STRATEGIES FOR ADDING CONTROL INFORMATION TO DECLARATIVE GRAMMARS

Hans Uszkoreit
University of Saarbrücken
and German Research Center
for Artificial Intelligence (DFKI)
W-6600 Saarbrücken 11, FRG
uszkoreit@coli.uni-sb.de

Abstract

Strategies are proposed for combining different kinds of constraints in declarative grammars with a detachable layer of control information. The added control information is the basis for parametrized dynamically *controlled linguistic deduction*, a form of linguistic processing that permits the implementation of plausible linguistic performance models without giving up the declarative formulation of linguistic competence. The information can be used by the linguistic processor for ordering the sequence in which conjuncts and disjuncts are processed, for mixing depth-first and breadth-first search, for cutting off undesired derivations, and for constraint-relaxation.

1 Introduction

Feature term formalisms (FTF) have proven extremely useful for the declarative representation of linguistic knowledge. The family of grammar models that are based on such formalisms include Generalized Phrase Structure Grammar (GPSG) [Gazdar et al. 1985], Lexical Functional Grammar (LFG) [Bresnan 1982], Functional Unification Grammar (FUG) [Kay 1984], Head-Driven Phrase Structure Grammar (HPSG) [Pollard and Sag 1988], and Categorial Unification Grammar (CUG) [Karttunen 1986, Uszkoreit 1986, Zeevat et al. 1987].

Research for this paper was carried out in parts at DFKI in the project DISCO which is funded by the German Ministry for Research and Technology under Grant-No.: ITW 9002. Partial funding was also provided by the German Research Association (DFG) through the Project BiLD in the SFB 314: Artificial Intelligence and Knowledge-Based Systems. For fruitful discussions we would like to thank our colleagues in the projects DISCO, BiLD and LILOG as well as members of audiences at Austin, Texas, and Kyoto, Japan, where preliminary versions were presented. Special thanks for valuable comment and suggestions go to Gregor Erbach, Stanley Peters, Jim Talley, and Gertjan van Noord.

The expressive means of feature term formalisms have enabled linguists to design schemes for a very uniform encoding of universal and language-particular linguistic principles. The most radical approach of organizing linguistic knowledge in a uniform way that was inspired by proposals of Kay can be found in HPSG.

Unification grammar formalisms, or constraint-based grammar formalisms as they are sometimes called currently constitute the preferred paradigm for grammatical processing in computational linguistics.

One important reason for the success of unification grammars[1] in computational linguistics is their purely declarative nature. Since these grammars are not committed to any particular processing model, they can be used in combination with a number of processing strategies and algorithms. The modularity has a number of advantages:

- freedom for experimentation with different processing schemes,
- compatibility of the grammar with improved system versions,
- use of the same grammar for analysis and generation,
- reusability of a grammar in different systems.

Unification grammars have been used by theoretical linguists for describing linguistic competence. There exist no processing models for unification grammars yet that incorporate at least a few of the most widely accepted observations about human linguistic performance.

- Robustness: Human listeners can easily parse illformed input and adapt to patterns of ungrammaticality.

1The notion of grammar assumed here is equivalent to the structured collection of linguistic knowledge bases including the lexicon, different types of rule sets, linguistic principles, etc.

- Syntactic disambiguation in parsing: Unlikely derivations should be cut off or only tried after more likely ones failed. (attachment ambiguities, garden paths)
- Lexical disambiguation in parsing: Highly unlikely readings should be suppressed or tried only if no result can be obtained otherwise.
- Syntactic choice in generation: In generation one derivation needs to be picked out of a potentially infinite number of paraphrases.
- Lexical choice in generation: One item needs to be picked out of a large number of alternatives.
- Relationship between active and passive command of a language: The set of actively used constructions and lexical items is a proper subset of the ones mastered passively.

The theoretical grammarian has the option to neglect questions of linguistic performance and fully concentrate on the grammar as a correct and complete declarative recursive definition of a language fragment. The psycholinguist, on the other hand, will not accept grammar theory and formalism if no plausible processing models can be shown.

Computational linguists–independent of their theoretical interests–have no choice but to worry about the efficiency of processing. Unfortunately, as of this date, no implementations exist that allow efficient processing with the type of powerful unification grammars that are currently preferred by theoretical grammarians or grammar engineers. As soon as the grammar formalism employs disjunction and negation, processing becomes extremely slow. Yet the conclusion should not be to abandon unification grammar but to search for better processing models.

Certain effective control strategies for linguistic deduction with unification grammars have been suggested in the recent literature. [Shieber et al. 1990, Gerdemann and Hinrichs 1990] The strategies do not allow the grammar writer to attach control information to the constraints in the grammar. Neither can they be used for dynamic preference assignments. The model of control proposed in this paper can be used to implement these strategies in combination with others. However, the strategies are not encoded in the program but control information and parametrization of deduction.

The claim is that unification grammar is much better suited for the experimental and inductive development of plausible processing models than previous grammar models. The uniformily encoded constraints of the grammar need to be enriched by control information. This information serves the purpose to reduce local indeterminism through reordering and pruning of the search graph during linguistic deduction.

This paper discusses several strategies for adding control information to the grammar without sacrificing its declarative nature. One of the central hypotheses of the paper is that–in contrast to the declarative meaning of the grammar–the order in which subterms in conjunctions and disjunctions are processed is of importance for a realistic processing model. In disjunctions, the disjuncts that have the highest probability of success should be processed first, whereas in conjunctions the situation is reversed.

2 Control information in conjunctions

2.1 Ordering conjuncts

In this context conjuncts are all feature subterms that are combined explicitly or implicitly by the operation of feature unification. The most basic kind of conjunctive term that can be found in all FTFs is the conjunction of feature-value pairs.

$$\begin{bmatrix} f_1 : v_1 \\ f_2 : v_2 \\ \vdots \\ f_n : v_n \end{bmatrix}$$

Other types of conjunctive terms in the knowledge base may occur in formalisms that allow template, type or sort names in feature term specifications.

$$\begin{bmatrix} \text{Verb} \\ \text{Transitive} \\ \text{3rdSing} \\ \text{lex : hits} \\ \text{sem : hit'} \end{bmatrix}$$

If these calls are processed (expanded) at compile time, the conjunction will also be processed at compile time and not much can be gained by adding control information. If, however, the type or template calls are processed on demand at run time, as it needs to be the case in FTFs with recursive types, these names can be treated as regular conjuncts.

If a conjunction is unified with some other feature term, every conjunct has to be unified. Controlling the order in which operands are processed in conjunctions may save time if conjuncts can be processed first that are most likely to fail. This observation is the basis for a reordering method proposed by Kogure [1990]. If, e.g., in syntactic rule applications, the value of the attribute *agreement* in the representation of nominal elements

leads to clashes more often than the value of the attribute *definiteneess*, it would in general be more efficient to unify agreement before definiteness.

Every unification failure in processing cuts off some unsuccessful branch in the search tree. For every piece of information in a linguistic knowledge base we will call the probability at which it is directly involved in search tree pruning its failure potential. More exactly, the failure potential of a piece of information is the average number of times, copies of this (sub)term turn to $\perp$ during the processing of some input.

The failure path from the value that turns to $\perp$ first up to the root is determined by the logical equivalences

$\perp = \alpha : \perp$ (for any attribute α)
$\perp = [\perp\ \tau]$ (for any term τ)
$\tau = \{\perp\ \tau\}$ (for any term τ)
$\perp = \{\perp\}$
plus the appropriate associative laws.

Our experience in grammar development has shown that it is very difficult for the linguist to make good guesses about the relative failure potential of subterms of rules, principles, lexical entries and other feature terms in the grammar. However, relative rankings bases on failure potential can be calculated by counting failures during a training phase.

However, the failure potential, as it is defined here, may depend on the processing scheme and on the order of subterms in the grammar. If, e.g., the value of the agreement feature *person* in the definition of the type *Verb* leads to failure more often than the value of the feature *number*, this may simply be due to the order in which the two subterms are processed. Assume the unlikely situation that the value of *number* would have led to failure–if the order had been reversed–in all the cases in which the value of *person* did in the old order.

Thus for any automatic counting scheme some constant shuffling and reshuffling of the conjunct order needs to be applied until the order stabilizes (see also [Kogure 1990]).

There is a second criterion to consider. Some unifications with conjuncts build a lot of structure whereas others do not. Even if two conjuncts lead to failure the same number of times, it may still make a difference in which order they are processed.

Finally there might good reasons to process some conjuncts before others simply because processing them will bring in additional constraints that can reduce the size of the search tree. Good examples of such strategies are the so-called head-driven or functor-driven processing schemes.

The model of controlled linguistic deduction allows the marking of conjuncts derived by failure counting, processing effort comparisons, or psycholinguistic observations. However, the markings do not by themselves cause a different processing order. Only if deduction is parametrized appropriately, the markings will be considered by the type inference engine.

2.2 Relaxation markings

Many attempts have been made to achieve more robustness in parsing through more or less intricate schemes of rule relaxation. In FTFs all linguistic knowledge is encoded in feature terms that denote different kinds of constraints on linguistic objects. For the processing of grammatically illformed input, constraint relaxation techniques are needed.

Depending on the task, communication type, and many other factors certain constraints will be singled out for possible relaxation.

A relaxation marking is added to the control information of any subterm c encoding a constraint that may be relaxed. A relaxation marking consists of a function r_c from relaxation levels to relaxed constraints, i.e., a set of ordered pairs $<i, c_i>$ where i is an integer greater than 0 denoting a relaxation level and c_i is a relaxed constraint, i.e., a term subsuming c.[2]

The relaxation level is set as a global parameter for processing. The default level is 0 for working with an unrelaxed constraint base. Level 1 is the first level at which constraints are weakened. More than two relaxation levels are only needed if relaxation is supposed to take place in several steps.

If the unification of a subterm bearing some relaxation marking with some other term yields $\perp$, unification is stopped without putting $\perp$ into the partial result. The branch in the derivation is discontinued just as if a real failure had occurred but a continuation point for backtracking is kept on a backtracking stack. The partial result of the unification that was interrupted is also kept. If no result can be derived using the grammar without relaxation, the relaxation level is increased and backtracking to the continuation points is activated. The

[2]Implicitely the ordered pair $<0, c>$ is part of the control information for every subterm. Therefore it can be omitted.

subterm that is marked for relaxation is replaced by the relaxed equivalent. Unification continues. Whenever a (sub)term c from the grammar is encountered for which $r_C(i)$ is defined, the relaxed constraint is used.

This method also allows processing with an initial relaxation level greater than 0 in applications or discourse situations with a high probability of ungrammatical input.

For a grammar G let G_i be the grammar G except that every constraint is replaced by $r_C(i)$. Let L_i stand for the language generated or recognized by a grammar G_i. If constraints are always properly relaxed, i.e., if relaxation does not take place inside the scope of negation in FTFs that provide negation, L_i will always be a subset of L_{i+1}.

Note that correctness and completeness of the declarative grammar G_0 is preserved under the proposed relaxation scheme. All that is provided is an efficient way of jumping from processing with one grammar to processing with another closely related grammar. The method is based on the assumption that the relaxed grammars are properly relaxed and very close to the unrelaxed grammar. Therefore all intermediate results from a derivation on a lower relaxation level can be kept on a higher one.

3 Control information in disjunctions

3.1 Ordering of disjuncts

In this section, it will be shown how the processing of feature terms may be controlled through the association of preference weights to disjuncts in disjunctions of constraints. The preference weights determine the order in which the disjuncts are processed. This method is the most relevant part of controlled linguistic deduction. In one model control information is given statically, in a second model it is calculated dynamically.

Control information cannot be specified independent from linguistic knowledge. For parsing some readings in lexical entries might be preferred over others. For generation lexical choice might be guided by preference assignments. For both parsing and generation certain syntactic constructions might be preferred over others at choice points. Certain translations might receive higher preference during the transfer phase in machine translation.

Computational linguists have experimented with assignments of preferences to syntax and transfer rules, lexical entries and lexical readings. Preferences are usually assigned through numerical preference markers that guide lexical lookup and lexical choice as well as the choice of rules in parsing, generation, and transfer processes. Intricate schemes have been designed for arithmetically calculating the preference marker of a complex unit from the preference markers of its parts.

In a pure context-free grammar only one type of disjunction is used which corresponds to the choice among rules. In some unification grammars such as lexical functional grammars, there exist disjunction between rules, disjunction between lexical items and disjunction between feature-values in f-structures. In such grammars a uniform preference strategy cannot be achieved. In other unification grammar formalisms such as FUG or HPSG, the phrase structure has been incorporated into the feature terms. The only disjunction is feature term disjunction. Our preference scheme is based on the assumption that the formalism permits one type of disjunction only.

For readers not familiar with such grammars, a brief outline is presented. In HPSG grammatical knowledge is fully encoded in feature terms. The formalism employs conjunction (unification), disjunction, implication, and negation as well as special data types for lists and sets. Subterms can also be connected through relational constraints. Linguistically relevant feature terms are order-sorted, i.e., there is a partially ordered set of sorts such that every feature term that describes a linguistic object is assigned to a sort.

The grammar can be viewed as a huge disjunctive constraint on the wellformedness of linguistic signs. Every wellformed sign must unifiy with the grammar. The grammar consists of a set of universal principles, a set of language-particular principles, a set of lexical entries (the lexicon), and a set of phrase-structure rules.

The grammar of English contains all principles of universal grammar, all principles of English, the English lexicon, and the phrase-structure rules of English. A sign has to conform with all universal and language-particular principles, therefore these principles are combined in conjunctions. It is either a lexical sign in which case it has to unify with at least one lexical entry or it is a phrasal sign in which case it needs to unify with at least one phrase-structure rule. The lexicon and the set of rules are therefore combined in disjunctions.

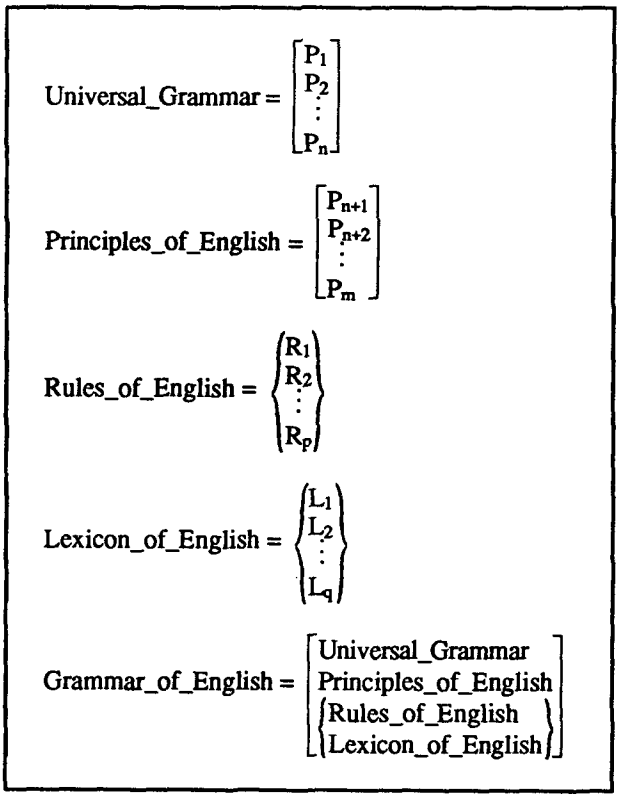
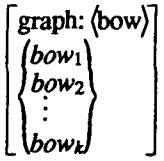


Figure 1. Organization of the Grammar of English in HPSG

Such a grammar enables the computational linguist to implement processing in either direction as mere type inference. However, we claim that any attempts to follow this elegant approach will lead to terribly inefficient systems unless *controlled linguistic deduction* or an equally powerful parametrizable control scheme is employed.

Controlled linguistic deduction takes advantage of the fact that a grammar of the sort shown in Figure 1 allows a uniform characterization of possible choice points in grammatical derivation. Every choice point in the derivation involves the processing of a disjunction. Thus feature disjunction is the only source of disjunction or nondeterminism in processing. This is easy to see in the case of lexical lookup. We assume that a lexicon is indexed for the type of information needed for access. By means of distributive and associative laws, the relevant index is factored out. A lexicon for parsing written input is indexed by a feature with the attribute *graph* that encodes the graphemic form. A lexicon with the same content might be used for generation except that the index will be the semantic content.

An ambiguous entry contains a disjunction of its readings. In the following schematized entry for the English homograph *bow* the disjunction contains everything but the graphemic form.[3]



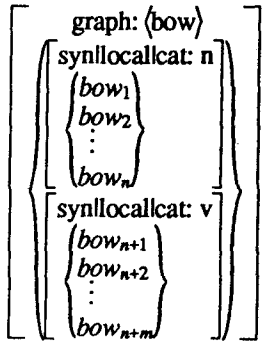
3.2 Static preferences

There exist two basic strategies for dealing with disjunctions. One is based on the concept of backtracking. One disjunct is picked (either at random or from the top of a stack), a continuation point is set, and processing continues as if the picked disjunct were the only one, i.e., as if it were the whole term. If processing leads to failure, the computation is set back completely to the fixed continuation point and a different (or next) disjunct is picked for continuation. If the computation with the first disjunct yields success, one has the choice of either to be satisfied with the (first) solution or to set the computation back to the continuation point and try the next disjunct. With respect to the disjunction, this strategy amounts to depth-first search for a solution.

The second strategy is based on breadth-first search. All disjuncts are used in the operation. If, e.g., a disjunction

---

[3]Additional information such as syntactic category might also be factored out within the entry:



However, all we are interested in in this context is the observation that in any case the preferences among readings have to be associated with disjuncts.

is unified with a nondisjunctive term, the term is unified with every disjunct. The result is again a disjunction.

The strategy proposed here is to allow for combinations of depth-first and breadth-first processing. Depth-first search is useful if there are good reasons to believe that the use of one disjunct will lead to the only result or to the best result. A mix of the two basic strategies is useful if there are several disjuncts that offer better chances than the others.

Preference markers (or preference values) are attached to the disjuncts of a disjunction. Assume that a preference value is a continuous value p in $0 \leq p \leq 10$. Now a global width factor w in $0 \leq w \leq 10$ can be set that separates the disjuncts to be tried out first from the ones that can only be reached through backtracking.

All disjuncts are tried out first in parallel whose values p_i are in $p_{max}-w \leq p_i \leq p_{max}$. If the width is set to 2, all disjuncts would be picked that have values p_i in $p_{max}-2 \leq p_i \leq p_{max}$. Purely depth-first and purely breadth-first search are forced by setting the threshold to 0 or 10 respectively.

3.3 Dynamic preferences

One of the major problems in working with preferences is their contextual dependence. Although static preference values can be very helpful in guiding the derivation, especially for generation, transfer, or limiting lexical ambiguity, often different preferences apply to different contexts.

Take as an example again the reduction of lexical ambiguity. It is clearly the context that influences the hearers preferences in selecting a reading.[4]

The astronomer married a star. vs.
The movie director married a star.

The tennis player opened the ball. vs.
The mayor opened the ball.

Preferences among syntactic constructions, that is preferences among rules, depend on the sort of text to be processed.

A trivial but unsatisfactory solution is to substitute the preference values by a vector of values. Depending on the subject matter, the context, or the approriate style or register, different fields of the vector values might be considered for controlling the processing.

However, there are several reasons that speak against such a simple extension of the preference mechanism. First of all, the number of fields that would be needed is much too large. For lexical disambiguation, a mere classification of readings according to a small set of subject domains as it can be found in many dictionaries is much too coarse.

Take, e.g., the English word *line*. The word is highly ambiguous. We can easily imagine appropriate preferred readings in the subject domains of telecommunication, geometry, genealogy, and drug culture. However, even in a single computer manual the word may, depending on the context, refer to a terminal line, to a line of characters on the screen, to a horizontal separation line between editing windows, or to many other things. (In each case there is a different translation into German.)

A second reason comes from the fact that preferences are highly dynamic, i.e., they can change at any time during processing. Psycholinguistic experiments strongly suggest that the mere perception of a word totally out of context already primes the subject, i.e., influences his preferences in lexical choice. [Swinney 1979]

The third reason to be mentioned here is the multifactorial dependency of preferences. Preferences can be the result of a combination of factors such as the topic of the text or discourse, previous occurrence of priming words, register, style, and many more.

In order to model the dynamics of preferences, a processing model is proposed that combines techniques from connectionist research with the declarative grammar formalisms through dynamic preference values.

Instead of assigning permanent preference values or value vectors to disjuncts, the values are dynamically calculated by a spreading-activation net. So far the potentials of neural nets for learning (e.g. backpropagation schemes) have not been exploited. Every other metaphor for setting up weighted connections between constraints in disjunctions would serve our purpose equally well.[5]

---

[4] The first example is due to Reder [1983].

[5] For an introduction to connectionist nets see Rumelhart, Hinton, and McClelland [1986]. For an overview of different connectionist models see Feldman and Ballard [1982] and Kemke [1988].

The type of net employed for our purposes is extremely simple.[6] Every term in the linguistic knowledge bases whose activation may influence a preference and every term whose preference value may be influenced is associated with a unit. These sets are not disjoint since the selection of one disjunct may influence other preferences. In addition there can be units for extralinguistic influences on preferences. Units are connected by unidirectional weighted links. They have an input value i, an activation value a, a resting value r, and a preservation function f. The input value is the sum of incoming activation. The resting value is the minimal activation value, i.e., the degree of activation that is independent from current or previous input. The activation value is either equal to the sum of input and some fraction of the previous activation, which is determined by the preservation function or it is equal to the resting value, whichever is greater.

$a_{i+1} = max\{r, i_i + f(a_i)\}$.

In this simple model the output is equal to the activation. The weights of the links l are factors such that $0 \leq l \leq 1$. If a link goes from unit u_1 to unit u_2, it contributes an activation of $l*a_{u_1}$ to the input of u_2.

4 Conclusion and future research

Strategies are proposed for combining declarative linguistic knowledge bases with an additional layer of control information. The unification grammar itself remains declarative. The grammar also retains completeness. It is the processing model that uses the control information for ordering and pruning the search graph. However, if the control information is neglected or if all solutions are demanded and sought by backtracking, the same processing model can be used to obtain exactly those results derived without control information.

Yet, if control is used to prune the search tree in such a way that the number of solutions is reduced, many observations about human linguistic performance some of which are mentioned in Section 1 can be simulated. Criteria for selection among alternatives can be encoded. The smaller set of actively used constructions and lexemes is simply explained by the fact that for all the items in the knowledge base that are not actively used there are alternatives that have a higher preference.

The controlled linguistic deduction approach offers a new view of the competence-performance distinction, which plays an important rôle in theoretical linguistics. Uncontrolled deduction cannot serve as a plausible performance model. On the other hand, the performance model extends beyond the processing model, it also includes the structuring of the knowledge base and control information that influence processing.

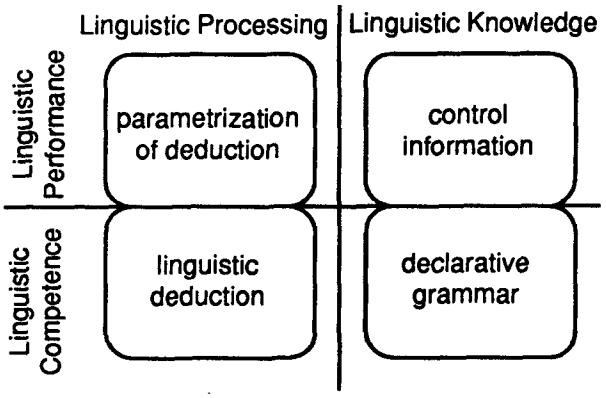


Figure 2. A new view of the competence-performance distinction

Since this paper reports about the first results from a new line of research, many questions remain open and demand further research.

Other types of control need to be investigated in relation with the strategies proposed in this paper. Uszkoreit [1990], e.g., argues that functional uncertainty needs to be controlled in order to reduce the search space and at the same time simulate syntactic preferences in human processing.

Unification grammar formalisms may be viewed as constraint languages in the spirit of constraint logic programming (CLP). Efficiency can be gained through appropriate strategies for delaying the evaluation of different constraint types. Such schemes for delayed evaluation of constraints have been implemented for LFG. They play an even greater role in the processing of Constraint Logic Grammars (CLG) [Balari et al. 1990]. The delaying scheme is a more sophisticated

[6] The selected simple model is sufficient for illustrating the basic idea. Certainly more sophisticated connectionist models will have to be employed for cognitively plausible simulation. One reason for the simple design of the net is the lack of a learning. At this time, no learning model has been worked out yet for the proposed type of spreading-activation nets. For the time being it is assumed that the weights are set by hand using linguistic knowledge, corpora, and association dictionaries.

method for the ordering of conjuncts. More research is needed in this area before the techniques of CLP/CLG can be integrated in a general model of controlled (linguistic) deduction.

So far the weight of the links for preference assignment can only be assigned on the basis of association dictionaries as they have been compiled by psychologists. For nonlexical links the grammar writer has to rely on a trial and error method.

A training method for inducing the best conjunct order on the basis of failure potential was described in Section 2.1. The training problem, .ie., the problem of automatic induction of the best control information is much harder for disjunctions. Parallel to the method for conjunctions, during the training phase the success potential of a disjunct needs to be determined, i.e., the average number of contributions to successful derivations for a given number of inputs. The problem is much harder for assigning weights to links in the spreading-activation net employed for dynamic preference assignment.

Hirst [1988] uses the structure of a semantic net for dynamic lexical disambiguation. Corresponding to their marker passing method a strategy should be developed that activates all supertypes of an activated type in decreasing quantity. Wherever activations meet, a mutual reinforcement of the paths, that is of the hypotheses occurs.

Another topic for future research is the relationship between control information and feature logic. What happens if, for instance, a disjunction is transformed into a conjunction using De Morgans law?

$$\begin{Bmatrix}\tau_1\\ \tau_2\end{Bmatrix} = \neg\begin{bmatrix}\neg\tau_1\\ \neg\tau_2\end{bmatrix}$$

The immediate reply is that control structures are only valid on a certain formulation of the grammar and not on its logically equivalent syntactic variants. However, assume that a fraction of a statically or dynamically calculated fraction involving success potential *sp and* failure potential *fp* is attached to every subterm. For disjuncts, *sp* is divided by *fp*, for conjuncts *fp* is divided by *sp*.

$$\left\{\begin{matrix}\frac{sp_1}{fp_1}\tau_1\\ \frac{sp_2}{fp_2}\tau_2\end{matrix}\right\} \qquad \left[\begin{matrix}\frac{fp_1}{sp_1}\tau_1\\ \frac{fp_2}{sp_2}\tau_2\end{matrix}\right]$$

De Morgans law yields an intuitive result if we assume that negation of a term causes the attached fraction to be inverted. More research needs to be carried out before one can even start to argue for or against a preservation of control information under logical equivalences.

Head-driven or functor-driven deduction has proven very useful. In this approach the order of processing conjuncts has been fixed in order to avoid the logically perfect but much less effcient orderings in which the complement conjuncts in the phrase structure (e.g., in the value of the *daughter* feature) are processed before the head conjunct. This strategy could not be induced or learned using the simple ordering criteria that are merely based on failure and success. In order to induce the strategy from experience, the relative computational effort needs to be measured and compared for the logically equivalent orderings. Ongoing work is dedicated to the task of formulating well-known processing algorithms such as the Earley algorithm for parsing or the functor-driven approach for generation purely in terms of preferences among conjuncts and disjuncts.

References

Balari, S. and L. Damas (1990) CLG(n): Constraint Logic Grammars. In *COLING 90.*

Bresnan, J. (Ed.) (1982) *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, Mass.

Feldman, J.A. and D.H. Ballard (1982) Connectionist models and their Properties. In *Cognitive Science,* 6:205-254.

Gazdar, G., E. Klein, G. K. Pullum, and I. A. Sag (1985) *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge, Mass.

Gerdemann, D. and E. W. Hinrichs (1990) Functor-Driven Natural Language Generation with Categorial-Unification Grammars. In *COLING 90.*

Hirst, G. (1988) Resolving Lexical Ambiguity Computationally with Spreading Activation and Polaroid Words. In S.I. Small, G.W. Cottrell and M.K. Tanenhaus (eds.), *Lexical Ambiguity Resolution,* pp.73-107. San Mateo: Morgan Kaufmann Publishers.

Karttunen, L. (1986) *Radical Lexicalism.*Technical Report CSLI-86-66, CSLI - Stanford University.

Kasper, R. and W. Rounds (1986) A Logical Semantics for Feature Structures. In *Proceedings of the 24th ACL.*

Kay, M. (1984) Functional Unification Grammar: A Formalism for Machine Translation. In *COLING 84.*

Kemke, C. (1988) Der Neuere Konnektionismus - Ein Überblick. *Imformatik-Spektrum,* 11:143-162.

Kogure, K. (1990) Strategic Lazy Incremental Copy Graph Unification. In *COLING 90.*

Pollard, C. and I. A. Sag (1988) An Information-Based Syntax and Semantics, Volume.1 Fundamentals, CSLI Lecture Notes 13, CSLI, Stanford, CA.

Reder, L.M., (1983) What kind of pitcher can a catcher fill? Effects of priming in sentence comprehension. In *Journal of Verbal Learning and Verbal Behavior* 22 (2):189-202.

Rumelhart, D.E., G.E. Hinton and J.L. McClelland (1986) A general framework for parallel distributed processing. In Rumelhart, D.E., McClelland, J.L., and the PDP Research Group, editors, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition: Foundations,* volume 1 pages 45-76. Cambridge, MA: MIT Press.

Shieber, S. (1984) The Design of a Computer Language for Linguistic Information. In S. Shieber, L. Karttunen, and F. Pereira (Eds.) *Notes from the Unification Underground.* SRI Technical Note 327, SRI International, Menlo Park, CA.

Shieber, S. M. (1986) An Introduction to Unification-Based Approaches to Grammar. CSLI Lecture Notes 4. CSLI, Stanford, CA.

Shieber, S., H. Uszkoreit, J. Robinson, and M. Tyson (1983) The Formalism and Implementation of PATR-II. In *Research on Interactive Acquisition and Use of Knowledge.* SRI International, Menlo Park, CA.

Shieber, S., G. van Noord, R.C. Moore, and F.C.N. Pereira (1990) Semantic-Head-Driven Generation In *Computational Linguistics* 16(1).

Smolka, G. (1988) *A Feature Logic with Subsorts.* LILOG Report 33, IBM Germany, Stuttgart.

Smolka, G., and H. Aït-Kaci (1987) *Inheritance Hierarchies: Semantics and Unification.* MCC Report AI-057-87, MCC Austin, TX.

Swinney, D.A. (1979) Lexical Access during sentence comprehension: (Re)Consideration of context effects. In *Journal of Verbal Learning and Verbal Behavior* 18(6):645-659.

Uszkoreit, H. (1986) Categorial Unification Grammars. In *COLING 86,* Bonn.

Uszkoreit, H. (1988) From Feature Bundles to Abstract Data Types: New Directions in the Representation and Processing of Linguistic Knowledge. In A. Blaser (Ed.) *Natural Language at the Computer.* Springer, Berlin, Heidelberg, New York.

Uszkoreit, H. (1990) „Extraposition and Adjunct Attachment in Categorial Unification Grammar" In W. Bahner (Hrsg.) *Proceedings of the XIVth International Congress of Linguists,* August 1987, Akademie Verlag Berlin, DDR, 1990.

Zeevat, H., E. Klein, and J. Calder (1987) Unification Categorial Grammar. In Haddock, Klein, and Morrill (Eds.) *Categorial Grammar, Unification Grammar, and Parsing.* Edinburgh Working Papers in Cognitive Science, Vol.1. Centre for Cognitive Science, University of Edinburgh, Edinburgh.