (1) lead to the following derivation of the relative clause *man that Bob saw*:

```
(2)  man      that        Bob       saw
     ---  -----------   -------  -----------
      n    (n\n)/(s/np)    np     (s\np)/np
                         ------->T
                         s/(s\np)
                         ------------------->B
                                s/np
          ---------------------------------->
                        n\n
     -------------------------------<
                  n
```

The OpenNLP CCG system uses a multi-modal version of CCG (Baldridge, 2002; Baldridge and Kruijff, 2003), which has a fully universal rule component that makes it possible to write more efficient unification schemes for rule application than for the original CCG framework.

3.2 Hybrid Logic Dependency Semantics

Like other compositional grammatical frameworks, CCG allows logical forms to be built in parallel with the derivational process. Traditionally, the λ-calculus has been used to express semantic interpretations, but work in other frameworks has moved to using more flexible representations in computational implementations, such as the MRS framework (Copestake et al., 2001) used for HPSG. In the context of categorial grammar, Kruijff (2001) proposes a framework that utilizes hybrid logic (Blackburn, 2000) to realize a dependency-based perspective on meaning. Baldridge and Kruijff (2002) show how this framework, Hybrid Logic Dependency Semantics (HLDS), relates closely to MRS, and show how terms of HLDS can be built compositionally with CCG via unification. In the next section, we show how HLDS's flexibility enables an approach to semantic construction that ensures semantic monotonicity, simplifies equality tests, and avoids copying in coordinate constructions.

Hybrid logic provides a language for representing relational structures that overcomes standard modal logic's inability to directly reference states in a model. It does so by using *nominals*, a new sort of basic formula with which we can explicitly name states. In addition to propositions, nominals are first-class citizens of the object language: formulas can be formed using propositions, nominals, standard boolean operators, and the *satisfaction operator* "@". A formula $@_i(p \wedge \langle \mathrm{F} \rangle (j \wedge q))$

[4]That is, to allow the client program to request the best solution found so far at any time. We are currently exploring strategies for ranking partial solutions based on n-gram measures (Varges, 2001).

indicates that the formulas p and $\langle \mathrm{F} \rangle (j \wedge q)$ hold at the state named by i and that the state j is reachable via the modal relation F.

In HLDS, hybrid logic is used as a language for describing discourse representation structures — which have their own underlying semantics — as follows. Each semantic head is associated with a nominal that identifies its *discourse referent*, and heads are connected to their dependents via dependency relations, which are modeled as modal relations. As an example, the sentence *Bob saw Gil* receives the represention in (3).

(3) $@_e(\mathbf{see} \wedge \langle \textsc{tense} \rangle \mathbf{past} \wedge \langle \textsc{Act} \rangle (b \wedge \mathbf{Bob}) \wedge \langle \textsc{Pat} \rangle (g \wedge \mathbf{Gil}))$

In this example, e is a nominal that labels the predications and relations for the head **see**, and b and g label those for the the **Bob** and **Gil**, respectively. The relations ACT and PAT represent the dependency roles **Actor** and **Patient**, respectively.[5]

By using the @ operator, hierarchical terms such as (3) can be flattened to an equivalent conjunction of fixed-size *elementary predications* (EPs), closely related to MRS terms:

(4) $@_e\mathbf{see} \wedge @_e\langle \textsc{tense} \rangle \mathbf{past} \wedge @_e\langle \textsc{Act} \rangle b \wedge @_e\langle \textsc{Pat} \rangle g \wedge @_b\mathbf{Bob} \wedge @_g\mathbf{Gil}$

As (4) shows, EPs come in three varieties: lexical predications, (e.g. $@_e\mathbf{see}$); semantic features (e.g. $@_e\langle \textsc{tense} \rangle \mathbf{past}$); and relations, (e.g. $@_e\langle \textsc{Act} \rangle b$).

3.3 Semantic Construction

To facilitate realization from HLDS terms, we have slightly changed Baldridge and Kruijff's (2002) approach to semantic construction to one which uses maximally flat representations such as (4). In our revised approach, EPs are paired with syntactic categories in the lexicon to form signs, as shown in (5)–(7) below. Each atomic category has an index feature which makes a nominal available for capturing syntactically induced dependencies; these indices are shown as subscripts on the category labels.

(5) *saw* $\vdash (\mathsf{s}_e\backslash \mathsf{np}_x)/\mathsf{np}_y$: $@_e\mathbf{see} \wedge @_e\langle \textsc{tense} \rangle \mathbf{past} \wedge @_e\langle \textsc{Act} \rangle x \wedge @_e\langle \textsc{Pat} \rangle y$

(6) *Bob* $\vdash \mathsf{np}_b : @_b\mathbf{Bob}$

(7) *Gil* $\vdash \mathsf{np}_g : @_g\mathbf{Gil}$

[5]To refrain from committing to a particular set of dependency roles, relations such as ARG1, ARG2, etc. can be used.

In derivations, applications of the combinatory rules coindex the appropriate nominals via unification on the categories, and the EPs are then conjoined to form the resulting interpretation. For example, (6) can type-raise and compose with (5) to yield (8), where x has been coindexed with b, and where the EPs have been conjoined; (8) can then apply to (7) to yield (9), which has the same conjunction of predications as (4).[6]

(8) *Bob saw* $\vdash \mathsf{s}_e/\mathsf{np}_y$: $@_e\mathbf{see} \wedge @_e\langle \textsc{tense} \rangle \mathbf{past} \wedge @_e\langle \textsc{Act} \rangle b \wedge @_e\langle \textsc{Pat} \rangle y \wedge @_b\mathbf{Bob}$

(9) *Bob saw Gil* $\vdash \mathsf{s}_e$: $@_e\mathbf{see} \wedge @_e\langle \textsc{tense} \rangle \mathbf{past} \wedge @_e\langle \textsc{Act} \rangle b \wedge @_e\langle \textsc{Pat} \rangle g \wedge @_b\mathbf{Bob} \wedge @_g\mathbf{Gil}$

Since the EPs are always conjoined by the combinatory rules, semantic construction is guaranteed to be *monotonic* — in the sense that no semantic information can be dropped during the course of a derivation — which is an essential property for ensuring that the realization algorithm is complete (Copestake et al., 2001).

Another benefit of this approach to semantic construction is that it becomes easier to perform equality tests on signs, since the flat conjunctions of EPs can be sorted into a canonical order and compared in turn. Such equality tests can be used to avoid adding duplicate entries into the chart when there are multiple equivalent derivations for a given sign, thereby alleviating the problem of so-called "spurious" ambiguity (Steedman, 2000b).

A final benefit of simply conjoining EPs in derivations is that it avoids any copying of predications in coordinate constructions. In contrast, the approach implicit in Baldridge and Kruijff (2002) yields duplicate predications in examples such as *Bob heard and Ted saw Gil*, where the proposition **Gil** appears twice (ignoring tense):

(10) $@_{e_1}(\mathbf{hear} \wedge \langle \textsc{Act} \rangle (b \wedge \mathbf{Bob}) \wedge \langle \textsc{Pat} \rangle (g \wedge \mathbf{Gil}) \wedge \langle \textsc{Coord} \rangle (e_2 \wedge \mathbf{see} \wedge \langle \textsc{Act} \rangle (t \wedge \mathbf{Ted}) \wedge \langle \textsc{Pat} \rangle (g \wedge \mathbf{Gil})))$

As we will show in §5, by avoiding such duplicate predications, the present approach to semantic construction keeps the output of the parser in line with the expected input of the realizer.

[6]There is another more traditional (but less incremental) derivation for *Bob saw Gil*, where *saw* combines first with the object *Gil* before combining with the subject *Bob*.

4 The Algorithm

4.1 Data Structures

The input to the algorithm is a logical form encoded as an HLDS term. The input term is flattened to a list of EPs, so that the extent to which partial realizations cover the input LF can be tracked positionally. For example, to realize *man that Bob saw*, the hierarchically structured input in (11) is flattened to (12):

(11) $@_x(\textbf{man} \wedge \langle \text{GenRel} \rangle (e \wedge \textbf{see} \wedge \langle \text{tense} \rangle \textbf{past} \wedge \langle \text{Act} \rangle (b \wedge \textbf{Bob}) \wedge \langle \text{Pat} \rangle x))$

(12) $0 : @_x\textbf{man},\ 1 : @_x\langle \text{GenRel} \rangle e,\ 2 : @_e\textbf{see}$
$3 : @_e\langle \text{tense} \rangle \textbf{past},\ 4 : @_e\langle \text{Act} \rangle b$
$5 : @_e\langle \text{Pat} \rangle x,\ 6 : @_b\textbf{Bob}$

The algorithm makes use of three principal data structures: edges, an agenda and a chart. An *edge* is just a CCG sign plus a couple of bit vectors which record the sign's coverage of the input LF and the sign's indices (nominals) that are syntactically available. These bit vectors make it possible to instantly check whether two edges cover disjoint parts of the input LF and whether they have any indices in common. For example, the edges for the finite past and non-finite forms of *see* are given below, with the bit vectors for the EPs and indices shown in braces:

(13) $\{2, 3, 4, 5\}\ \{e, b, x\}$
saw $\vdash (\mathsf{s}_{e,fin} \backslash \mathsf{np}_b)/\mathsf{np}_x :$
$@_e\textbf{see} \wedge @_e\langle \text{tense} \rangle \textbf{past} \wedge @_e\langle \text{Act} \rangle b \wedge @_e\langle \text{Pat} \rangle x$

(14) $\{2, 4, 5\}\ \{e, b, x\}$
see $\vdash (\mathsf{s}_{e,nonfin} \backslash \mathsf{np}_b)/\mathsf{np}_x :$
$@_e\textbf{see} \wedge @_e\langle \text{Act} \rangle b \wedge @_e\langle \text{Pat} \rangle x$

The *agenda* is a priority queue of edges which manages the edges that have yet to be added to the chart. Using the agenda makes it easy to vary the search order by changing the edge sorting strategy.

The *chart* is a collection of edges that enables a dynamic programming search for realizations. Whereas a chart for parsing uses string positions to track partial parses, one for realization uses an edge's coverage vector to track partial realizations.

4.2 Lexical Lookup

In the first phase of the algorithm, for each EP in the flattened input LF, relevant lexical entries are accessed according to the following indexing scheme. Most lexical items are indexed by the principal lexical predicate which they introduce. However, if a lexical item (e.g. a relative pronoun) only introduces a dependency relation or a semantic feature, it is indexed by the relation or feature. Semantically null lexical items, i.e. ones which introduce no EPs (e.g. infinitival *to*), are not indexed at all; instead, they receive special handling in the combinatory rule phase (see step 4 in figure 1). Case marking prepositions and particles are only considered when there is a matching feature on one of the indexed lexical items indicating that they may be needed.

Once a lexical entry indexed by the current EP has been accessed, instantiation is attempted. During instantiation, the current EP is unified first, and then unification of the remaining EPs in the lexical entry is attempted against the remaining EPs in the input LF. The lexical entry is allowed to introduce extra semantic features, enabling some limited underspecification in the input LF.

For example, the predicational EP $@_e\textbf{see}$ triggers the lookup of the edges shown in (13) and (14). Note that the present tense form *sees* is accessed as well, but instantiation fails due to its incompatible $\langle \text{tense} \rangle$ value (whereas the non-finite form *see* has no $\langle \text{tense} \rangle$ value). The relational EP $@_x\langle \text{GenRel} \rangle e$ triggers the lookup and instantiation of the two edges for the relative pronoun shown in (15) and (16) below. Similarly, the featural EP $@_e\langle \text{tense} \rangle \textbf{past}$ triggers the introduction of the auxiliary *did*.

(15) $\{1\}\ \{e, x\}$
that $\vdash (\mathsf{n}_x \backslash \mathsf{n}_x)/(\mathsf{s}_{e,fin} \backslash \mathsf{np}_x) : @_x\langle \text{GenRel} \rangle e$

(16) $\{1\}\ \{e, x\}$
that $\vdash (\mathsf{n}_x \backslash \mathsf{n}_x)/(\mathsf{s}_{e,fin} / \mathsf{np}_x) : @_x\langle \text{GenRel} \rangle e$

4.3 Combinatory Rules

In the second, main phase of the algorithm — at a high level — edges are successively moved from the agenda to the chart and combined with the edges already on the chart, with any resulting new edges added to the agenda, until no more combinations are possible and the agenda becomes empty. Figure 1 describes the main loop in more detail.

Continuing our example, some of the edges generated during the combinatory rule phase are shown in (17)–(21) below, without the bit vectors. The edge for *Bob* is type-raised, yielding (17), and

Until the agenda is empty:

1. Remove the first edge from the agenda and set it to be the current edge. If the chart contains an already derived equivalent edge, skip the rest of the loop.
2. Combine the current edge with the edges already on the chart. More specifically, for each chart edge:
   (a) Check the coverage bit vectors for the current edge and the chart edge for intersection. If they overlap, skip the chart edge.
   (b) Check the index bit vectors for intersection. If they do not overlap, only combine the current edge with the chart edge if the input LF contains an appropriate ⟨PAIREDWITH⟩ relation (cf. §5 for discussion).
   (c) Combine the current edge with the chart edge using all available binary combinatory rules, and add any resulting new edges to the agenda.
3. Apply all unary combinatory rules to the current edge, adding any resulting new edges to the agenda.
4. Combine the current edge with edges for all semantically null lexical items, as if these were chart edges.
5. Add the current edge to the chart.

Figure 1: Main loop

the edge for *see* (14) combines with the semantically null infinitival *to*, yielding (18); (17) then forward composes with both *saw* (13) and *to see* (18), yielding (19) and (20). Since *Bob to see* (20) is marked syntactically as infinitival rather than finite, the relative pronoun edge (16) will only combine (via forward application) with *Bob saw* (19), before combining (via backward application) with *man* to yield the complete edge in (21).

(17) *Bob* $\vdash \mathsf{s}_{1}/(\mathsf{s}_{1}\backslash\mathsf{np}_b) : @_b\mathbf{Bob}$

(18) *to see* $\vdash (\mathsf{s}_{e,inf}\backslash\mathsf{np}_b)/\mathsf{np}_x :$
$@_e\mathbf{see} \wedge @_e\langle\textsc{Act}\rangle b \wedge @_e\langle\textsc{Pat}\rangle x$

(19) *Bob saw* $\vdash \mathsf{s}_{e,fin}/\mathsf{np}_x :$
$@_e\mathbf{see} \wedge @_e\langle\text{TENSE}\rangle\mathbf{past}$
$\wedge @_e\langle\textsc{Act}\rangle b \wedge @_e\langle\textsc{Pat}\rangle x \wedge @_b\mathbf{Bob}$

(20) *Bob to see* $\vdash \mathsf{s}_{e,inf}/\mathsf{np}_x :$
$@_e\mathbf{see} \wedge @_e\langle\textsc{Act}\rangle b \wedge @_e\langle\textsc{Pat}\rangle x \wedge @_b\mathbf{Bob}$

(21) *man that Bob saw* $\vdash \mathsf{n}_x :$
$@_x\mathbf{man} \wedge @_x\langle\textsc{GenRel}\rangle e$
$\wedge @_e\mathbf{see} \wedge @_e\langle\text{TENSE}\rangle\mathbf{past}$
$\wedge @_e\langle\textsc{Act}\rangle b \wedge @_e\langle\textsc{Pat}\rangle x \wedge @_b\mathbf{Bob}$

5 Coordination

5.1 Sentential Coordination

CCG's flexible approach to constituency delivers derivations for a wide variety of coordinate structures, often involving the coordination of such "non-standard" constituents as s/np, as in the following right node raising example:

(22) [Bob saw]$_{\mathsf{s}/\mathsf{np}}$ and [Ted heard]$_{\mathsf{s}/\mathsf{np}}$ Gil.

Examples like (22) can be handled using the category for *and* given in (23), where s\$ schematizes over functions into s:[7]

(23) *and* $\vdash (\mathsf{s}_e\$_1\backslash\mathsf{s}_{e_1}\$_1)/\mathsf{s}_{e_2}\$_1 :$
$@_e\mathbf{and} \wedge @_e\langle\textsc{List}\rangle e_1 \wedge @_{e_1}\langle\textsc{Coord}\rangle e_2$

Category (23) enables *Bob saw* and *Ted heard* to coordinate as follows:

(24) *Bob saw and Ted heard* $\vdash$
$\mathsf{s}_e/\mathsf{np}_x :$
$@_e\mathbf{and} \wedge @_e\langle\textsc{List}\rangle e_1 \wedge @_{e_1}\langle\textsc{Coord}\rangle e_2$
$\wedge @_{e_1}\mathbf{see} \wedge \ldots @_{e_1}\langle\textsc{Pat}\rangle x$
$\wedge @_{e_2}\mathbf{hear} \wedge \ldots @_{e_2}\langle\textsc{Pat}\rangle x$

Category (24) can then be combined with *Gil* to yield a flat conjunction of HLDS terms equivalent to the one below (ignoring tense), which has been collapsed into hierarchical form for readability:

(25) $@_e(\mathbf{and}$
$\wedge \langle\textsc{List}\rangle(e_1 \wedge \mathbf{see} \wedge \langle\textsc{Act}\rangle(m\wedge\mathbf{Bob}) \wedge \langle\textsc{Pat}\rangle g$
$\wedge \langle\textsc{Coord}\rangle(e_2 \wedge \mathbf{hear}$
$\wedge \langle\textsc{Act}\rangle(t\wedge\mathbf{Ted}) \wedge \langle\textsc{Pat}\rangle g)))$
$\wedge @_g\mathbf{Gil}$

Since the present approach to semantic construction does not produce duplicate EPs for *Gil*, the output of the CCG parser for (22) shown in (25) can be directly reversed by the realizer. In contrast, the duplicate EPs seen in (10) (cf. §3.3) would cause problems for the realizer's tracking of input LF coverage. Indeed, the LF in (10) is perhaps more similar to the one for the clause-level coordination in (26) below than it is to (25):[8]

(26) *Bob heard Gil and Ted saw Gil* $\vdash$
$\mathsf{s} : @_e(\mathbf{and} \wedge \langle\textsc{List}\rangle(e_1 \wedge \mathbf{heard} \wedge \langle\textsc{Act}\rangle(b\wedge\mathbf{Bob})$
$\wedge \langle\textsc{Pat}\rangle(g_1\wedge\mathbf{Gil}) \wedge \langle\textsc{Coord}\rangle(e_2 \wedge \mathbf{see}$
$\wedge \langle\textsc{Act}\rangle(t\wedge\mathbf{Ted}) \wedge \langle\textsc{Pat}\rangle(g_2\wedge\mathbf{Gil}))))$

[7]The relations ⟨LIST⟩ and ⟨COORD⟩ encode a linked list; ⟨LIST⟩ points to the first item in the list, and ⟨COORD⟩ points from one item to the next.

[8]Note that each use of a lexical item gives rise to a distinct index nominal, similarly to DRT.

The HLDS terms in (25) and (26) show how differences in the realizer's input logical form — which are reminiscent of the differences between reduced and unreduced λ-terms — can be used to control the choice of coordination options made available by the grammar.[9]

5.2 NP Coordination

Of the multiple possible readings involving NP coordination, we will only focus on the distributive reading here. As Moore (1989) points out, NPs such as *Ted and Gil* in (27) below pose a challenge for first-order unification–based approaches to semantic construction, since the index x cannot be unified with the referents for both *Ted* and *Gil*:[10]

(27) $[\text{Bob saw}]_{\mathsf{s}_e/\mathsf{np}_x}$ Ted and Gil.

Following (Moore, 1989), we tackle this problem by introducing a λ-binder into the semantic representation for (27), while still eschewing the use of λ's in variable binding:

(28) $@_s(\textbf{and} \wedge \langle \textsc{List}\rangle(t \wedge \textbf{Ted} \wedge \langle \textsc{Coord}\rangle(g \wedge \textbf{Gil}))$
$\wedge \langle \textsc{Pred}\rangle(l \wedge \textbf{lambda} \wedge \langle \textsc{BoundVar}\rangle x$
$\wedge \langle \textsc{Body}\rangle(e \wedge \textbf{see} \wedge \langle \textsc{Act}\rangle(b \wedge \textbf{Bob}) \wedge \langle \textsc{Pat}\rangle x)))$

The HLDS term in (28) is intended to be equivalent to the conjunction of the terms formed by distributing the λ-term across each member of the list. (27) can be parsed and realized with the semantics in (28) using the category (29), which takes the two NPs and forms a type-raised NP:

(29) *and* $\vdash ((\mathsf{s}_s\$\backslash(\mathsf{s}_e\$/\mathsf{np}_x))\backslash\mathsf{np}_{x_1})/\mathsf{np}_{x_2}$:
$@_s\textbf{and} \wedge @_s\langle \textsc{List}\rangle x_1 \wedge @_{x_1}\langle \textsc{Coord}\rangle x_2$
$\wedge @_s\langle \textsc{Pred}\rangle l \wedge @_l\textbf{lambda}$
$\wedge @_l\langle \textsc{BoundVar}\rangle x \wedge @_l\langle \textsc{Body}\rangle e$

5.3 Argument Clusters and Gapping

The above approach to distributive NP coordination can be extended to handle argument clusters — as in (30) below — without the need to invoke otherwise unnecessary deletion operations.

(30) $[\text{Bob gave}]_{(\mathsf{s}_e/\mathsf{np}_y)/\mathsf{np}_x}$
$[\text{Ted}_t \text{ a dog}_d]_{\mathsf{s}\backslash(\mathsf{s}/\mathsf{np}_d/\mathsf{np}_t)}$ and
$[\text{Gil}_g \text{ a cat}_c]_{\mathsf{s}\backslash(\mathsf{s}/\mathsf{np}_c/\mathsf{np}_g)}$

To handle (30), we introduce a $\langle \textsc{PairedWith}\rangle$ relation to connect pairs of NP referents and bound variables, in the following category for *and*:

(31) *and* $\vdash ((\mathsf{s}_s\$\backslash((\mathsf{s}_e\$/\mathsf{np}_y)/\mathsf{np}_x))$
$\backslash(\mathsf{s}\$\backslash((\mathsf{s}\$/\mathsf{np}_{y_1})/\mathsf{np}_{x_1})))$
$/(\mathsf{s}\$\backslash((\mathsf{s}\$/\mathsf{np}_{y_2})/\mathsf{np}_{x_2}))$:
$@_s\textbf{and} \wedge @_s\langle \textsc{List}\rangle x_1 \wedge @_{x_1}\langle \textsc{PairedWith}\rangle y_1$
$\wedge @_{x_1}\langle \textsc{Coord}\rangle x_2 \wedge @_{x_2}\langle \textsc{PairedWith}\rangle y_2$
$\wedge @_s\langle \textsc{Pred}\rangle l \wedge @_l\textbf{lambda} \wedge @_l\langle \textsc{Body}\rangle e$
$\wedge @_l\langle \textsc{BoundVar}\rangle x \wedge @_x\langle \textsc{PairedWith}\rangle y$

Category (31) enables (30) to be parsed into a semantic representation analogous to (28). The derivation of (30) requires the base NPs $\textit{Ted}_t$ and $\textit{a dog}_d$ to type raise and compose together into the category $\mathsf{s}\backslash(\mathsf{s}/\mathsf{np}_d/\mathsf{np}_t)$, as indicated (and similarly for $\textit{Gil}_g$ and $\textit{a cat}_c$). Reversing this derivation during realization thus requires $\textit{Ted}_t$ and $\textit{a dog}_d$ to combine, even though they have no indices in common. Since removing the index intersection filter from the realization algorithm entirely would let all NPs combine via type-raising and composition in all possible orders, we instead require the indices to be in a $\langle \textsc{PairedWith}\rangle$ relation in the input LF in order for the NPs to combine.

To handle gapping examples like (32), a similar category can be supplied for *and*, as shown in (33) without the semantics, which remains unchanged:

(32) $\text{Ted}_t \text{ received}_{(\mathsf{s}_e\backslash\mathsf{np}_x)/\mathsf{np}_y} \text{ a dog}_d$ and
$[\text{Gil}_g \text{ a cat}_c]_{\mathsf{s}\backslash((\mathsf{s}\backslash\mathsf{np}_g)/\mathsf{np}_c)}$

(33) *and* $\vdash (((\mathsf{s}_s\backslash\mathsf{np}_{x_1})\backslash((\mathsf{s}_e\backslash\mathsf{np}_x)/\mathsf{np}_y))\backslash\mathsf{np}_{y_1})$
$/(\mathsf{s}\backslash((\mathsf{s}\backslash\mathsf{np}_{x_2})/\mathsf{np}_{y_2}))$

Category (33) combines first with the pair of NPs *Gil a cat* on the right, then successively with the NP *a dog*, the transitive verb *received* and the NP *Ted* on the left. As such, it handles gapping without appealing to reanalysis, as in Steedman (2000b), though at the expense of requiring *and* to coordinate unlike categories, suggesting that (33) should be viewed as a compiled-out version of Steedman's (2000b) approach to gapping.

6 Efficiency

As Moore (2002) notes, it appears that the realization problem is inherently exponential in worst case complexity unless one is willing to rely on

[9]Cf. (Prevost, 1995) for a related use of unreduced λ-terms in the context of representing information structural units.

[10]In the collective reading, also plausible in (27), x can simply be unified with a set-valued referent for *Ted and Gil*; with *Ted or Gil*, in contrast, only the distributive reading is possible.

	First	All
Avg	**0.19**	1.32
Max	0.98	13.0

Table 1: Realizer Timing (in seconds)

	First	All
Avg	0.50	13.3
Max	**3.84**	349

Table 2: Realizer Timing Without Index Filter

the potentially arbitrary order of LF conjuncts. In practice, as Carroll et al. (1999) explain, the main complexity issue is the factorial number of possible word orders that can arise when the grammar leaves modifier order relatively unconstrained. Our current strategy to address this issue is to concentrate on reliably finding good realizations in a reasonably short time span when running the algorithm in anytime mode, rather than worrying about the amount of time it might take on occasion to find all possible realizations. We suggest that this anytime focus is appropriate for practical use in dialog systems.

To test whether our realizer's speed is in the right ballpark for dialog applications, we have measured its performance on a pre-existing set of test phrases — namely all those discussed in Baldridge (2002) — using a small but linguistically rich grammar covering heavy NP shift, non-peripheral extraction, parasitic gaps, particle shift, relativization, right node raising, topicalization, and argument cluster coordination. On this test suite, the performance is reasonably promising, averaging under 200 ms. until the first realization is found, on a Linux PC. Table 1 shows the average and maximum times until the first realization is found and until all realizations are found.

Even with this small test suite, it is clear that the index filter is essential for efficient realization. Table 2 shows the comparable realization times with the index filter turned off. As the table shows, the average time until the first realization more than doubles, and the maximum time until the first realization is nearly four times worse. The expected exponential increase in realization times (cf. §5) can be seen in the times to find all realizations.

To increase performance, there is ample room to make improvements to the unification algorithm. While the index filter reduces the number of unification operations attempted, unification still dominates the realization time. The implementations of the combinatory rules have been optimized as described in Baldridge (2002), but unification is otherwise naive and performs more copying than necessary.

Employing packing and pruning strategies could also improve performance. Currently, there is no structure sharing among edges, and no means to prune low ranked edges from the chart.

7 Conclusions and Future Work

Our approach to chart realization with CCG is most closely related to Carroll et al. (1999), which in turn builds upon much earlier work cited therein, such as Kay (1996). Moore (2002) presents a related algorithm for a broad class of context free grammars.

Compared to Carroll et al. (1999), we have employed a similar but more straightforward approach to semantic construction than described in Copestake et al. (2001), since we do not allow underspecification of the logical scope of quantifiers,[11] and since there is no need for special treatment of external arguments to handle control phenomena in CCG. We also have not tried delaying the insertion of intersective modifiers (Carroll et al., 1999), in part because doing so would complicate the use of n-gram ranking strategies.

The primary novel contribution of our approach is showing how to efficiently realize a wide range of coordination phenomena with CCG. In particular, we have shown how to use an index filter sensitive to paired entities in the input LF in order to handle argument cluster coordination and gapping.

In future work, we plan to take several steps to make the realizer more practical. As already mentioned, we are currently exploring strategies for ranking partial solutions based on n-gram measures, and we plan to improve efficiency via enhancements to our unification algorithm. We are also currently investigating techniques for handling Steedman's (2000a) approach to information

[11]Cf. Steedman (1999) for discussion.

structure and intonation. In addition, we plan to bootstrap a wide coverage grammar for English from the CCG Bank (Hockenmaier and Steedman, 2002), and to develop improved XML grammar management tools.

Beyond these practically-oriented steps, we also plan to investigate new techniques for coupling CCG realization with higher level planning components. A particularly appealing direction is to see whether the present approach to coordination can simplify the treatment of aggregation in higher level planning components used in conjunction with the realizer. Since current bottom-up approaches to aggregation such as Dalianis (1996) and Shaw (1998) combine simple syntactic phrases into more complex ones by looking for patterns of related semantic material, they do not fit naturally into applications where it makes sense to group semantic material during content planning, based on intentions or information structural considerations. In contrast, working with our realizer, content planning components could specify their aggregation decisions via distinctions made at the level of logical form, taking advantage of the realizer's ability to use differences in the input LF to control the choice of coordination options made available by the grammar.

Acknowledgements

We would like to thank Mark Steedman, Geert-Jan Kruijff, Johanna Moore, Jon Oberlander and Mary Ellen Foster for helpful discussions. This work was supported in part by the COMIC (IST-2001-32311) and ROSIE (Edinburgh-Stanford Link R36763) projects.

References

Jason Baldridge and Geert-Jan Kruijff. 2002. Coupling CCG and Hybrid Logic Dependency Semantics. In *Proc. of 40th Annual Meeting of the Association for Computational Linguistics*, pages 319–326.

Jason Baldridge and Geert-Jan Kruijff. 2003. Multi-Modal Combinatory Categorial Grammar. In *Proc. of 10th Annual Meeting of the European Association for Computational Linguistics*.

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Patrick Blackburn. 2000. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–625.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznański. 1999. An efficient chart generator for (semi-) lexicalist grammars. In *Proc. of the 7th European Workshop on Natural Language Generation*, pages 86–95.

Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proc. of the 39th Annual Meeting of the Association of Computational Linguistics*, pages 132–139.

Hercules Dalianis. 1996. *Concise Natural Language Generation from Formal Specifications*. Ph.D. thesis, Royal Institute of Technology, Stockholm.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proc. of the Third International Conference on Language Resources and Evaluation*.

Beryl Hoffman. 1995. *Computational Analysis of the Syntax and Interpretation of 'Free' Word-order in Turkish*. Ph.D. thesis, University of Pennsylvania. IRCS Report 95-17.

Martin Kay. 1996. Chart generation. In *Proc. of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 200–204.

Geert-Jan M. Kruijff. 2001. *A Categorial Modal Architecture of Informativity: Dependency Grammar Logic & Information Structure*. Ph.D. thesis, Charles University.

Robert C. Moore. 1989. Unification-based semantic interpretation. In *Proc. of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 33–41.

Robert C. Moore. 2002. A complete, efficient sentence-realization algorithm for unification grammar. In *Proc. of the 2nd International Natural Language Generation Conference*.

Scott Prevost. 1995. *A Semantics of Contrast and Information Structure for Specifying Intonation in Spoken Language Generation*. Ph.D. thesis, University of Pennsylvania. IRCS TR 96-01.

James Shaw. 1998. Clause aggregation using linguistic knowledge. In *Proc. of the Ninth International Workshop on Natural Language Generation*, pages 138–148.

Stuart Shieber. 1993. The problem of logical-form equivalence. *Computational Linguistics*, 19(1):179–190.

Mark Steedman. 1999. Quantifier Scope Alternation in CCG. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 301–308.

Mark Steedman. 2000a. Information structure and the syntax-phonology interface. *Linguistic Inquiry*, 31(4):649–689.

Mark Steedman. 2000b. *The Syntactic Process*. MIT Press.

Sebastian Varges. 2001. Instance-based natural language generation. In *Proc. of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 1–8.

Experiments with discourse-level choices and readability

†Sandra Williams, †Ehud Reiter and ‡Liesl Osman
†Department of Computing Science, ‡Department of Medicine and Therapeutics
University of Aberdeen
{swilliam,ereiter}@csd.abdn.ac.uk med078@abdn.ac.uk

Abstract

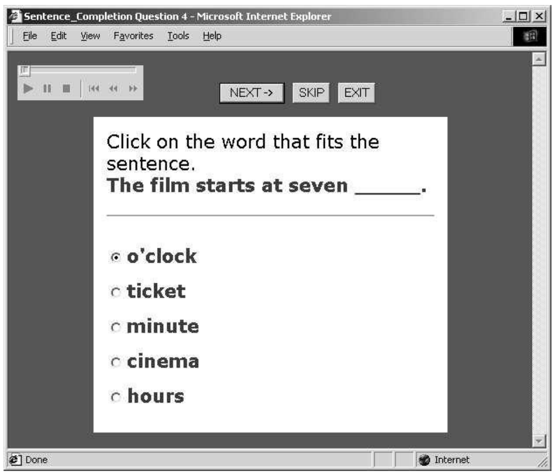
This paper reports on pilot experiments that are being used, together with corpus analysis, in the development of a Natural Language Generation (NLG) system, GIRL (Generator for Individual Reading Levels). GIRL generates reports for individuals after a literacy assessment.

We tested GIRL's output on adult learner readers and good readers. Our aim was to find out if choices the system makes at the discourse-level have an impact on readability. Our preliminary results indicate that such choices do indeed appear to be important for learner readers. These will be investigated further in future larger-scale experiments. Ultimately we intend to use the results to develop a mechanism that makes discourse-level choices that are appropriate for individuals' reading skills.

1. Introduction

The Generator for Individual Reading Levels (GIRL) project is developing a Natural Language Generation (NLG) system that generates feedback reports for adults after a web-based literacy assessment (Williams 2002).

The literacy assessment was designed by NFER-Nelson for the Target Skills application (2002) and it is aimed at adults with poor basic literacy. It produces a multi-level appraisal of a candidate's literacy skills. It tests eight skills: letter recognition, sentence completion, word ordering, form filling, punctuation and capitals, spelling, skimming and scanning and listening. The entire assessment consists of ninety questions, but the more difficult tests are only given to stronger candidates who have scored well on earlier tests. All questions are multiple choice. Figure 1 shows a screenshot of a typical question in the sentence completion test.


Sentence_Completion Question 4 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
NEXT -> SKIP EXIT
Done
Internet

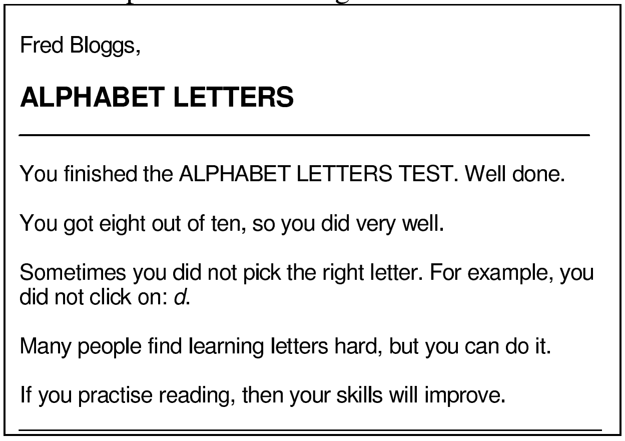
Click on the word that fits the sentence.
The film starts at seven ______.

[✓ o'clock]
ticket
minute
cinema
hours

Figure 1. A screen shot of a literacy test question.

In our implementation, each question is assembled on-the-fly by a web server program which retrieves question data (question text, graphics, audio file and multiple-choice answers) from a database. As each question page is downloaded, an audio file of spoken instructions plays automatically. A candidate can play the instructions again by clicking on the audio player graphic.

The inputs to the NLG system, GIRL, are the answers a candidate gives to questions in the literacy assessment. GIRL currently generates a feedback report after every test in the assessment. An example is shown in Figure 2.


Fred Bloggs,

ALPHABET LETTERS

You finished the ALPHABET LETTERS TEST. Well done.

You got eight out of ten, so you did very well.

Sometimes you did not pick the right letter. For example, you did not click on: *d*.

Many people find learning letters hard, but you can do it.

If you practise reading, then your skills will improve.

Figure 2. A type A "easy" report generated by GIRL.

GIRL is being developed with the goal of tailoring output texts to the individual reading skills of users (readers). In working towards this goal, we hope to find out more about generating documents for readers at different reading levels, how to test a system on real users, and how to implement reading level decision-making mechanisms as part of the generation process, and which decisions produce the most marked impact on the readability of the output texts.

It is very important to base the development of this system on solid empirical evidence rather than on our own intuitions. There has been very little empirical work on what kinds of texts are most appropriate for people with good reading skills and even less on what is appropriate for people with poor literacy. We were therefore motivated to do our own empirical studies. We are attacking the problem on two fronts: corpus analysis (Williams and Reiter 2003) and experiments with real readers, the subject of this paper.

1.1 Readability

Following Kintsch and Vipond (1979), we relate readability directly to readers' performance on the reading task (i.e. reading speed, ability to answer comprehension questions and ability to recall content). In these experiments, we measured reading speed and comprehension. We also analysed errors made in reading aloud, but that is not described here. The measures of readability we use are thus quantitative and are based on the hypotheses that readability increases or decreases with:

- an increase or decrease in the average reading rate for a particular reader;
- an increase or decrease in the number of correct answers given to comprehension questions.

1.2 Related work

We decided to investigate the impact of discourse-level choices as a novel approach to the problem of how to modify reports for different reading levels. Related work can be found in the PSET project (Devlin et al. 2000). PSET investigated how lexical-level choices and syntactic-level choices affect readability for aphasic readers, but it did not consider discourse choices.

As we mentioned above, there is very little empirical work to date on the impact of NLG system choices on readability and this is why it is so important for this project to carry out empirical work. One exception is the SPOT project (Walker et al. 2002). SPOT investigated which types of system outputs readers prefer. Readers were asked to rate the system's output utterances on understandability, well-formedness and appropriateness for the dialogue context. Apart from understandability, these do not relate to readability and we cannot assume that readers always choose the most readable utterances. They could be influenced by many other factors such as style. Also, all the judges were good readers and their preferences may not in any case be appropriate for learner readers.

2. The NLG system

A data-to-text NLG system like GIRL is able to linguistically express its output in a variety of ways that might affect readability. Here we look at features the microplanner can vary when linguistically realising discourse relations.

2.1 Deriving microplanner rules from a corpus analysis

Decisions about realising discourse relations are made in a module called the microplanner. The input is a tree of discourse relations joining pieces of information. It plans how the information from the tree will be ordered, whether it will be marked with discourse cue phrases (e.g. 'but', 'if' and 'for example'), and how it will be packed into sentences and punctuated.

To determine how human writers make these decisions for **good** readers, we carried out a corpus analysis (Williams and Reiter 2003). We used the RST Discourse Treebank Corpus (Carlson et al. 2002). The discourse relations analysed were *concession*, *condition*, *elaboration-additional*, *evaluation*, *example*, *reason* and *restatement*. The features we analysed are listed below.

- **Text span order.** The order of text spans in discourse relations. For instance "*because you got four out of ten, you need to practise*" or "*you need to practise because you got four out of ten* ".
- **Cue phrase existence and selection.** Whether cue phrases are present in a relation, or not, and which ones are used. For instance, cue phrases *if* and *then* are both present in "*if you practise, then you will improve*", but not in "*if you practise, you will improve*".
- **Cue phrase position.** The positions where cue phrases are located. For instance, for *example* is before the text span in *"for example, you did not click on the letter D"*, mid-span in *"you did not click, for example, on the letter D"*, and after it in *"you did not click on the letter D, for example"*. At present, GIRL can only handle positions *before* and *after*.
- **Existence and selection of between-span punctuation**. Sometimes there is punctuation between texts spans, e.g. the comma in *"many people find learning letters hard, but you can do it"* and the full stop in *"you finished the Alphabet Letters test. Well done",* sometimes there is none e.g. *"many people find learning letters hard but you can do it"*.
- **First text span length.** The length of the first text span in words.

The inspiration for choosing the first four features was Moser and Moore's analysis (1996).

The features are interdependent. For instance, choosing a particular ordering of text spans can constrain the choice of cue phrase. For instance, the first span in a relation can never have cue phrase *but* (Williams and Reiter 2003). The corpus analysis was therefore extremely useful in deriving a set of rules for choosing legal combinations of features for each relation.

We hypothesised that certain values for features were more likely to increase readability. Commas between segments make the discourse structure more explicit. Sentence-breaking punctuation gives shorter sentences and selection of short, common cue phrases can help learner readers. Sentence length and word length are both believed to have a major impact on readability (Flesch 1949).

The corpus analysis results were input to machine learning algorithms to derive decision trees and rules for GIRL's microplanner. But the analysis they are based on was a corpus written for good readers and we need data to adapt them for learner readers, so we carried out the experiments described here.

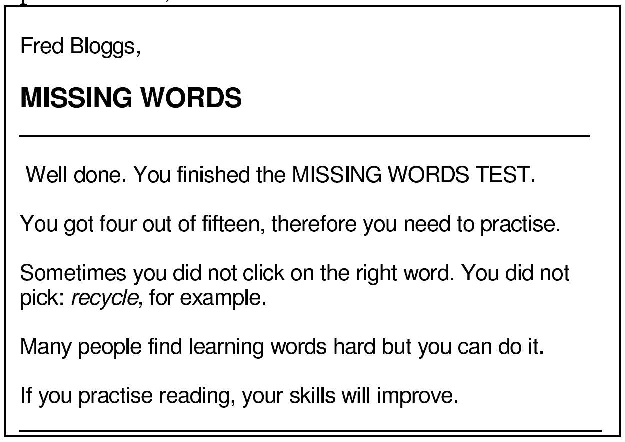
2.2 Modifications for the experiments

For the experiments, the system was modified to generate much shorter, more restricted, reports than those of the original GIRL system (Williams 2002). It was also modified to produce eight reports, one after each section of the literacy test, rather than a single long report after the entire test. These modifications increased our chances of collecting some reading data from each student, even if the student did not complete the entire literacy assessment (see section 4). Each short report consists of a salutation, a heading and exactly five paragraphs. Each paragraph consists of exactly one discourse relation. The system can produce two versions of each report, A and B (see Table 1).

In text types A and B, discourse relations were generated by varying only one discourse feature per paragraph. Table 1 shows which features were varied. Based on our corpus analysis results and on psycholinguistic evidence, we hypothesised that type A reports would be more readable ("easier") than type B reports ("harder").

Figure 2, shows an example of a type A report and Figure 3, a type B report. The text spans in the first paragraph are in statement:evaluation order in A and evaluation:statement in B. The second paragraph includes the cue phrase *so* in

A, and *therefore* in B. The third paragraph has *for example* before the second span in A and after it in B. The fourth paragraph has a comma present between text spans in A and no comma in B. Finally, the fifth paragraph has cue phrase *then* present in A, but not in B.



Fred Bloggs,

MISSING WORDS

Well done. You finished the MISSING WORDS TEST.

You got four out of fifteen, therefore you need to practise.

Sometimes you did not click on the right word. You did not pick: *recycle*, for example.

Many people find learning words hard but you can do it.

If you practise reading, your skills will improve.

Figure 3, A type B "harder" report generated by GIRL

Varying options in the manner shown in Table 1 ignores any crossover effects since there are actually 32 text types which would be possible from a 2x2x2x2x2 matrix of features. Also, any cumulative effects from having more than one discourse relation per paragraph are ignored in this design. We chose a simplified experimental design, since this was a pilot experiment. We wanted to test a large number of options and were aiming only to get indications of which options would have the greatest effects on readability. Future, more detailed experiments should look at crossover and cumulative effects.

3. Experiments

3.1 Participants

There were twenty-seven participants over the entire series of pilot experiments: twenty-one adults on basic skills literacy programmes (learner readers), four Ph.D. students and on other good readers. Because the design of the experiment was evolving, the conditions changed and we only use results from the final version. That is, nine learners and five good readers for reading speed and eleven learners for comprehension.

People who register for literacy courses are poor readers for a variety of reasons such as: missed school, learning difficulties, dyslexia, poor eyesight, poor hearing, short-term memory problems, or a combination of these. Personal data was recorded for each participant including age range, gender, first language, eyesight problems, hearing problems and any known reading problems (e.g. dyslexia). This data could be used to sub-classify readers, but the number of participants was too small to do this.

3.2 Method

Each participant underwent a web-based literacy assessment as described in the Introduction. After completion of each test in the assessment, GIRL generated feedback on how well the participant had done. The report is one of the two types described above and chosen by the system at random. In total, each participant was presented with between three and five reports of type A and three to five of type B.

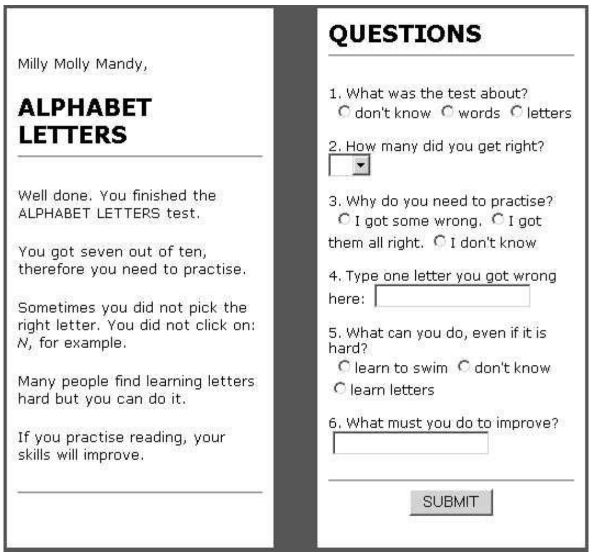
Each participant was recorded reading his/her reports aloud, rather than recording silent reading times. This is because we discovered in an earlier pilot that following the more usual procedure of asking participants to read silently and then click a button led to erroneous reading times for learners (Williams 2002). We could not be certain whether they had actually 'read' the reports or not. Recordings provide evidence that reading has, in fact, occurred.

The recordings were made digitally and were annotated by hand by the first author using CSLU's SpeechViewer software (Hosom et al. 1998). The speech waveforms were annotated with beginnings and ends of words and pauses

Paragraph	Discourse relation	Feature Varied	Report type A "easier"	Report type B "harder"
1	evaluation	text span order	statement:evaluation	evaluation:statement
2	reason/result	cue phrase choice	"so"	"therefore"
3	example	cue phrase position	before segment	after segment
4	concession	comma between spans	comma	no comma
5	condition	existence of cue phrase	"if" and "then"	"if" only

Table 1. The discourse features varied in each paragraph in reports type A and B.

and with reading errors. Using the resulting annotation files, timings for each word, pause and paragraph could be calculated accurately (to within, say, 10ms). Only paragraph time and some pause times are used here


Milly Molly Mandy,

ALPHABET LETTERS

Well done. You finished the ALPHABET LETTERS test.

You got seven out of ten, therefore you need to practise.

Sometimes you did not pick the right letter. You did not click on: *N*, for example.

Many people find learning letters hard but you can do it.

If you practise reading, your skills will improve.

QUESTIONS

1. What was the test about?
   ◯ don't know ◯ words ◯ letters
2. How many did you get right?
3. Why do you need to practise?
   ◯ I got some wrong. ◯ I got them all right. ◯ I don't know
4. Type one letter you got wrong here:
5. What can you do, even if it is hard?
   ◯ learn to swim ◯ don't know ◯ learn letters
6. What must you do to improve?

Figure 4. Comprehension questions are presented alongside a second view of the first report

After a participant had seen a screen showing his/her first report, had read aloud from that report and had been recorded, comprehension questions were presented (see Figure 4). The questions are displayed alongside a second view of the report and thus involved only comprehension, not recall. The experimenter read the questions aloud to learners, if necessary. Questions asked the meaning of information items in the report and of certain discourse relations. For example, question three is a 'why' question to determine if the reader has understood the relation in the second paragraph. Comprehension questions were administered only once, because an earlier pilot demonstrated that the meanings of each report are similar enough to prime readers.

On completion of the entire literacy assessment, an overall literacy level was calculated. This is subdivided into overall reading, overall writing and overall listening. Overall reading and overall writing are further subdivided into word focus, sentence focus and text focus scores.

If there was time after the experiment, informal chats with each participant provided useful information about readers' attitudes to the assessment and the reports. Participants offered ideas and suggestions for improvements. Basic skills tutors who were present during the pilots offered valuable suggestions.

4. Results

Not all learners managed to complete the literacy assessment because of time limits. Some learners and all good readers completed the test within an hour. Other learners took much longer, with one taking four hours! Two learners did not wish to be recorded reading aloud (although the majority of people were willing, sometimes even eager, to be recorded). Also, some recordings turned out to be too noisy. So we do not have a complete set of recordings for every person. For fourteen people, a maximum of 154 full text recordings were possible (720 paragraphs). We have good recordings of 297 paragraphs. All eleven learners completed the comprehension test.

4.1 Reading speed

Reading speeds were calculated in milliseconds per word (ms/word). Individuals, and particularly learner readers, vary a great deal in their reading aloud rates, so we calculated adjusted reading times for each person. The adjusted time is an individual's *raw time per word* for a single paragraph less that same person's *average time per word* over all of his/her recordings. In other words, adjusted times are a person's deviations from his/her average time. If the adjusted time is zero, then it is the same as that person's average time. A negative adjusted time means the person read faster and a positive adjusted time means they read slower. We were thus able to compare reading times for both versions of a paragraph, for all readers and calculate which version was read faster.

4.1.1 Paragraph 1: order of text spans

	statement :evaluation		evaluation :statement	
	#	adj.time	#	adj.time
Learners	26	-111.4	12	-47.1
Good readers	23	-37.3	16	-30.6

Table 2. Mean adjusted times in ms/word on two orderings, where # = number of samples.

Table 2 shows that learner readers read statement:evaluation order on average 64.3ms/word faster than evaluation:statement order. This result agrees with our RST Discourse Treebank corpus analysis (Williams and Reiter 2003) where we found that the statement:evaluation order is far more common. In fact we found this ordering present in 86% of *evaluation* relations. There is little difference in times for good readers (6.7 ms/word).

4.1.2 Paragraph 2: cue phrase selection

Table 3 shows that learner readers read relations with the cue phrase *so* on average 90.8ms/word faster than those containing *therefore*. Good readers' times showed a small difference of only 6.4ms/word. It could be argued that these differences are due to the fact that *therefore* has more, and longer, syllables than *so*. However, if this were the reason, then the differences would be the same for both types of reader.

	"so"		"therefore"	
	#	adj.time	#	adj.time
Learners	20	-126.3	12	-35.5
Good readers	13	-60.8	11	-54.4

Table 3, Average adjusted times in ms/word for *so* and *therefore*, where # = number of samples

4.1.3 Paragraph 3: cue phrase position

	before 2nd span		after 2nd span	
	#	adj. time	#	adj. time
Learners	20	7.5	12	-25.0
Good readers	13	17.0	15	13.8

Table 4, Average adjusted times in ms/word with for *example* before or after the 2nd span, where # = number of samples

Table 4 shows that learner readers were on average 32.5ms/word faster when *for example* was positioned *after* the second span, compared to before it. Again, there is little difference in times for good readers (3.2ms/word). The result for learners was unexpected because we thought people would read faster when they were told in advance that the information they were about to read was going to be an example (i.e. when *for example* is before the second span). If it is after the span, they have to re-evaluate the information they have just read. Also, we found very few examples of the *after* position in our RST Discourse Treebank analysis (Williams and Reiter 2003). Scarcity of the easier-to-read version in the corpus may provide further evidence for Oberlander's theory (Oberlander 1998) that writers do not always 'do the right thing' for readers. This result will be investigated further in future experiments.

4.1.4 Paragraph 4: between-span comma

Table 5 shows that learner readers read this paragraph on average 26.8ms/word faster when a comma was present. This is what we expected. The comma between text spans indicates the discourse structure more explicitly and we would expect it to help learner readers. Once again there is little difference in times for good readers (5.6ms/word). Since the cue phrase is present before the second span, the comma may be redundant for good readers. Future experiments will investigate this.

	comma		no comma	
	#	adj.time	#	adj.time
Learners	20	-59.6	12	-32.8
Good readers	7	-64.3	9	-69.9

Table 5, Average adjusted times in ms/word for a between-span comma or no comma, where # = number of samples

4.1.5 Paragraph 5: presence of second cue

	then		no *then*	
	#	adj. time	#	adj. time
Learners	18	-8.3	12	-37.3
Good readers	7	-57.8	9	-51.0

Table 6, Average adjusted times in ms/word with and without cue phrase *then* where # = number of samples

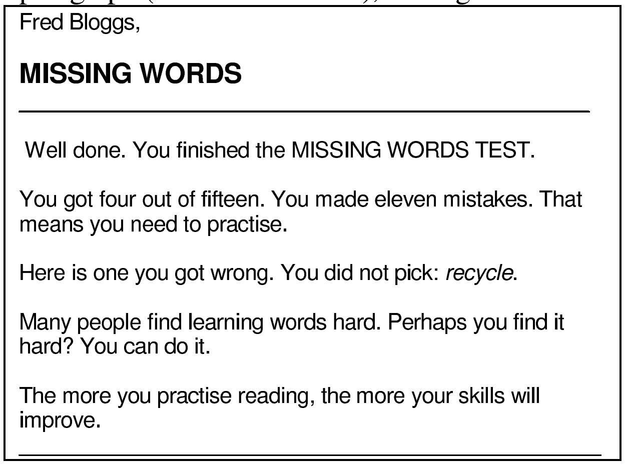
Table 6 shows that learner readers read relations with no second cue phrase 29.0ms/word faster. Good readers showed little difference in times (6.8ms/word). This is not what we expected. We expected the second cue phrase to help learners because it makes the condition relation more explicit when both *if* and *then* are present. This result ties in with our corpus analysis (Williams and Reiter 2003) where few cases with both cues present were found. Writers do not often use both cue phrases and learner readers seem to find an extra phrase adds difficulty rather than helping.

4.1.6 Sentence length

The figures for reading times vs. sentence length show that all readers are slower on sentences above 23 words in length, and some learners are slower above 18 words. We require more data to verify this.

4.2 Comprehension

We found that learner readers can have problems with answering comprehension questions, even when the questions are administered verbally. Some learner readers are unfamiliar with reasoning about textual meanings. Some find it very hard to create answers using different words from those that are present in the text they have just read. We therefore implemented a version of the system that generated more explanation for each paragraph (discourse relation), see Figure 5.



Fred Bloggs,

MISSING WORDS

Well done. You finished the MISSING WORDS TEST.

You got four out of fifteen. You made eleven mistakes. That means you need to practise.

Here is one you got wrong. You did not pick: *recycle*.

Many people find learning words hard. Perhaps you find it hard? You can do it.

The more you practise reading, the more your skills will improve.

Figure 5. 'Explanation' text

Figure 6 shows the comprehension scores for eleven learner readers. Scores, shown on the x-axis, are number of questions answered correctly out of six. Learner readers' scores are shown as gray bars and their mean scores are black bars.

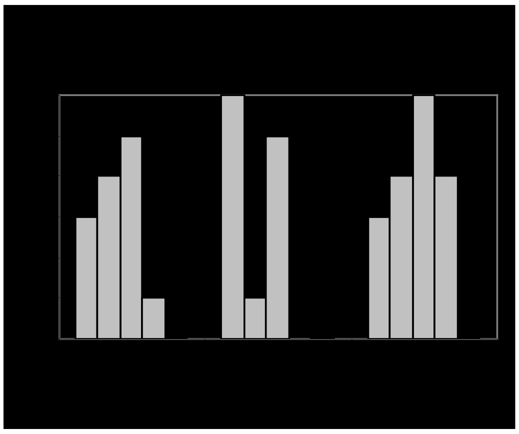


Figure 6, Grey bars = Learner readers' scores, Black bars = Mean learner readers' scores

Learners' highest mean comprehension score was on the *explanation* text type, but there is little difference between this and other mean scores. We simplified comprehension questions at the same time as introducing the explanation text, so we need to do further experiments to determine which has the most impact.

5. Conclusions

We previously analysed a corpus to determine how writers linguistically realise a number of discourse relations (Williams and Reiter 2003). Since the features analysed for each relation are interdependent (see section 2.2). Interdependencies can be conceptualised as a matrix like that shown in Table 7, where each cell (shown blank) actually contains rules (e.g. length vs. punctuation rules might be 1-10 words -> comma and >10 words -> full stop).

	length	order	punctuation	cue choice	cue position
length	■				
order		■			
punctuation			■		
cue choice				■	
cue position					■

Table 7. Interdependencies of features for one discourse relation

The corpus analysis results were input to machine learning algorithms to derive decision trees and sets of rules for GIRL's microplanner, so that given input text spans of fixed lengths linked in a discourse relation tree, it can determine ordering, between-span punctuation, cue choice and cue position. Since the corpus analysis was based on a corpus written for good readers, we required data from experiments like this to adapt the rules for learner readers. To find out in detail how to adapt each cell of the matrix for each relation, we need more extensive experiments than these. Nevertheless, our pilot experiments are a good start. They enabled us to develop and refine our experimental method. Our preliminary reading speed results show:

- **Text span order.** Learners were slightly faster reading statement:evaluation order. Good readers' speeds showed only small differences.
- **Cue phrase choice.** Learners were faster reading relations containing *so* than those containing *therefore*. Good readers were also slightly faster reading *so*. This result was the only statistically significant one.
- **Cue phrase position** Learner readers were slightly faster when *for example* was positioned **after** the second segment. The position made very little difference to good readers.

- **Presence of punctuation.** Learner readers were slightly faster when there was a comma between discourse segments. This made very little difference to good readers.

- **Cue phrase existence.** Learner readers were slightly faster when *then* is not present. This made very little difference to good readers.

Sentence length and comprehension results require further investigation. The reading speed results indicate that discourse realisation choices make a greater impact on the reading speeds of learner readers than on those of good readers. This is an important first step in acquiring empirical evidence from real readers who have poor literacy skills. Discourse-level choices do indeed make a difference for these readers. This information is very valuable for the development of the GIRL NLG system.

We require more extensive, larger-scale experiment to derive rules appropriate for adapting our existing corpus-analysis-based models to individuals' reading skills. We need to know the impact of each feature on all the others. For instance (a) and (b), below, are almost equally likely according to our previous corpus analysis of the *condition* relation:

a) If you need help, ask your tutor.
b) Ask your tutor if you need help.

These experiments have shown that learners find *a* easier than *b*, since it has a comma. However, the order of text spans in *b* could be easier. We do not yet know how each feature affects the others and which has the most impact. Nor do we yet know if readability changes for a particular feature as the discourse relation changes. The choices seem deceptively simple, but their impact on people with poor literacy can be considerable.

Acknowledgements

This research is supported by an EPSRC studentship. We are grateful to: the tutors and students who took part in experiments at: Moray College, South Shields College, Southampton City College, Banff and Buchan College and the University of Aberdeen; CTAD Ltd. for their help and advice and for permission to use the Target Skills Literacy Assessment; Somayajulu Sripada and the EWNLG reviewers for comments and advice.

References

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2002. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In *Current Directions in Discourse and Dialogue*, Jan van Kuppevelt and Ronnie Smith (eds.), Kluwer Academic Publishers.

Siobhan Devlin, Yvonne Canning, John Tait, John Carroll, Guido Minnen and Darren Pearce. 2000. An AAC aid for aphasic people with reading difficulties. In *Proceeding of the 9th Biennial Conference of the International Society for Augmentative and Alternative Communication* (ISADC 2000). August 2000, Washington D.C.

Rudolph Flesch. 1949. *The Art of Readable Writing. Harper*. USA.

John-Paul Hosom, Mark Fanty, Pieter Vermeulen, Ben Serridge and Tim Carmel. 1998. The Center for Spoken Language Understanding, Oregan Graduate Institute for Science and Technology.

Walter Kintsch and Douglas Vipond. 1979. Reading Comprehension and Readability in Educational Practice and Psychological Theory. In L.G. Nilsson (ed.) *Perspectives on Memory Research.* Lawrence Erlbaum.

Megan Moser and Johanna Moore 1996 On the correlation of cues with discourse structure: results from a corpus study. Unpublished manuscript.

Jon Oberlander. 1998. Do the Right Thing ...but Expect the Unexpected. *Computational Linguistics.* 24, 3. 501-507

Target Skills Literacy Assessment. 2002. Published by the Basic Skills Agency, in association with Cambridge Training and Development Ltd. and NFER-Nelson Ltd.

Marilyn Walker, Owen Rambow and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. In *Computer Speech and Language*. 16, 409-433.

Sandra Williams. 2002. Natural Language Generation of discourse connectives for different reading levels. In *Proceedings of the 5th Annual CLUK Research Colloquium.*

Sandra Williams and Ehud Reiter. 2003. A corpus analysis of discourse relations for Natural Language Generation. To appear in *Proceedings of Corpus Linguistics 2003.*

Author index