UNIFICATION-BASED SEMANTIC INTERPRETATION IN THE BBN SPOKEN LANGUAGE SYSTEM

David Stallard

BBN Systems and Technologies Corporation
Cambridge, MA 02138

ABSTRACT

This paper describes the current state of work on unification-based semantic interpretation in HARC (for Hear and Recognize Continous speech) the BBN Spoken Language System. It presents the implementation of an integrated syntax/semantics grammar written in a unification formalism similar to Definite Clause Grammar. This formalism is described, and its use in solving a number of semantic interpretation problems is shown. These include, among others, the encoding of semantic selectional restrictions and the representation of relational nouns and their modifiers.

1 INTRODUCTION

Over the past year work on semantic interpretation in the BBN Spoken Language System has shifted from a Montague Grammar (Montague, 1973) style rule-for-rule approach to one which attempts to carry out semantic interpretation directly in the unification grammar rules themselves. This is accomplished by adding semantic features to the grammar rules, placing them on the same footing as the existing syntactic features. Meaning representations are thereby constructed, and semantic filtering constraints applied, as part of parsing the utterance.

We view such a move as having essentially three advantages:

- more information is available to semantic interpretation, so it is possible to gain higher coverage
- syntax and semantics are integrated, so semantic filtering constraints can be applied as constituents are built and attached
- this integration is simple and does not require any complex engineering of cooperating software modules

All three of these advantages are important ones for a spoken language system.

The HARC system has the following overall organization. Spoken input is initially analyzed by the "N-best" algorithm(Chow and Schwartz, 1989), converting it into a rank-ordered set of N best word-sequence hypotheses (for a given value of N). These N hypotheses are then analyzed by the parser, using the combined syntactic/semantic grammar. Those hypotheses which are syntactically and semantically allowed emerge from the parser as initial logical forms in which quantifiers are interpreted "in place". Next, the quantifier module assigns scopes and passes the translation to the anaphora component, which then resolves the referent of intra- and extra-sentential pronouns. The completed logical form is then passed to the back-end component whose responsibility is to compute the appropriate response to the user's input.

The present paper confines itself to a description of the combined syntactic/semantic grammar, along with some discussion of the parsing algorithm. We will first consider the representational framework in which the grammar is written.

2 THE GRAMMAR FORMALISM

The BBN grammar formalism, described in detail in (Boisen *et al.*, 1989), is most closely related to Definite Clause Grammar (Pereira and Warren, 1980). Rules consist of a single left-hand term and zero or more right hand terms. Terms can have features, whose values are themselves terms. Variables, indicated by the ":" prefix, indicate identity between different slots in and among terms.

Here is an example of a simple grammar rule written in this formalism:

```
(VP (AGR :P :N) :MOOD (WH-) :TRX :TRY)
→
(V :CONTRACT (TRANSITIVE) :P :N :MOOD)
(NP :NSUBCATFRAME (WH-) :TRX :TRY)
```

This rule says that a VP can derive a transitive verb followed by an object NP. The person, number and mood of the VP — indicated by the variables :P, :N and :MOOD respectively — must be the same as the person, number and mood of the verb.

The distinguishing feature of the BBN formalism, in which it differs from DCG, is its strict system of typing. Each functor that can head a term is associated with a type and a fixed set of argument positions that are also typed. For example, the functor "AGR" has the type AGREEMENT, and the argument types PERSON and NUMBER. Variables are also typed; for example the variable :P has the type PERSON and the variable :N has the type NUMBER. A given grammar written in the formalism, then, has two components: a set of functor type declarations and a set of grammar rules. This typing feature enables the the grammar rules to be statically checked against the type declarations. A large class of errors in the grammar — such as accidentally ommitted features, transpositions or mispellings — can be caught when the grammar is loaded into the computer instead of through run-time debugging. This capability has proven very useful indeed in the course of creating and modifying a large (≈800 rule) grammar.

3 BASIC EXAMPLES

We now present a very simple example of the use of semantic features in unification, adding semantic features to the VP rule considered earlier. These new features are underlined:

```
(VP (AGR :P :N) :MOOD (WH-)
    :TRX :TRY :SUBJ :WFF)
→
(V (TRANSITIVE :WFF :SUBJ :OBJ)
    :P :N :MOOD)
(NP :NSUBCATFRAME (WH-) :TRX :TRY :OBJ)
```

This rule passes up a formula as the semantics of the VP, indicated by the variable :WFF. The semantics of the subject of the clause, indicated by the variable :SUBJ, is passed down to the verb, as is the semantics of the object NP, indicated by the variable :OBJ.

For the transitive verb "hire", we have the following lexical rule:

```
(V (TRANSITIVE (HIRE' :SUBJ :OBJ) :SUBJ :OBJ)
   :P :N :MOOD)
→
(hire)
```

We can think of this rule in functional terms as taking semantic arguments :SUBJ and :OBJ and returning as a value the wff (HIRE' :SUBJ :OBJ). Note the placement of semantic arguments to the verb inside the subcategorization term (headed by the functor "TRANSITIVE") instead of at the top-level of the "V". This means that a verb with a differing number of arguments, such as "give", has a different subcategorization functor with a corresponding number of argument places for the semantic translations of these arguments.

Like Definite Clause Grammar, the BBN formalism does not require that every term on the right-hand side of the rule derive a non-empty string. This is neccesary to handle traces. But empty-deriving terms are also made use of in some grammar rules as so-called "constraint nodes". These do not generate a real constituent of the parse tree, but instead stipulate that a particular relationship hold between these constituents. For example, in the rule turning a PP into a (post-copular) VP, we require that the PP semantics be construable as a specifying a predication on the subject of the VP:

```
(VP (AGR :P :N) :MOOD (PRED) :SUBJ :WFF)
→
(PP (PREDICATE-P) (WH-) :PP)
(PREDICATIVE-PP :PP :SUBJ :WFF))
```

PREDICATIVE-PP is a constraint clause, taking the semantics of an NP and a PP and returning a wff that is a predication that the PP may be construed as making of the NP. As the rule shows, the wff returned is passed up as the translation of the VP.

Constraint nodes are used not only to impose a stipulation on the constituents of a rule but also to allow for multiple ways to satisfy these constituents. For example, the PP "in the Information Sciences Division" can apply differently to different subjects: to a person, in which case it indicates that the person is an employee of the Information Sciences Division, or to a department, in which it indicates that the department is one of those making up the Information Sciences Division.

So far we have not indicated how the system would distinguish between these two cases: in other words, how it would tell a person and a department apart. In the next section, we discuss the internal structure of NP semantics where such information is stored.

4 REPRESENTATION OF PHRASAL SEMANTICS

The variables :SUBJ and :OBJ in the previously presented lexical rule for "hire" are typed to range over term structures that represent noun phrase semantics. These structures have the following form:

(Q-TERM QUANTIFIER VAR NOM-SEM)

The QUANTIFIER is one of the many quantifiers that correspond to determiners in English: ALL, SOME, THE and various WH determiners. Proper NPs are treated as definite descriptions in our system; they are thus represented using the THE quantifier.

The VAR denotes a variable of the object language, and is left uninstantiated (being filled in by a unique object-language variable by the quantifier module). The NOM-SEM represents the set that the quantification ranges over; it effectively represents the semantics of the head of the NP after modification by the NP's other constituents.

NOM-SEMs have a structure of their own. The principal functor of this type is NOM, which has the argument structure:

(NOM PARAM-LIST SET-EXP SORT)

The PARAM-LIST is a (possibly empty) list of parameters, used to indicate the free argument places in a relational noun. SET-EXP is a logical expression which denotes a set of individuals. SORT is a term structure which represents the semantic class of the elements of SET-EXP.

Note that this means that the SORT field of the NOM is accesible, via one level of indirection, from the Q-TERM NP representation. It is this feature which provides the means for selectional restriction based on semantic class.

Semantic classes (arranged in a hierarchy) are represented as complex terms, whose arguments may themselves be complex terms. A translation (described in the next section) is established between semantic classes and these terms such that non-empty overlap between two classes corresponds to unifiability of the corresponding terms, and disjointness between classes corresponds to non-unifiability of the corresponding terms.

As an example, we give a second version of the rule for "hire", this time incorporating the selectional restriction that a department hires a person:

```
(V (TRANSITIVE
     (HIRE' #1=(Q-TERM :Q1 :VAR1
                  (NOM :PARS1 :SET1
                       (INANIMATE (DEPTS))))
            #2=(Q-TERM :Q2 :VAR2
                  (NOM :PARS2 :SET2
                       (PERSON))))
     #1#
     #2#
   :P :N :MOOD)
→
(hire)
```

The use of the numbers "1" and "2" above is intended to indicate the multiple occurrences of the complex forms they label. (Note that this is simply the Common Lisp (Steele Jr., 1984) convention for re-entrant list structure in the rule above. This is at present only used for notational compactness; the system does not currently attempt to take computational advantage of re-entrancy during unification or other processing.)

Adjective phrase semantic representations (ADJ-SEMs) come in two varieties:

(MODIFYING-ADJ NOM-SEM NOM-SEM)

and

(PREDICATIVE-ADJ NP-SEMANTICS WFF)

These represent different semantic types of adjective, and will be explained in a later section.

The last major category whose semantic representation we consider here is the prepositional phrase. PPs in our system are given only partial semantic interpretations consisting of the preposition of the PP and the translation of the PP's NP object. Their representations are thus of the following form:

(PP-SEM PREP NP-SEMANTICS)

5 ENCODING SEMANTIC CLASSES AS TERMS

The translation from semantic classes to complex terms can be performed systematically. In this section we present an algorithm for translating semantic classes to terms, designed to work on taxonomies of semantic classes represented in a system such as KL-ONE (Schmolze and Israel, 1983) or NIKL (Moser, 1983). It has the advantage, important from the point of view of such systems, that it correctly handles the distinction between "primitive" and "defined" classes — "defined" meaning that the class is simply an intersection of two or more other classes.

The algorithm is seen in Figure 1, where the main work is done by the function TRANSLATE. Throughout, the symbol ":ANY" indicates a "don't care" variable, unifying with anything. This is in fact the only use of variables made. The operation REGULARIZE is used to remove non-primitive classes from the taxonomy, and set them aside. It is simple and we do not give it here.

We now consider the classes PERSON, MALE, FEMALE, ADULT, CHILD, MAN and PRIEST. MALE and FEMALE are disjoint sub-classes of PERSON, as are ADULT and CHILD. MAN is the class which is the intersection of ADULT and MALE. PRIEST is a subclass of MAN, but not identical to it. Following are the translations the algorithm in Figure 1 gives to several of these classes:

```
PERSON → (PERSON :ANY :ANY)
ADULT → (PERSON (ADULT :ANY) :ANY)
MALE → (PERSON :ANY (MALE:ANY))
MAN → (PERSON (ADULT :ANY) (MALE :ANY))
PRIEST → (PERSON (ADULT (PRIEST))
                 (MALE (PRIEST)))
```

Essentially, the algorithm works by mapping each set of mutually disjoint children of the class to an argument place of the term to be associated with that class. The term associated with a class has the same depth as the depth of the class in the taxonomy.

The translation produces by this algorithm are similar to those produced by the algorithm by Mellish (Mellish, 1988). We claim two advantages for ours. First, and as already pointed out, it takes into account the difference between "if" (primitive) and if-and-only-if (non-primitive) axiomitizations, where it would seem that the Mellish algorithm does not. Second, it is simpler, not requiring such notions as "paths" and extensions "to" and "beyond" them.

As a final comment on the issue of encoding semantic classes as terms, we note that there is another encoding method which may have been overlooked: that is, encoding each class as a term which has the same number of arguments as there are classes. It works as follows. In the argument position corresponding to the class being translated put a "1", and put a "1" in argument positions corresponding to subsuming classes as well. In argument positions corresponding to disjoint classes put a "0". In all other positions put a "dont-care" variable. While perhaps using space inefficiently, this encoding will have all the desired properties.

6 ANALYSIS OF NOUN PHRASES AND NOUN MODIFIERS

The following is a simplified version of the rule for regular count noun phrases:

```
(NP :NSUBCATFRAME (AGR :P :N) :WH
    (Q-TERM :Q :VAR :NOM5))
→
(DETERMINER :N :WH :NOM1 :NOM2 :Q)
(OPTNONPOSADJP (AGR :P :N) :NOM4 :NOM5)
(OPTADJP (AGR :P :N) (PRENOMADJ) :NOM3
    :NOM4)
(N-BAR :NSUBCATFRAME (AGR :P :N) :NOM1)
(OPTNPADJUNCT (AGR :P :N) :NOM2 :NOM3)
```

This rule generates NPs that have at least a determiner and a head noun, and which have zero or more prenominal superlative or comparative adjectives ("fastest", "bigger" etc.), prenominal positive adjectives ("red","alleged") and adjuncts ("in the house", "that came from Florida"). Its effect is to take the NOM-SEM semantics of the head noun (the N-BAR) and thread it through the various modifications, add a quantifier and a variable for quantification and deliver the resulting package as the semantics for the whole NP.

The initial NOM-SEM comes from the N-BAR, and is signified by :NOM1, the variable in that position. It is first of all passed to the DETERMINER. Along with a quantifier, :Q, the DETERMINER passes back a possibly modified NOM-SEM, :NOM2. The reason for this is that the determiner may be possessive, and a possessive determiner effectively functions as a noun modifier which enters into scope relations with other modifiers of the NP. Consider the noun phrase "John's best book". This cannot be analyzed as

```
(SET X (BEST' BOOK') (EQUAL (AUTHOR-OF X)
                            JOHN'))
```

that is, as the subset of the best books in the world that also happen to be written by John. Instead, it must be analyzed as:

```
(BEST' (SET X BOOK' (EQUAL (AUTHOR-OF X)
                           JOHN')))
```

```
  TRANSLATE-TAXONOMY(top)

  ::=
[

  CONJUNCTION-CLASSES := REGULARIZE(top)

  TRANSLATIONS := TRANSLATE(top)

  (for pairing in CONJUNCTION-CLASSES
    do  tmp := :ANY
        (for class in pairing[2]
             do tmp := UNIFY(TRANSLATIONS(class),tmp))
        TRANSLATIONS(pairing[1]) := tmp)

 TRANSLATIONS

]

TRANSLATE(concept) ::=

 [
   DISJOINTNESS-CLASSES :=
      (PICK-ARBITRARY-ORDER
         (SET s (POWER (CHILDREN concept))
            (AND (NON-EMPTY s)
                 (FORALL x s (FORALL y s (-> (NOT (= x y))
                                             (DISJOINT x y)))))))

  (for class in DISJOINTNESS-CLASSES
    do (for sub-concept in class
          do (for trans in (TRANSLATE sub-concept)
               do (TRANSLATIONS trans[1]) :=
                    (UNIFY (CONS concept
                                 (for class' in DISJOINTNESS-CLASSES
                                    collect (if (= class class')
                                                trans[2]
                                                :ANY)))
                           (TRANSLATIONS trans[1])))))

   TRANSLATIONS(concept) :=
        (CONS concept (for class in DISJOINTNESS-CLASSES collect :ANY))

   TRANSLATIONS

  ]
```

Figure 1: Translation Algorithm

The essential point is that the possessive DETERMINER must carry out its modification before other elements of the NP can, yet must still follow all other modifications in affixing a quantifier to the final result of the NP. If the determiner is conceived of as just a higher-order function returning a single value, as in Montague Grammar, it is difficult to see how this can be done. The virtue of our unification approach is that it allows the determiner to return as separate values both a quantifier and a suitably modified nominal.

If the determiner is not possessive it simply passes up the same NOM-SEM it was originally given. The NOM-SEM returned by the DETERMINER, whether modified or not, is then passed down to the adjuncts of the NP as :NOM2, which modify it and return :NOM3. This is then passed to the regular (non-superlative) prenominal adjectives for further modification, returning :NOM4. Finally, :NOM4 is passed to the constituent OPTNONPOSADJP, the optional superlative adjectives. The final NOM-SEM, :NOM5, is passed up to become an element of the complete Q-TERM semantics of the NP.

As an example of the action of the modifying elements in the above rule, consider the following rule for generating an NP adjunct from a PP:

```
(OPTNPADJUNCT :NOM1 :NOM2)
→
(PP :PP)
(MODIFYING-PP :PP :NOM1 :NOM2)
```

The NOM-SEM passed in from the containing NP, :NOM1, is in turn passed down to a constraint node, MODIFYING-PP, which takes the semantics of the PP, :PP, and "computes" the modified NOM-SEM, :NOM2, which is then passed back to the NP as the result of the modification.

MODIFYING-PP is used to encompass different kinds of PP modification. Relational modification, where the PP essentially fills in an argument, is handled by the following solution to MODIFYING-PP:

```
(MODIFYING-PP (PP-SEM (OFPREP) :NP)
              (NOM (PARAM :NP) :SET :SORT)
              (NOM (NO-PARAM) :SET :SORT))
→
```

Since this rule is a constraint node solution, its right-hand side is empty. It unifies the NP object of the PP with the "parameter NP" of the argument nominal. Of course, it will not be unifiable if the argument nominal does not contain a parameter NP, or if the parameter NP of the argument nominal contains the wrong semantic type.

The lexical rule for relational noun "salary" is as follows:

```
(N (NOM (PARAM #1=(Q-TERM :Q :VAR
                          (NOM :PARS :SET
                               (PERSON))
        (SETOF (SAL' #1#))
        (INANIMATE (DOLLAR-AMT))
→
(salary)
```

Note that requirement that the filler of the slot be of sort PERSON, and the co-occurence of this filler inside the NOM's set expression.

Of course PPs can also occur in a predicative sense. For example, a person can be "in" in a department. To handle this we have the following solution to to the constraint node PREDICATIVE-PP:

```
(PREDICATIVE-PP
    (PP-SEM (INPREP)
            #1=(Q-TERM :Q1 :VAR1
                    (NOM :PARS1 :SET1
                         (INANIMATE (DEPTS)))))
    #2=(Q-TERM :Q2 :VAR2
               (NOM :PARS2 :SET2 (PERSON)))
    (EQUAL (DEPT-OF #2#) #1#))
→
```

Note that this constraint solution will only unify if the class of the NP object of the PP unifies with DEPTS, and the class of the NP being predicated of unifies with PERSON.

When such a PP occurs as an adjunct to an NP, the derivation passes through the following indirect "lifting" rule:

```
(MODIFYING-PP
    :PP
    (NOM :PAR :SET :SORT)
    (NOM :PAR (SET :VAR :SET :WFF) :SORT))
→
(PREDICATIVE-PP
    :PP
    (Q-TERM (BOUND-Q) :VAR
        (NOM :PAR :SET :SORT))
    :WFF)
```

Although the right-hand side of the rule is in this case not empty, it will like all constraint nodes derive the empty string in the end.

Similar distinctions of modificational power are seen in the case of adjectives, where an adjective like "average" or "previous" has the power to abstract over free parameters of the noun meaning, and an adjective like "female" does not. Consider the rule below:

```
(OPTADJP (AGR :P :N) :POSIT :NOM1 :NOM3)
→
(ADJP (AGR :P :N) :POSIT :ADJ-SEM)
(OPTADJP (AGR :P :N) :POSIT :NOM1 :NOM2)
(MODIFIYING-ADJ-READING :ADJ-SEM :NOM2
                                    :NOM3)
```

This rule generates a string of one or more adjectives. Nominal semantics is threaded through the adjectives right to left. Adjectives like "previous", with the power to modify the whole noun, have a semantic representation headed by the functor "MODIFYING-ADJ", while adjectives like "female", which only operate upon individual elements of the noun's extension, have a representation headed by the functor "PREDICATIVE-ADJ". The constraint node MODIFYING-ADJ-READING accepts the first kind of adjective unchanged and lifts the second kind to the appropriate level. Note that while predicative PPs and adjectives can be "lifted" to the noun modifying level, the converse is not true. That is, the system does not allow "That value is previous" or "That salary is of Clark".

7 CONSTRAINT NODES AND THEIR IMPACT ON PARSING

Constraint nodes are generally useful in that they allow one to give a name to a particular condition and use it in multiple places throughout the grammar. Consider verbs which take PPs and ADJPs as complements. In "John became happy", it is intended that that the adjective "happy" apply to the subject "John". It would not make sense to say "The table became happy". Similarly, in "I put the book on the floor", the PP "on the floor" is intended to apply to the object NP "the book" and it would not make sense to say "I put the idea on the floor" Semantic type constraints in such cases clearly hold not just between the verb and its various arguments, but *between the arguments themselves.* A constraint node like PREDICATIVE-PP can be used to express this relationship between arguments where it is needed.

The HARC system currently employs a bottom-up left-to-right parser. The decision to use a bottom-up parser was made in order to facilitate the eventual handling of fragmentary and ill-formed input.

The parser is based on the algorithm of Graham, Harrison and Ruzzo (Susan L. Graham, 1980), but has been modified to work with a unification grammar(Haas, to appear). Formally speaking, this algorithm can parse the kind of grammar we have been discussing without any modification, since the constraint nodes and their solutions can simply be incorporated into the algorithm's empty symbols table.

For a non-toy domain, however, this increases the size of the parse tables intolerably. We have therefore modified the algorithm so that it treats constraint node empty symbols specially, not expanding them when the parse tables are built but instead waiting until parse time where it solves them top-down through a process that might be thought of as a kind of all-paths non-backtracking Prolog.

A problem still appears when constraint nodes receive traces as arguments. Until a trace is bound, it of course contains very little information, and hence unifies with almost any constraint node solution. Since bottom-up parsing often hypothesizes traces, there is a consequent combinatorial explosion which can lead to slow parsing.

The obvious solution to this problem is simply to defer the attempt to solve constraint nodes until the point in the parse where they have received adequate instantiation. The definition of "adequate" clearly differs from constraint node to constraint node: in the case of PREDICATIVE-PP it might be that the preposition and class of NP object be known. Until constraint nodes are sufficiently instantiated to bother solving, they can simply be carried as extra riders on chart edges, being passed up as new edges are built. At the time of writing this solution is in the process of being implemented.

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. 00014–89–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.

The author would like to thank Andy Haas, who was the original impetus behind the change to a unification-style semantics in the BBN SLS project, and who made important contributions to the ideas presented here.

References

S. Boisen, Y. Chow, A. Haas, R. Ingria, S. Roukos, R. Scha, D. Stallard, and M. Vilain. *Integration of Speech and Natural Language: Final Report.* Technical Report 6991, BBN Systems and Technologies Corporation, Cambridge, Massachusetts, 1989.

Yen-lu Chow and Richard Schwartz. The Optimal N-Best algorithm: An efficient procedure for finding the top N sentence hypotheses. In *Proceedings of the Speech and Natural Language Workshop October 1989.* DARPA, Morgan Kaufmann Publishers, Inc., October 1989.

Andrew Haas. A New Parsing Algorithm for Unification Grammar. *Compational Linguistics,* (to appear).

C.S. Mellish. Implementing systemic classification by unification. *Computational Linguistics,* 14(1):40–51, 1988.

R. Montague. The proper treatment of quantification in ordinary english. In *Approaches to Natural Language. Proceedings of the 1970 Stanford Workship on Grammar and Semantics,* pages 221–242. Dordrecht: D.Reidel, 1973.

Margaret Moser. *An Overview of NIKL.* Technical Report Section of BBN Report No. 5421, Bolt Beranek and Newman Inc., 1983.

Fernando C.N. Pereira and David H.D. Warren. Definite clause grammars for language analysis – a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence,* 13:231–278, 1980.

J. Schmolze and D. Israel. KL-ONE: Semantics and classification. In *Research in Knowlege Representation for Natural Language Understanding, Annual Report: 1 September 1982 to 31 August 1983.* BBN Report No. 5421, 1983.

Guy L. Steele Jr. *Common LISP: The Language.* Digital Press, Digital Equipment Corporation, 1984.

Walter L. Ruzzo Susan L. Graham, Michael A. Harrison. An improved context-free recognizer. *ACM Transactions on Programming Languages and Systems,* 2(3):415–461, 1980.