Guarded Constraints in Natural Language Processing

Thesis Summary

Kathryn L. Baker

April 24, 2002

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:
Bob Carpenter, Chair
Teruko Mitamura, Co-Chair
Chris Manning
Carl Pollard
Rich Thomason

Contents

0.1 Introduction

In a linguistic theory, features constrain a well-formed constituent, such as a phrase or a sentence. Identifying such a constituent from an input can be cast as a constraint-solving problem. Delaying the resolution of certain constraints is an established method for solving goals in logic programming languages. This method can be adapted not only for linguistic problem solving in general, but specifically for constraints in the form of complex, typed feature structures. Constraints which wait upon more information are called guarded constraints.

This thesis generalizes delays for processing a linguistic theory by providing a specification for guarded constraints as descriptions of typed feature structures. I extend the description language, a shorthand for typed feature structures, described fully in Carpenter (1992). My work enables the successful evaluation of modern grammars as written, even with highly lexicalized constraints. The work includes a full description of the implementation of three relevant linguistic examples. These are the Japanese causative, German topicalization, and quantifier scoping in English.

By generalizing delays over feature structures, I make three primary contributions. First, I give the operational semantics of guarding on typed feature structures. I show how either rules can be guarded, or descriptions of feature structures themselves can be guarded. Descriptions constrain the choice of available goals during resolution. Second, I place the work in a linguistic context, presenting a variety of analyses for which delaying is a good approach. The work enables a uniform treatment of many phenomena, all of which are analyzed linguistically via argument sharing. Third, with guards, I extend ALE (Carpenter & Penn 1994), a logic programming system that can be used for parsing and generation of large-scale grammars. The work is language independent and independent of the underlying parsing or generation algorithm.

The thesis is organized as follows. In chapter two, I review the relevant linguistic literature, providing a wide range of linguistic data for which the technique of delaying will prove useful in implementation. I show an example of a problem which is encountered when attempting to evaluate underspecified feature structures during processing using traditional means. In chapter three, I review the field of Constraint Logic Programming (CLP) and its application in Computational Linguistics. In chapter four, I lay out the framework of feature structures, and review the existing CLP systems for processing feature based grammars. Next, in chapter five, I give the operational semantics of delaying on typed feature structure descriptions. I show how delaying is used in the case of verb raising by auxiliary in German. I discuss my approach to inequations as an instance of guarded descriptions. In chapter six, I use the binding theory as a linguistic test case for the system. I present this example for Japanese. I also present quantifier retrieval in English as an example of guarding on types. I evaluate the work with respect to the specification, the grammar and lexicon, and the parsing algorithm. Finally, in chapter seven, I draw conclusions about this work. The thesis is extensible to a wide range of linguistic analyses. The resolution algorithm is abstracted away from the parsing architecture, providing a practical advantage for implementation efforts.

0.2 Describing Linguistic Objects

I begin by presenting a cross section of linguistic data which analyzes arguments as being shared by more than one head. For example, in the case of raising structures, a verb like *seem* in English might share an NP subject with another verb, a predicate for which it subcategorizes. A variety of contexts include argument raising by Germanic and Romance auxiliaries, sharing by semantic predicates in Chicheŵa and Japanese, lexical rules for syntactic operations such as the passive and complement extraction, quantifier raising, and argument sharing in word order domains. These analyses are unified by the fact that arguments are shared across more than one structure. This may be a semantic predicate, a subcategorization frame, a word order domain, or a nonlocal set (such as the SLASH value or the quantifier store).

Argument sharing is relevant in processing because one head may rely on information about an argument that is provided only within the context of the other head. And so, the order of argument instantiation becomes relevant. I address the impact of structure sharing on evaluation. I use the case of raising by the German auxiliary *werden* as a specific example. The question we consider in particular is whether typical processing methods have been sufficient for cases of argument sharing.

A problem we discover is that one head might need to gain information which is projected from a different head, but underspecified. In the processing paradigm, one way a feature structure is underspecified is if the path value at the node which is being evaluated is undefined. It may also be that the type in a typed feature structure is not a maximally specific type, or that an entire piece of the structure is co-referenced or shared with another. In a parsing or generation process, we store all the information we gain as we go along in a common feature structure, updating that information and creating new path values as we process. It may be the case that a feature value that is undefined or not maximally resolved early in the computation will be more specific later, as a result of gaining new information during the parse.

A simple feature structure showing the semantic features for the noun *sheep* might look as in example 1. The type **nom-object** refers to the fact that a sheep belongs to the class of things that are nominal objects.

$$(1)\quad \begin{bmatrix} \textbf{nom-object} & \\ \textsc{content}: & \begin{bmatrix} \textbf{content} \\ \textsc{person}:\textbf{3rd} \\ \textsc{number}:\textbf{number} \\ \textsc{relation}:\textbf{sheep} \end{bmatrix} \end{bmatrix}$$

Clues from other parts of the grammar could help to make the number feature more specific when the feature structure gains information during processing. We refer to these clues as constraints. In the sentence *The sheep graze*, the filled-in version of the feature structure has the subtype of **number** resolved to **plural** in 2:

$$(2)\quad \begin{bmatrix} \textbf{nom-object} & \\ \textsc{content}: & \begin{bmatrix} \textbf{content} \\ \textsc{person}:\textbf{3rd} \\ \textsc{number}:\textbf{plural} \\ \textsc{relation}:\textbf{sheep} \end{bmatrix} \end{bmatrix}$$

The sentence *The sheep bleats* would likewise resolve to the singular form. The number for the noun is obtained via unification with the number form on the inflected verb. Typically, the number for the verb form is added into a basic, also underspecified feature structure for the verb's base form, by way of some morphological analysis.

For our test case, we look at a lexical rule that has the potential to extract underspecified complements from an auxiliary at the point of the lexicon. The feature structure template for the German raising auxiliary *werden* (*will*) is shown here as example 3.

$$(3)\quad \begin{bmatrix} \textbf{werden} & \\ \textsc{head}: & \begin{bmatrix} \textbf{verb} \\ \textsc{vform}:\textbf{bse} \\ \textsc{aux}:\textbf{+} \end{bmatrix} \\ \textsc{subj}:\boxed{1}\langle \textbf{NP} \rangle & \\ \textsc{comps}:\boxed{2} + & \left\langle \begin{bmatrix} \textsc{head}: \begin{bmatrix} \textbf{verb} \\ \textsc{vform}:\textbf{bse} \end{bmatrix} \\ \textsc{comps}:\boxed{2} \\ \textsc{subj}:\boxed{1} \\ \textsc{npcomp}:\textbf{-} \end{bmatrix} \right\rangle \end{bmatrix}$$

I introduce Partial Verb Phrase (PVP) fronting as a technique for obtaining fronted verb phrases in Germanic languages (examples 4). A partial verb phrase has a verb for its head and some number of the verb's complements, but it may not be saturated.[1]

[1]The word "saturated" is used to mean that a head has picked up all of its complements and no longer subcategorizes for any of them in the phrase.

(4) a. **Das Märchen erzählen** wird er ihr.
The fairy-tale[acc] tell will he[nom] to-her[dat]
He will tell her the fairy-tale.

b. **Ihr erzählen** wird er das Märchen.
to-her[dat] tell will he[nom] the fairy-tale[acc
He will tell her the fairy-tale.

(5) **Das Examen bestehen** wird er können.
The exam[acc] pass will he[nom] be-able-to
He will be able to pass the exam.

PVP Fronting is a specific instance of the Complement Extraction Lexical Rule (CELR) of Pollard & Sag (1994) (6):

(6) Simplified Complement Extraction Lexical Rule:

$$\begin{bmatrix}\textbf{synsem}\\ \text{LOCAL:}\begin{bmatrix}\textbf{local}\\ \text{CAT:}\begin{bmatrix}\textbf{category}\\ \text{COMPS:}\langle\ldots,\boxed{1},\ldots\rangle\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

$$\longmapsto$$

$$\begin{bmatrix}\textbf{synsem}\\ \text{LOCAL:}\begin{bmatrix}\textbf{local}\\ \text{CAT:}\begin{bmatrix}\textbf{category}\\ \text{COMPS:}\langle\ldots\ldots\rangle\end{bmatrix}\end{bmatrix}\\ \text{NONLOCAL:}\begin{bmatrix}\textbf{nonlocal}\\ \text{SLASH:}\{\boxed{1}\}\end{bmatrix}\end{bmatrix}$$

The phenomenon of partial verb phrase fronting coupled with the CELR causes some interesting results. The output of the CELR is grammatical in Head-Driven Phrase Structure grammar (HPSG theory). In practice, however, we need to be a bit wary:

- First, we want to curb the application of the rule over and over, potentially infinitely, on entries where the argument structure is largely variable, and the output could easily match the input. This problem is described by Höhle (1995), who suggests suitable restrictions on the ARGSTORE of output to the rule.

- Second, we wonder if our process will terminate. In Prolog, for example, the *concatenate* operation, which relates two lists, e.g. the subcategorization lists of a verb and its governing auxiliary, will go into an infinite search loop if the first or third arguments are variable.

- Third, even for non-auxiliaries, whose arguments are available in the lexicon, we will increase local ambiguity during parsing if our lexicon contains a priori an entry for every word with every possible complement extracted.

Are these real problems during a parsing or generation process? Certainly the CELR could be tailored for a specific case via greater specification of inputs and outputs (as in Meurers 1994), or the processing system could set bounds on the number of rule applications (as in Carpenter & Penn 1994), which provides for limiting the total number of lexical rule applications) in order to curb these problems. However, the former solution will certainly result in a loss of generality, and the second, though easy to implement, is only a partial fix. In operational terms, a lexical argument extraction should only take place once when a single extracted argument is present in SLASH.

In order to use control explicitly in grammar descriptions, yet not be worried unduly with attaching delay statements to lexical or grammar constructs, one can build delays into the description language. The solution we seek is to extend the description language for typed feature structures in Carpenter (1992) to include waits on satisfaction of feature structure descriptions. The waits may occur any place feature structure terms are used, including lexical descriptions, constraint descriptions, or definite clause statements. We will show how delayed rules may also be used. The idea of waiting began with negations in Prolog II Naish (1985) and has been generalized to the paradigm of Constraint Logic Programming.

0.3 Logic Programming

As a parallel exposition to the linguistic literature, I review the field of Logic Programming. I review the methodology for SLD resolution (Robinson 1965), and include a definition for resolution in Constraint Logic Programming (CLP). I introduce the concept of delaying and show guarded rules for CLP. I give an example of guarding from NLP and consider some other approaches to constraint solving. A formal description of feature structures follows. We are then able to show how guarded rules and feature structures can be used together to solve the general problem of processing underspecified linguistic objects.

0.4 Feature Structures and Grammar Processing

A number of feature-based grammar formalisms have been used for computation, including Functional Unification Grammar (FUG) (Kay 1979; Kay 1983; Kay 1985), PATR-II (Shieber *et al.* 1983), Lexical-Functional Grammar (LFG) (Bresnan 1982), Generalized Phrase Structure Grammar (GPSG) (Gazdar *et al.* 1985), and Head-Driven Phrase Structure Grammar (HPSG) (Pollard & Sag 1987; Pollard & Sag 1994). The combining operation for these grammars is unification, which is a method of uniting two terms, or in this case, two feature structures Shieber (1986) provides a good overview of the first formalisms for unification based grammars. Shieber divides these into two categories, the *tool* type (e.g. PATR-II, FUG, DCG (Definite Clause Grammars) (Pereira & Warren 1980)) and the *theory* type (e.g. LFG, GPSG). He reviews the fact that tools for computation generally possess general mechanisms for expression, while theories incorporate "very specific" mechanisms related to a particular type of linguistic analysis. Though tools for processing feature based grammars are undoubtably more sophisticated in the current day than at the time of Shieber's comments, it is worth noting that the tool/grammar distinction still has merit. It is this distinction that we will address. We show that a tool with guarding is appropriate for implementing modern feature based grammars.

I provide an overview of the theory we will use. I follow Carpenter (1992). Carpenter employs a general attribute-value logic, following Rounds and Kasper (Kasper & Rounds 1986; Kasper & Rounds 1990). I then review of the tools that have been developed for processing feature structures, using logic programming. We will be extending one of these, the ALE system (Carpenter & Penn 1994). I start with early systems for feature structure processing, including FUG (Kay 1983; Kay 1985), PATR-II (Shieber *et al.* 1983), and LOGIN (Aït-Kaci & Nasr 1986). I then move on to more recent systems for feature structure processing, including the work of Dörre & Dorna (1993), Erbach *et al.* (1995), Carpenter & Penn (1994), Elhadad *et al.* (1997), Götz *et al.* (1997), and Dörre & Manandhar (1997).

The above systems are frameworks for resolving constraints on feature structures. These have included delay functions, in order for the grammar writer to impose some control on specific feature values. They have generally been restricted to waiting on a specific part of the feature structure, and may be limited to particular operations (e.g. list append). The goal is to allow more freedom with respect to the placement of constraints in the grammar, and to remove infinite resolution chains in particular. I also note the LinGO system (Copestake *et al.* 1999;

Copestake & Flickinger 2000) because this system is among the largest and most widely used system for processing HPSG grammars, although this is not a delaying system.

0.5 Guarded Constraints on Feature Structures

I present two specific ways to describe guarding on feature structures. One is by using *guarded descriptions*, which adds to the expressive power of the description language, but does not affect the rule set. The other is via guarded rules. I provide the operational semantics for guarded constraints on typed feature structures in the context of SLD resolution. I also describe how to write negations such that they are an instance of delaying. This will be important in linguistics when handling constraints which are written as statements of negation, such as inequations.

The specification of a guarded feature structure description from chapter five is given below. The gain in expressive power is that the descriptions themselves become procedural statements whereas the resolution algorithm can remain unchanged.

A guarded description is shown in 7:

(7)

$$\phi \rightarrow \psi; \upsilon \in \mathbf{Desc} \text{ if } \phi, \psi, \upsilon \in \mathbf{Desc}$$

The entailment conditions for guarded descriptions are in 8:

(8) for $\phi, \psi, \upsilon \in$ Desc

$F \models (\phi \rightarrow \psi; \upsilon)$ iff

- $F \models \phi$ and $F \models \psi$ (entailment) OR
- $(\forall F')F \sqcup F' \not\models \phi$ and $F \models \upsilon$ (disentailment)

We define negation as a special case of a guarded description:

(9)

$$\neg(\phi) = (\phi \rightarrow fail;\ true)$$

With the addition of negation, inequations may be implemented as guarded descriptions, or with guarded rules. The implementation of the binding theory in the case of the Japanese causative handles inequations as guarded constraints. The algorithm for resolution with guarded rules is the following:

(10) Given goal $p(f_1, \ldots, f_n)$ and a (renamed) clause $C \mid A' \leftarrow B_1, B_2, \ldots, B_n$, such that goal p, A', C unify with mgu θ, and C_i a constraint on f_i, $C_i = X_i : \phi \in C$:

- If f_i neither satisfies nor dissatisfies ϕ, a *suspension* of a goal $p(f_1, \ldots, f_n)$is created. A suspension is a pointer to the conjunction of goals which are delayed by the constraint. If ϕ is already constrained to satisfy any $p'(f_1, \ldots, f_n)$, then p is added to the conjunction of goals.
- p is wakened if X_i is unified with a feature structure that satisfies ϕ. Then, the action taken is to first execute the suspension of p, and then resume the present goal.

0.6 Implementation and Evaluation

The ALE system (Carpenter & Penn 1994) combines phrase structure parsing and constraint logic programming with typed feature structures as terms. The addition of guarding to this system allows a number of complex linguistic analyses to be processed using guarded descriptions. First I describe a general approach to the implementation of guarding for the case of typed feature structures. Then, I present specific test cases. These are quantifier raising in English and binding in Japanese. The quantifier raising example associates delays with a general type, the type **sign**. This demonstrates that delays may be associated with virtually any or all objects in the domain, by virtue of their types. A trace of the parse of this example is provided. I then move on to a specific instance of binding. Linguistic binding conditions are stated in part negatively. The Japanese example shows the handling of inequations as delays in detail.

The set of examples, including the example of German partial verb phrase fronting, can be evaluated with respect to three criteria. First, are these grammars a faithful rendering of the specification? Second, is the satisfaction algorithm independent of the parser? Third, what are the advantages of using this approach compared with an implementation without guarding?

The grammars are instances of the guarded descriptions in the theory of typed feature structures. They are full and correct implementations of the linguistic theories as written. We try the same guarded grammars with different parsers, and find no differences in the output of the parse, though the constraints have been delayed in different orders. Practical advantages of guarding are that lexical size may be reduced substantially, and grammar rules may streamlined in the direction toward a statement of constraints on more abstract grammatical concepts, such as types, or typed feature structures. Performance improvements are thus tied to gains associated with improvements to the grammar. We look at the alternatives of *brute force* (multiplying out possibilities), *accommodation* (changing the meaning of the program), and hand-threaded rules, comparing and constrasting these methods. In particular, the approach of hand-threading constraints through rules is knowledge-intensive on the part of the grammar writer.

0.7 Conclusions

The data covered in implementation supports the generality of the thesis such that it is extensible to a wide range of linguistic analyses, and is language independent. The specification is generalizable for linguistic analyses of morphologically shared predicates, verb raising, long-distance dependencies, clitics, quantifier raising, and linear order constraints. The reason that all of these phenomena can be unified by the analysis presented is that in all cases, there are arguments which are shared across structures by more than one head. A constraint may exist between a head and one of its arguments, yet another head may project the features which fully instantiate that argument at a later time.

The description of the resolution algorithm is abstracted away from the parsing architecture, providing a practical advantage for implementation efforts. The gain is advancement along a continuum toward the statement of independent constraints and away from the ordering of rules. Another approach, the proliferation of possible instantiations in the case of "brute force," is also eliminated, and gains are made especially by reducing the number of locally ambiguous lexical entries.

Future work should include re-implementing the satisfaction algorithm to maximize efficiency. In a test situation, hand-threading guards in the German grammar decreases processing time vs. the guarded version for sentences with initial auxiliaries, as there is no overhead due to the unblocking and reblocking of constraints. There is no absolute need to trigger a constraint check each time the type variable for a feature structure changes, unless the desired type has been obtained. In other words, the satisfaction conditions are currently constraints that wait on the gain of any new type information by a feature structure. The satisfaction condition itself could be the delay constraint in a more sophisticated implementation. That is, wait on the satisfaction conditions themselves. I would favor that approach.

Bibliography

AÏT-KACI, H., & R. NASR. 1986. LOGIN: A logic programming language with built-in inheritance. *Journal of Logic Programming* 3.185–215.

BRESNAN, JOAN (ed.) 1982. *The Mental Representation of Grammatical Models*. Cambridge, Mass: MIT Press.

CARPENTER, BOB. 1992. *The Logic of Typed Feature Structures with Applications to Unification-based Grammars, Logic Programming and Constraint Resolution*, volume 32 of *Cambridge Tracts in Theoretical Computer Science*. New York: Cambridge University Press.

CARPENTER, BOB, & GERALD PENN. 1994. ALE: The Attribute Logic Engine: User's guide. Laboratory for Computational Linguistics Report CMU-LCL-94, Laboratory for Computational Linguistics, Carnegie Mellon University. Version 2.0.1.

COPESTAKE, ANN, JOHN CARROLL, ROB MALOUF, STEPHAN OEPEN, & OTHERS, 1999. *The (new) LKB system*. CSLI, Stanford University, 5.2 edition. A grammar and lexicon development environment for use with constraint-based linguistic formalisms.

COPESTAKE, ANN, & DAN FLICKINGER. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

DÖRRE, JOCHEN, & MICHAEL DORNA. 1993. CUF – a formalism for linguistic knowledge representation. ESPRIT Basic Research Action 3175, ESPRIT, Institut für maschinelle Sprachverarbeitung, Universität Stuttgart.

DÖRRE, JOCHEN, & SURESH MANANDHAR. 1997. On constraint-based lambek calculi. In *Specifying Syntactic Structures*, ed. by Blackburn & de Rijke, Studies in Logic, Language and Information. Stanford: CSLI Publications.

ELHADAD, MICHAEL, KATHLEEN MCKEOWN, & JACQUES ROBIN. 1997. Floating constraints in lexical choice. *Computational Linguistics* .

ERBACH, G., M. VAN DER KRAAN, S. MANANDHAR, H. RUESSINK, W. SKUT, & C. THIERSCH. 1995. Extending unification formalisms. In *Proceedings of the 2nd Language Engineering Convention*, London. to appear.

GAZDAR, GERALD, EWAN KLEIN, GEOFFREY K. PULLUM, & IVAN A. SAG. 1985. *Generalized Phrase Structure Grammar*. Oxford: Basil Blackwell.

GÖTZ, THILO, DETMAR MEURERS, & DALE GERDEMANN, 1997. *The ConTroll Manual (ConTroll v.1.0β, XTroll v.5.0β)*. Seminar for Sprachwissenschaft, Universität Tübingen.

HÖHLE, TILMANN, 1995. CELR and variable argument raising. Presented at the HPSG Workshop, Tübingen, June 22, 1995. Unpublished ms.

KASPER, ROBERT T., & WILLIAM C. ROUNDS. 1986. A logical semantics for feature structures. In *Proceedings of the Twenty-Fourth Annual Meeting of the ACL*, 235–242. Association for Computational Linguistics.

—— 1990. The logic of unification in grammar. *Linguistics and Philosophy* 13.35–58.

KAY, MARTIN. 1979. Functional grammar. In *Proceedings of the 5th annual meeting*, Berkeley, CA. Berkeley Linguistic Society.

——. 1983. Unification grammar. Technical report, Xerox Palo Alto Research Center.

——. 1985. Parsing in functional unification grammar. In *Natural Language Parsing: Psychological, Computational and Theoretical Perspectives*, chapter 7, 251–278. Cambridge, England: Cambridge University Press.

MEURERS, DETMAR, 1994. On implementing an HPSG theory: Aspects of the logical architecture, the formalization and the implementation of head-driven phrase structure grammars. Master's thesis, Universitt Tbingen. Appeared as Part 2 of SFB Report 58.

NAISH, L., 1985. *Negation and Control in Prolog*. University of Melbourne dissertation.

PEREIRA, F. C. N., & D. H. D. WARREN. 1980. Definite clause grammars for language analysis: A survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence* 13.231–278.

POLLARD, CARL, & IVAN A. SAG. 1987. *Information-based Syntax and Semantics, Vol. 1*. Number 13 in Lecture Notes. Stanford University: CSLI Publications. Distributed by University of Chicago Press.

——. 1994. *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

ROBINSON, J. A. 1965. A machine-oriented logic based on the resolution principle. *Journal of the ACM* 12.23–41.

SHIEBER, S. M., H. USZKOREIT, F. C. N. PEREIRA, J. ROBINSON, & M. TYSON. 1983. The formalism and implementation of PATR-II. Technical report, SRI International, Menlo Park, California.

SHIEBER, STUART. 1986. *An Introduction to Unification-Based Approaches to Grammar*. Number 4 in Lecture Notes. Stanford University: Center for the Study of Language and Information.