# Syntactic and Semantic Knowledge in the DELPHI Unification Grammar

**R. Bobrow**
**Robert Ingria**
**David Stallard**

BBN Systems and Technologies Inc.
10 Moulton Street, Mailstop 6/4C
Cambridge, MA 02138

## Abstract

This paper presents recent natural language work on HARC, the BBN Spoken Language System. The HARC system incorporates the Byblos system [6] as its speech recognition component and the natural language system Delphi, which consists of a bottom-up parser paired with an integrated syntax/semantics unification grammar, a discourse module, and a database question-answering backend. The paper focuses on the syntactic and semantic analyses made in the grammar.

## Introduction

This paper discusses the unification-based grammar component of the BBN Delphi system, the natural language component of the BBN Spoken Language System. This grammar, previously discussed in [4][5], is written in a variant of the Definite Clause Grammar formalism [17] and simultaneously assigns both syntactic structure and semantic interpretation to an utterance. The grammar comprises some 1300 rules (including those for terminal items), and has recently been evaluated on the ATIS air travel information domain with favorable results [2].

The paper has two main purposes. The first is to present this grammar's treatment of a number of syntactic and semantic phenonomena in the ATIS domain, in the belief that comparison of analyses are useful in and of themselves, particularly in the context of a common evaluation. In particular, we discuss the constraints imposed by subgrammars on conjunction, relaxation of subject verb agreement, optional argument subcategorization, and semantic interpretation of PPs and nominal compounds.

The second purpose is to argue for the necessity of a device which is frequently made use of Definite Clause Grammar work, but which has less often been utilized in the work on unification grammar that has appeared more recently ([19] [14]). This is the right-hand side "constraint relation" that does not derive any constituent of the utterance, but rather serves to constrain in various ways the feature assigments of those rule elements which do. Constraint relation elements are thus to be distinguished from other non-terminals which derive empty constituents, such as gaps or zero morphemes.

While constraint relations have long been used in DCG-based work to treat, among other things, problems of quantifier scoping and the interaction of quantifer scope and anaphora [16] [18] we argue that there are other, more low-level issues of lexical semantics and syntax for which constraint relations are either conceptually useful or formally necessary, even in unification formalisms which allow full disjunction across features, such as [10] [12] [11] [9].

The remainder of the paper is divided into sections. In the next section, we first briefly review the formalism we use. In the two sections after that, we present syntactic analyses and semantic analyses made by the Delphi grammar, with emphasis placed on the use of constraint relation elements. Finally, in the last section we discuss the question of whether or not the constraint elements can be compiled out of grammar rules.

## Grammar Formalism

The Delphi grammar formalism is a variant of Definite Clause Grammar and is discussed in more detail in [4][5]. Left- and right-hand side rule elements have a functor (their major category) and zero or more features in a fixed-arity, positional order. Features are slots which are filled by terms. Terms can either be variables, which are atoms prefixed by colons ":", or functional terms consisting of a functor and zero or more features, again in fixed positional order, occupied by other terms. Note that this means that constants in Delphi are just nullary functions. Disjunctions of purely atomic values are allowed as terms, with the functor ":OR".

The following is an example rule, much simplified for expository purposes, for handling NP conjunctions:

```
(NP (AGR :P (PLURAL)) (REALNP :REALZ))
→
(NP (AGR :PERSONX :NUMX) (REALNP :REALX))
(CONJ)
(NP (AGR :PERSONY :NUMY) (REALNP :REALY))
(P-MIN :PERSONX :PERSONY :P)
(NPTYPE-FILTER :REALX :REALY :REALZ)
```

P-MIN and NPTYPE-FILTER are constraint relations whose function will be discussed in the next section.

This formalism maintains the term unification practice of using functors with obligatory, positional arguments, rather than functorless feature structures with optional, labelled arguments, as in much recent work. There are several reasons for this. First of all, we find the functor very useful as a way of indicating just what features are allowed in a given structure (and it is interesting to note attempts to restore it for just this purpose in [13].) Second, argument labels contribute their own clutter to the rule. They would thus seem to be a notational win only if more than half the features of a given element are "don't-cares". In our grammar, however, we find on average that only about 3% of the feature slots in rules are "don't-cares".

We should emphasize, however, that none of the work presented here hinges on these notational choices, and all the points made in subsequent sections carry over just as well to other unification notations.

## Syntactic Constraint Analyses

Restrictions on agreement between features sometimes depend on the values of still other features. For example, English noun phrase conjunctions require that if one of the conjuncts belongs to a particular subgrammar (date, time etc.) then the other conjunct(s) must also belong to that subgrammar, but if neither of the conjuncts belongs to a subgrammar, there is no restriction. This can be done by including in the NP conjunction rule a constraint relation which provides a case-analyzing effect that could not be provided merely by unifying the variables ranging over the semantic type. This function is carried out by NPTYPE-FILTER, which we introduced above. This relation takes as its arguments the NPTYPE feature of the first conjunct, the NPTYPE of the second conjunct, and the NPTYPE feature of the conjunction as a whole. This relation, then, can be thought of as constraining the possible triples of these values.[1] We present four instances of this rule for illustration, where "$\rightarrow \emptyset$" indicates production of the empty string.

```
(NPTYPE-FILTER (NONUNITNP (−PRO (TIMENP)))
               (NONUNITNP (−PRO (TIMENP)))
               (NONUNITNP (−PRO (TIMENP))))
               → ∅
(NPTYPE-FILTER (NONUNITNP (−PRO (DATENP)))
               (NONUNITNP (−PRO (DATENP)))
               (NONUNITNP (−PRO (DATENP))))
               → ∅
(NPTYPE-FILTER (NONUNITNP (−PRO (MISCNP)))
               (NONUNITNP (+PRO :PRO-TYPE))
               (NONUNITNP (−PRO (MISCNP))))
               → ∅
(NPTYPE-FILTER (NONUNITNP (+PRO :PRO-TYPE))
               (NONUNITNP (−PRO (MISCNP)))
               (NONUNITNP (−PRO (MISCNP))))
               → ∅
```

The first two rules require that if either of the conjuncts of a conjoined NP belongs to the TIMENP or DATENP subgrammar, than the other conjunct must belong to that subgrammar, as well. The last two rules allow ordinary pronominal and non-pronominal NPs to conjoin freely.[2] In this example, note that while the first two rules could be collapsed into a single rule utilizing unifying variables, the third and fourth cannot.

Another use of constraint relations is to "compute" a value from the feature values of the relevant consitutents, rather than requiring identity of features. For example, in NP conjunction with "and" in English, the person of the conjoined NP is first person if any of the conjuncts is first person, second person, if any of the conjuncts is second person and none is first person, and third person if all the conjuncts are third person.[3] This can be easily handled by the P-MIN constraint relation, which takes as its arguments the PERSON feature of the first conjunct, the PERSON of the second conjunct, and the PERSON feature of the conjunction as a whole. It has the following solutions:

```
(P-MIN (1ST) :P (1ST)) → ∅
(P-MIN (3RD) :P :P) → ∅
(P-MIN (2ND) (1ST) (1ST)) → ∅
(P-MIN (2ND) (3RD) (2ND)) → ∅
(P-MIN (2ND) (2ND) (2ND)) → ∅
```

Still another case in which constraint relations provide a kind of flexibility greater than that available using standard unification is to allow the grammar to express "degrees of grammaticality". For example, in standard written English, it is common for a verb to agree in number with its subject.[4]

What do the restrictions represent?
What does restriction VU/1 mean?
What do the transport codes AL and R mean?

However, in spoken English, conjoined noun phrases sometimes appear with singular agreement on the verb.

What does RETURN MIN and RETURN MAX mean?
What does class B and class Y mean?

In still looser speech, agreement disappears even with non-conjoined subject noun phrases:

List all the airlines that flies from Dallas to Boston nonstop.

These facts can be handled by modifying the standard sentence rule:

```
(ROOT-S :MOOD)
→
(NP :AGR)
(VP :AGR :MOOD)
```

to the following:

```
(ROOT-S :MOOD)
→
(NP :AGR :CONJC)
(VP :AGRX :MOOD)
(SUBJECT-VERB-AGREEMENT :AGR :AGRX :CONJC)
```

[1]Alternatively, it can be viewed as computing a value for the NPTYPE feature of the conjunction from the values of the individual conjuncts.

[2]They also set the NPTYPE of the conjunction to being non-pronominal, since it cannot function as a pronoun.

[3]Karttunen [10] handles such cases with a procees of generalization, rather than unification.

[4]All of the examples in this discussion of subject-verb agreement are taken from the ATIS corpora collected by TI and distributed by NIST.

and adding the following solutions for SUBJECT-VERB-AGREEMENT. This is a constraint relation that takes as its arguments the agreement feature of the sentence's subject NP, the agreement feature of the sentence's VP, and the conjunction feature of the subject NP—which indicates whether it is a conjunction or not.

```
(SUBJECT-VERB-AGREEMENT (AGR :P :N)
                        (AGR :P :N)
                        :CONJ)
                        → ∅
(SUBJECT-VERB-AGREEMENT (AGR :P (PLURAL))
                        (AGR :P (SINGULAR))
                        (+CONJ :CONJTYPE))
                        → ∅
(SUBJECT-VERB-AGREEMENT (AGR :P (PLURAL))
                        (AGR :P (SINGULAR))
                        (−CONJ))
                        → ∅
(SUBJECT-VERB-AGREEMENT (AGR :P (SINGULAR))
                        (AGR :P (PLURAL))
                        (−CONJ))
                        → ∅
```

The first of these solutions enforces standard subject verb agreement: whether the subject NP is a conjunction or not, the agreement features of the subject and the VP must be the same. The second allows the VP to bear the singular feature when the subject is plural, just in case the subject is a conjunction. The last two rules allow the subject and the VP to disagree, when the NP is not a conjunction. The statement about the conjunction status of the subject is necessary in these last two solutions to make them orthogonal to the first two, so that a single structure will not be unnecessarily analyzed with more than one solution.

This mechanism is superior to simply not requiring a VP to agree with its subject at all, by using distinct variables for the agreement features of the subject NP and the VP, since, given a large enough corpus, we can automatically associate different probabilities with the different solutions of this constraint relation. This is particularly useful in a spoken language system, since this will allow all the possibilities, with some degree of probability, but will always prefer the most common solution and will only choose a less likely solution if a more common one is unavailable.

## Semantic Constraint Analyses

The facility for case analysis is also used in semantic interpretation, where the meaning representation of constructions are computed in terms of values of certain features which cannot always be known in advance.

### PP Interpretation

Prepositional phrases, in both post-copular and post-nominal positions, are very common in the ATIS domain (and most other domains as well). Some examples:

Which flights are on Delta Airlines
Which flights are on Thursday
Which flights are after 4 pm

The rule in our grammar which generates a post-copular PP is the following:

```
(VP :SUBJ :WFF)
→
(V (BE))
(PP :PP)
(PREDICATIVE-PP :PP :SUBJ :WFF)
```

PREDICATIVE-PP is the constraint relation in the rule. It is responsible for specifying the formula meaning of the VP in terms of the translation of the PP (:PP) and the translation of the subject passed down from the clause (:SUBJ).

The PREDICATIVE-PP solution for the "flight-on-airline" sense is as follows:

```
(PREDICATIVE-PP (PP-SEM (:OR (ON)(ABOARD)
                             (ONBOARD))
                        (:NP AIRLINE))
                (:SUBJ FLIGHT)
                (EQUAL (FLIGHT-AIRLINE-OF :SUBJ)
                       :NP))
```

The first occurences of the variables :NP and :SUBJ above are paired with semantic types AIRLINE and FLIGHT; this is shorthand for the actual state of affairs in which a term representing a package of information (including encoding of semantic type) appears in the slots of the rule these variables occur in. This term carries quantification and semantic type information, as below:

```
(Q-TERM ⟨QUANTIFIER⟩ ⟨VARIABLE⟩
        (NOM ⟨PARAMETER⟩ ⟨SET⟩ ⟨SORT⟩))
```

This structure is so constructed as to not unify with another such structure if its semantic type is disjoint, using a method for encoding semantic types as terms described in [20].

The PP translation is also a package with the functor PP-SEM, containing the preposition and the translation of the NP object of the PP. No local attempt is made to translate the preposition.

When parsing with the above predicate PP rule, the system searches through a database of PREDICATIVE-PP solutions like the above, much as a PROLOG-based system would. If a solution succesfully unifies, the formula is passed up as the translation of the VP. Recursion is allowed, as in:

```
(PREDICATIVE-PP (PP-SEM :PREP (:NP TIME))
                (:SUBJ FLIGHT)
                :WFF)
→
(PREDICATIVE-PP (PP-SEM :PREP (:NP TIME))
                (DEPARTURE-TIME-OF :SUBJ)
                :WFF)
```

This rule says that any PP relating a flight to a time should be translated as if it related the departure time of the flight to that time. In this way, a common system of PREDICATIVE-PP solutions stipulating the meanings of various prepositional comparisons between times ("after", "before", "on", "during") can be specified just once, and used in multiple contexts.

Such a method for interpreting PPs can be compared to one that uses unification over features of constituent elements instead of constraint relations, such as proposed in [13] and [8]. There, the multiple meanings of a preposition are enumerated locally as a semantic feature of the preposition, using a form of full-scale disjunction. Type constraints filter out those which are not meaningful in the given context, just as in our work.

We claim, however, that the constraint relation method is superior exactly in view of its recursive power as described above. In order to handle such constructions as "flight after 4pm", "flight before 4 pm" etc. a strictly unification-based approach would have to compile out all the possibilities for the type of the clause subject (TIME, FLIGHT etc.) and store them as disjunctive values of the translation feature on the given preposition. Thus in the ATIS domain there would be (at least) two senses of the preposition "after", both expressing the "time-after" relation. Clearly such an approach would not capture the general relations on times that these prepositions express.

## Subcategorization

One of the devices used by the DELPHI grammar for encoding subcategorization is derived from GPSG [7]. This is to place a feature for subcategorization on the rule for a given verb, and key all VP rules off this feature:

```
(VP :SUBJ :WFF) ;for "John kicked Mary"
→
(V (TRANSITIVE :WFF :SUBJ :OBJ))
(NP :OBJ)
```

The semantics of the subject of the sentence is passed down through the :SUBJ variable to the V, along with the semantics of the complements. The V in turn passed back up the formula representing the semantics of the whole sentence, through the :WFF variable.

Of course this mechanism requires one VP rule for every subcategorization frame one wants to handle, and this can run to many rules (about 60 in the DELPHI grammar). A more serious problem, however, arises in the case of optional complements to verbs, as seen in the following actual ATIS training sentences that use the verb "arrive":

Show me all flights from Boston to Denver that arrive
 before 5 PM
Show me flights ... that arrive in Baltimore before noon
Show me all the nonstop flights arriving from Dallas
 to Boston by 10 PM ...
Show me flights departing Atlanta arriving San Francisco
 by 5 PM
Show me flights arriving into Atlanta by 10 PM from Dallas

We see here that, in addition to the temporal PP that always accompanies it, "arrive" can be followed by (1) nothing else, (2) a PP with "in", (3) a "from"-PP and a "to"-PP, (4) a bare noun phrase, or (5) an "into"-PP and a "from"-PP. The principle pattern that emerges is one of complete optionality and independence of order. Indeed, in the fifth example, the temporal PP, which might be more traditionally regarded as an adjunct rather than a complement, and thus as one of the siblings of the VP rather than one of its constituents, is instead interposed between two PPs complements, making the adjunct analysis rather problematic.[5]

The only way the subcategorization scheme presented above could deal with these variations would be to enumerate them all in separate rules. But this would clearly be infeasible. The solution we have adopted constructs a right-branching tree of verbal complements, where the particular constituents admitted to this tree are controlled by constraint relations keying off the lexical head of the verb. There are two main rules:

```
(VP :SUBJ (AND :INITIAL-WFF :COMP-WFF))
→
(V :LEX (INTRANSOPTCOMPS :SUBJ
                         :INITIAL-WFF))
(OPTCOMPS :LEX :SUBJ (DUMMY) :COMP-WFF)
```

and

```
(VP :SUBJ (AND :INIIIAL-WFF :COMP-WFF))
→
(V :LEX (TRANSITIVEOPTCOMPS :SUBJ
                            :OBJ
                            :INITIAL-WFF))
(NP :OBJ)
(OPTCOMPS :LEX :SUBJ :OBJ :COMP-WFF)
```

The category OPTCOMPS generates the right-branching tree of optional complements. It has the rules

```
(OPTCOMPS :LEX :SUBJ :OBJ (AND :WFF1 :WFF2))
→
(PP :PP)
(OPTCOMPS :LEX :SUBJ :OBJ :WFF1)
(OPTCOMP-PP :LEX :SUBJ :OBJ :PP :WFF2)
```

and

```
(OPTCOMPS :LEX :SUBJ :OBJ (TRUE)) → ∅
```

The OPTCOMP-PP is the constraint relation; it keys off the lexical head of the verb (the variable :LEX) and combines the subject, object, and PP complement translations to produce the contribution of the PP complement to the final formula that represents the sentence meaning. An arbitrary number of PP complements are provided for by the recursion of the first rule above, which bottoms out in the case of the second rule when there are no more complements. Phrasal types other than PPs are accomodated by similar rules.

The solution for PP complements to "arrive" such as "in Atlanta", "into Baltimore" "at Denver" etc. follows:

[5]To see that these PPs are truly associated with the verb rather than somehow modifying the subject flight-NP, one need only replace the subject with the pronoun "it".

```
(OPTCOMPS (ARRIVE)
          (:SUBJ type FLIGHT) :ANY
          (EQUAL (DESTINATION-CITY :SUBJ) :NP))
→
(PP (PP-SEM (:OR (INTO) (IN) (AT)) (:NP type CITY)))
```

This rule says that for a flight to "arrive" INTO, IN or AT a city means that the city equals the value of the flight's DESTINATION-CITY attribute. Semantic type information is here notated with a shorthand keyword "type"; in the actual system a partially-specified term that packages semantic type information in a specific slot is unified into such variables as :SUBJ, :OBJ and :NP. Note also the use of disjunction (the :OR) to combine different prepositions together.

Other devices for encoding subcategorization have been proposed. One, made use of in PATR-II and described in [19] encodes subcategorization information as a list of the complement patterns that are required to follow the verb, and generates a right-branching tree of VPs, each absorbing one of the complements in the list until no more are left. The difference between the PATR-II scheme and the one way presented here is that the complements in PATR-II are *required* and must follow one another in a predetermined order. The only obvious way it could handle the optionality and order independence seen in the "arrive" examples is just the same brute-force method of enumerating possiblities, using lists of differing order and length.

## Using Unification to Encode Semantic Constraints beyond Semantic Type

Not all constraints on meaningfulness are strictly reflections of the semantic type of phrase denotations. Consider the lexical item "L" in the ATIS training set. Seen in a number of different database fields, it can variously denote limousine availablity, lunch service, or other classes of service available on a flight. Yet in the following example it is clear that its usage is relevant to just the first of these:

What is transport code L?

Our claim is that not just the referent of "L"—limousine service for ground transportation—that plays a role here, but also the means by which it gets to that referent: namely, by being an abbreviation or code rather than a name. That is, "transport code L" is a meaningful compound, while "transport code limousine" would not be. The lexical entry for abbreviation terms like "L" reflects this by taking the form of an inverse function application: the referent of the lexical item "L" is the ground transportation type that has the string "L" as its abbreviation:

```
(INVERSE-VAL* (ABBREV-OF) "L"
              (GROUND-TRANSPORTATION))
```

While this has the same referent as the lexical entry for "limousine" it has a different form, one which the rule analyzing the construction above makes use of.

Nominal compounds in the DELPHI system are generated by the following rule:

```
(N-BAR :NOM3)
→
(N-BAR :NOM1)
(N :NOM2)
(NOM-COMP-READING :NOM1 :NOM2 :NOM3)
```

in which the constraint relation NOM-COMP-READING computes the semantics of the whole construction from the semantics of the head houn and the nominal modifier. NOM-COMP-READING has different solutions for different semantic types of noun translation. The relevant one here is:

```
(NOM-COMP-READING
    (NOM (CONS-P (REL1)
                 (:ARG type :ARG-SORT)
                 :PARS2)
             (REL-APPLY* :FUNCTION :NP)
                 :VAL-SORT)
    (NOM :PARS1 (INVERSE-VAL* :FUNCTION
                              :TERM1 :SET)
        :ARG-SORT)
    (NOM :PARS1 (INVERSE-VAL* :FUNCTION
                              :TERM1 :SET)
        :ARG-SORT)) → ∅
```

The first slot of the NOM terms above encodes the argument-taking property of relational nouns such as "code", "salary" or "speed", and has been described in an earlier paper [20]. The rule states that an inverted attribute reference (here, "L") preceded by a relational noun (here, "code") for that same attribute (here, "ABBREV-OF") simply refers to that inverted attribute reference.

Our view is that the preceding nominal modifier essentially performs a function of disambiguation: it serves to distinguish the desired sense of the head from any other possible one. This is reinforced when the whole compound—"transport code L"—is considered. Another NOM-COMP-READING rule is responsible for combining "code" with "transport":

```
(NOM-COMP-READING
    (NOM :PARS1 :ARG-SET :ARG-SORT)
    (NOM (CONS-P (REL1)
                 (:ARG type :ARG-SORT)
                 :PARS2)
        (REL-APPLY* :FUNCTION :NP)
        :VAL-SORT)
    (NOM (CONS-P (REL1)
                 :ARG
                 :PARS2)
        (REL-APPLY* (F-RESTRICT :FUNCTION
                                :ARG-SET)
                    :NP)
        :VAL-SORT)) → ∅
```

This constrains the domain of the relational noun it modifies to be just the set that is the translation of the modifying noun. The semantic type of the modifying noun must be unifiable with the argument type of the head relational noun. After this unification forms the translation of the compound "transport code", the resulting type constraints serve to distinguish the correct sense of "L"—that of ground transportation—from the meal service and other senses.

Our technique of allowing rules to impose restrictions on the forms of semantic translations themselves, rather than merely on the semantic types of this translations, bears some discussion because it differs from proposals made by others, such as [14]. In that work, the position is taken that inspecting or restricting the structure of a logical form is inadmissible on theoretical grounds, in that it violates or makes unenforceable the principle of compositionality. As far as theoretical matters go, we believe that the principle of compositionality is open to many interpretations (for example, see [15]) and that its most general interpretation does not exclude techniques such as ours.

A more practical concern, and one which may well underly or justify the theoretical qualm, is that rules based on the structure of logical form may not succeed if that structure happens to be transformed (say through wrapping with another structure) into some syntactically different form which cannot be recognized as one which the rule should allow.

This is not a problem for the rules presented in this section, since the operation of nominal compounding is so tightly localized, operating in a function-argument fashion that gets to the noun meaning "first", before other modifications such as postnominal adjuncts. Ultimately, though, we feel that the real solution must lie in a different approach to meaning representation, one in which non-denotational properties of the utterance meaning are encoded or highlighted in a way that stands above the variances of syntactic logical form. But this is a matter for future research.

## Can Constraint Relations be Compiled out of Rules?

A natural question to ask is whether or not constraint relations can be compiled out of grammar rules, turning them into unifications like any other over the features of "real" constituent elements.

One technique for this kind of compilation is certainly well-known in logic programming: the method of partial execution [16] in which the constraint relation is solved for at the time the rule is read in, not when it is used. The solution(s) so obtained are simply unified back into the remainder of the rule to create the compiled version. If the constraint relation is non-deterministic there will be more than one solution and hence more than one compiled version of the rule. This is certainly undesirable computationally, since multiple variants of the the same grammar rule will cause a parser to do redundant matching. On the other hand, if the unification method used permits full disjunction the variation can be kept "factored", and the redundant matching avoided.

Such a compilation will be possible, obviously, only if the constraint relation can be guaranteed to have finite number of solutions, given the degree of instantiation in which it appears in the rule. But this is actually a rather strong condition to place on constraint relations, particularly on those which participate in semantic interpretation. Semantic strategies which defer part of the semantic computation for a constituent can have a problem with this condition, since there is no bound (given conjunction) on the size of most types of constituents. An example in our work would be the treatment of PP semantics outlined earlier, in which the PP is not fully translated at its own level, but instead passed up as a package of preposition and NP translation. The obvious extension for a conjoined PP would be as list of such structures. But since there is no limit to the number of PPs that can be conjoined, there is no limit to the length of the list, and thus there can be no limit to the number of solutions for the attachment constraint.

Even if a finite number of solutions can be guaranteed, however, there are still reasons why one might not *want* to compile constraint conditions out of the grammar. One reason is that leaving them in allows one to add solutions to a constraint condition without re-compiling all the grammar rules and other constraint rules which use it (a situation exactly analogous to the macro/function distinction in Lisp programming). Another is that distinguishing certain constraints from the rest of the unifications of the grammar rule enables us to intervene in the search with specific indexing methods (say on semantic type). This is discussed in [3]. And a final benefit of distinguishing constraint conditions from other unifications is that relative frequency counts can more easily be made of their different solutions, just as can be made on semantic senses of lexical items. This is discussed in [1].

## Acknowledgements

The work reported here was supported by the Advanced Research Projects Agency and was monitored by the Office of Naval Research under Contract No. N00014-89-C-0008. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

## References

[1] Ayuso, D., Bobrow, R., MacLaughlin, D., Meteer, M., Ramshaw, L., Schwartz, R., and Weischedel, R. "Towards Understanding Text with a Very Large Vocabulary", this volume.

[2] Bates, M., Bobrow, R., Boisen, S., Ingria, R., and Stallard, D. "BBN ATIS Progress Report—June 1990", this volume.

[3] Bobrow, R. and Ramshaw, L. "On Deftly Introducing Procedural Elements into Unification Parsing", this volume.

[4] Boisen, S., Chow, Y., Haas, A., Ingria, R., Roucos, S., Scha, R., Stallard, D., and Vilain, M. *Integration of Speech and Natural Language: Final Report.* Report No. 6991, BBN Systems and Technologies Corporation, Cambridge, Massachusetts, 1989.

[5] Boisen, S., Chow, Y-L., Haas, A., Ingria, R., Roukos, S.; and Stallard, D. "The BBN Spoken Language System". *Proceedings of the Speech and Natural Language Workshop February 1989*, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989, pages 106–111.

[6] Chow, Y.L., Dunham, M.O., Kimball, O.A., Krasner, M.A., Kubala, G.F., Makhoul, J., Price, P.J., Roucos, S., and Schwartz, R.M. "BYBLOS: The BBN Continuous Speech Recognition System". *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Dallas, TX, April 1987, pp. 89–92, Paper No. 3.7.

[7] Gazdar, G., Klein, E., Pullum, G., and Sag, I. *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge, Massachusetts, 1985.

[8] Halvorsen P.-K. and Nerbonne, J. "Unification in the Syntax/Semantics Interface", tutorial presented at the 28th Annual Meeting of the Association for Computational Linguistics, Pittsburgh, Pennsylvania, 6 June, 1990.

[9] Johnson, M. E. *Attribute-Value Logic and the Theory of Grammar.* Center for the Study of Language and Information, 1989.

[10] Karttunen, L. "Features and Values". *Proceedings of Coling84*, Association for Computational Linguistics, Morristown, NJ, 1984, pp. 28–33.

[11] Kasper, R. T. "A Unification Method for Disjunctive Feature Descriptions". *25th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, 1987, pp. 235–242.

[12] Kasper, R. T. and Rounds, W. C. "A Logical Semantics for Feature Structures". *24th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, 1986, pp. 257–266.

[13] "Moens, M., Calder, J., Klein, E., Reape, M., and Zeevat, H. "Expressing generalizations in unification-based grammar formalisms". *Fourth Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, 1989, pp. 174–181.

[14] Moore, R. C. "Unification-Based Semantic Interpretation". *27th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, 1989, pp. 33–41.

[15] Partee, B. H. "Compositionality". *Varieties of Formal Semantics: Proceedings of the fourth Amsterdam Colloquium, September, 1982*, F. Landman and F. Veltman (eds.), FORIS PUBLICATIONS, Dordrecht - Holland/Cinnaminson - U.S.A., 1984, pp. 281–311.

[16] Pereira, F. C. N. and Shieber, S. M. *Prolog and Natural-Language Analysis.* Center for the Study of Language and Information, Stanford, CA, 1987.

[17] Pereira, F. C. N. and Warren, D. H. D. "Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks". *Artificial Intelligence* 13, 1980, pp. 231–278.

[18] Pollack, M. E. and Pereira, F. C. N. "An Integrated Framework for Semantic and Pragmatic Interpretation". *26th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, 1988, pp. 75–86.

[19] Shieber, S. M. *An Introduction to Unification-Based Approaches to Grammar.* Center for the Study of Language and Information, Stanford, CA, 1986.

[20] Stallard, D. "Unification-Based Semantic Interpretation in the BBN Spoken Language System". *Proceedings of the Speech and Natural Language Workshop October 1989*, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989, pages 39–46.