RECOVERING IMPLICIT INFORMATION

Martha S. Palmer, Deborah A. Dahl, Rebecca J. Schiffman, Lynette Hirschman,
Marcia Linebarger, and John Dowding
Research and Development Division
SDC -- A Burroughs Company
P.O Box 517
Paoli, PA 19301 USA

ABSTRACT

This paper describes the SDC PUNDIT, (Prolog UNDerstands Integrated Text), system for processing natural language messages.[1] PUNDIT, written in Prolog, is a highly modular system consisting of distinct syntactic, semantic and pragmatics components. Each component draws on one or more sets of data, including a lexicon, a broad-coverage grammar of English, semantic verb decompositions, rules mapping between syntactic and semantic constituents, and a domain model.

This paper discusses the communication between the syntactic, semantic and pragmatic modules that is necessary for making implicit linguistic information explicit. The key is letting syntax and semantics recognize missing linguistic entities as implicit entities, so that they can be labelled as such, and reference resolution can be directed to find specific referents for the entities. In this way the task of making implicit linguistic information explicit becomes a subset of the tasks performed by reference resolution. The success of this approach is dependent on marking missing syntactic constituents as elided and missing semantic roles as ESSENTIAL so that reference resolution can know when to look for referents.

[1] This work is supported in part by DARPA under contract N00014-85-C-0012, administered by the Office of Naval Research. APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED.

1. Introduction

This paper describes the SDC PUNDIT[2] system for processing natural language messages. PUNDIT, written in Prolog, is a highly modular system consisting of distinct syntactic, semantic and pragmatics components. Each component draws on one or more sets of data, including a lexicon, a broad-coverage grammar of English, semantic verb decompositions, rules mapping between syntactic and semantic constituents, and a domain model. PUNDIT has been developed cooperatively with the NYU PROTEUS system (Prototype Text Understanding System), These systems are funded by DARPA as part of the work in natural language understanding for the Strategic Computing Battle Management Program. The PROTEUS/PUNDIT system will map Navy CASREP's (equipment casualty reports) into a database, which is accessed by an expert system to determine overall fleet readiness. PUNDIT has also been applied to the domain of computer maintenance reports, which is discussed here.

The paper focuses on the interaction between the syntactic, semantic and pragmatic modules that is required for the task of making implicit information explicit. We have isolated two types of implicit entities: syntactic entities which are missing syntactic constituents, and semantic entities which are unfilled semantic roles. Some missing entities are optional, and can be ignored. Syntax and semantics have to recognize the OBLIGATORY missing entities and then mark them so that reference resolution knows to find specific referents for those entities, thus making the implicit information explicit. Reference resolution uses two different methods for filling the different types of entities which are also used for general noun phrase reference problems. Implicit syntactic entities, ELIDED CONSTITUENTS, are treated like pronouns, and implicit semantic entities, ESSENTIAL ROLES are treated like definite noun phrases. The pragmatic module as currently implemented consists mainly of a reference resolution component, which is sufficient for the pragmatic issues described in this paper. We are in the process of adding a time module to handle time issues that have arisen during the analysis of the Navy CASREPS.

2. The Syntactic Component

The syntactic component has three parts: the grammar, a parsing mechanism to execute the grammar, and a lexicon. The grammar consists of context-free BNF definitions (currently numbering approximately 80) and associated restrictions (approximately 35). The restrictions enforce context-sensitive well-formedness constraints and, in some cases, apply optimization strategies to prevent unnecessary structure-building. Each of these three parts is described further below.

[2] Prolog UNDderstands Integrated Text

2.1. Grammar Coverage

The grammar covers declarative sentences, questions, and sentence fragments. The rules for fragments enable the grammar to parse the "telegraphic" style characteristic of message traffic, such as ***disk drive down,*** and ***has select lock***. The present grammar parses sentence adjuncts, conjunction, relative clauses, complex complement structures, and a wide variety of nominal structures, including compound nouns, nominalized verbs and embedded clauses.

The syntax produces a detailed surface structure parse of each sentence (where "sentence" is understood to mean the string of words occurring between two periods, whether a full sentence or a fragment). This surface structure is converted into an "intermediate representation" which regularizes the syntactic parse. That is, it eliminates surface structure detail not required for the semantic tasks of enforcing selectional restrictions and developing the final representation of the information content of the sentence. An important part of regularization involves mapping fragment structures onto canonical verb-subject-object patterns, with missing elements flagged. For example, the **tvo** fragment consists of a **tensed verb** + **object** as in ***Replaced spindle motor.*** Regularization of this fragment, for example, maps the **tvo** syntactic structure into a verb+subject+object structure:

verb(replace),subject(X),object(Y)

As shown here, **verb** becomes instantiated with the surface verb, e.g., ***replace*** while the arguments of the **subject** and **object** terms are variables. The semantic information derived from the noun phrase object ***spindle motor*** becomes associated with **Y**. The absence of a surface subject constituent results in a lack of semantic information pertaining to **X**. This lack causes the semantic and pragmatic components to provide a semantic filler for the missing subject using general pragmatic principles and specific domain knowledge.

2.2. Parsing

The grammar uses the Restriction Grammar parsing framework [Hirschman1982, Hirschman1985], which is a logic grammar with facilities for writing and maintaining large grammars. Restriction Grammar is a descendent of Sager's string grammar [Sager1981]. It uses a top-down left-to-right parsing strategy, augmented by dynamic rule pruning for efficient parsing [Dowding1986]. In addition, it uses a meta-grammatical approach to generate definitions for a full range of co-ordinate conjunction structures [Hirschman1986].

2.3. Lexical Processing

The lexicon contains several thousand entries related to the particular subdomain of equipment maintenance. It is a modified version of the LSP lexicon with words classified as to part of speech and subcategorized in limited ways (e.g., verbs are subcategorized for their complement types). It also handles multi-word idioms, dates, times and part numbers. The lexicon can be expanded by means of an interactive lexical entry program.

The lexical processor reduces morphological variants to a single root form which is stored with each entry. For example, the form ***has*** is transformed to the root form ***have*** in ***Has select lock.*** In addition, this facility is useful in handling abbreviations: the term ***awp*** is regularized to the multi-word expression ***waiting^for^part.*** This expression in turn is regularized to the root form ***wait^for^part*** which takes as a direct object a particular part or part number, as in ***is awp 2155-6147.***

Multi-word expressions, which are typical of jargon in specialized domains, are handled as single lexical items. This includes expressions such as ***disk drive*** or ***select lock***, whose meaning within a particular domain is often not readily computed from its component parts. Handling such frozen expressions as "idioms" reduces parse times and number of ambiguities.

Another feature of the lexical processing is the ease with which special forms (such as part numbers or dates) can be handled. A special "forms grammar", written as a definite clause grammar[Pereira1980] can parse part numbers, as in ***awaiting part 2155-6147***, or complex date and time expressions, as in ***disk drive up at 11/17-1236.*** During parsing, the forms grammar performs a well-formedness check on these expressions and assigns them their appropriate lexical category.

3. Semantics

There are two separate components that perform semantic analysis, NOUN PHRASE SEMANTICS and CLAUSE SEMANTICS. They are each called after parsing the relevant syntactic structure to test semantic well-formedness while producing partial semantic representations. Clause semantics is based on Inference Driven Semantic Analysis [Palmer1985] which decomposes verbs into component meanings and fills their semantic roles with syntactic constituents. A KNOWLEDGE BASE, the formalization of each domain into logical terms, SEMANTIC PREDICATES, is essential for the effective application of Inference Driven Semantic Analysis, and for the final production of a text representation. The result of the semantic analysis is a set of PARTIALLY instantiated semantic predicates which is similar to a frame representation. To produce this representation, the semantic components share access to a knowledge base, the DOMAIN MODEL, that contains generic descriptions of the domain elements corresponding to the lexical entries. The model includes a detailed representation of the types of assemblies that these elements can occur in. The semantic components are designed to work independently of the particular model, and rely on an interface to ensure a well-defined interaction with the domain model. The domain model, noun phrase semantics and clause semantics are all explained in more detail in the following three subsections.

3.1. Domain Model

The domain currently being modelled by SDC is the Maintenance Report domain. The texts being analyzed are actual maintenance reports as they are called into the Burroughs Telephone Tracking System by the field engineers and typed in by the telephone operator. These reports give information about the customer who has the problem, specific symptoms of the problem, any actions take by the field engineer to try and correct the problem, and success or failure of such actions. The goal of the text analysis is to automatically generate a data base of maintenance information that can be used to correlate customers to problems, problem types to machines, and so on.

The first step in building a domain model for maintenance reports is to build a semantic net-like representation of the type of machine involved. The machine in the example text given below is the B4700. The possible parts of a B4700 and the associated properties of these parts can be represented by an **isa** hierarchy and a **haspart** hierarchy. These hierarchies are built using four basic predicates: **system,isa,hasprop, haspart**. For example the system itself is indicated by **system(b4700)**. The **isa** predicate associates TYPES with components, such as **isa(spindle^motor,motor)**. Properties are associated with components using the **hasprop** relationship, are are inherited by anything of the same type. The main components of the system: **cpu**, **power_supply**, **disk**, **printer**, **peripherals**, etc., are indicated by **haspart** relations, such as **haspart(b4700,cpu), haspart(b4700,power_supply), haspart(b4700,disk),**,etc. These parts are themselves divided into subparts which are also indicated by **haspart** relations, such as **haspart(power_supply, converter)**.

This method of representation results in a general description of a computer system. Specific machines represent INSTANCES of this general representation. When a particular report is being processed, **id** relations are created by noun phrase semantics to associate the specific computer parts being mentioned with the part descriptions from the general machine representation. So a particular B4700 would be indicated by predicates such as these: **id(b4700,system1), id(cpu,cpu1), id(power_supply,power_supply1),** etc.

3.2. Noun phrase semantics

Noun phrase semantics is called by the parser during the parse of a sentence, after each noun phrase has been parsed. It relies heavily on the domain model for both determining semantic well-formedness and building partial semantic representations of the noun phrases. For example, in the sentence, ***field engineer replaced disk drive at 11/2/0800***, the phrase ***disk drive at 11/2/0800*** is a syntactically acceptable noun phrase, (as in ***participants at the meeting***). However, it is not semantically acceptable in that ***at 11/20/800*** is intended to designate the time of the replacement, not a property of the disk drive. Noun phrase semantics will inform the parser that the noun phrase is not semantically acceptable, and the parser can then look for another parse. In order for this capability to be fully utilized, however, an extensive set of domain-specific rules about semantic acceptability is required. At present we have only the minimal set used for the development of the basic mechanism. For example, in the case described here, ***at 11/2/0800*** is excluded as a modifier for ***disk drive*** by a rule that permits only the name of a location as the object of ***at*** in a prepositional phrase modifying a noun phrase.

The second function of noun phrase semantics is to create a semantic representation of the noun phrase, which will later be operated on by reference resolution. For example, the semantics for ***the bad disk drive*** would be represented by the following Prolog clauses.

```
[id(disk^drive,X),
 bad(X),
 def(X),       that is, X was referred to with a full, definite noun phrase,
 full_npe(X)]  rather than a pronoun or indefinite noun phrase.
```

3.3. Clause semantics

In order to produce the correct predicates and the correct instantiations, the verb is first decomposed into a semantic predicate representation appropriate for the domain. The arguments to the predicates constitute the SEMANTIC ROLES of the verb, which are similar to cases. There are domain specific criteria for selecting a range of semantic roles. In this domain the semantic roles include: **agent**,**instrument**,**theme**, **object1**,**object2**, **symptom** and **mod**. Semantic roles can be filled either by a syntactic constituent supplied by a mapping rule or by reference resolution, requiring close cooperation between semantics and reference resolution. Certain semantic roles are categorized as ESSENTIAL, so that pragmatics knows that they need to be filled if there is no syntactic constituent available. The default categorization is NON-ESSENTIAL, which does not require that the role be filled. Other semantic roles are categorized as NON-SPECIFIC or SPECIFIC depending on whether or not the verb requires a specific referent for that semantic role (see Section 4). The example given in Section 5 illustrates the use of both a non-specific semantic role and an essential semantic role. This section explains the decompositions of the verbs relevant to the example, and identifies the important semantic roles.

The decomposition of ***have*** is very domain specific.

```
have(time(Per)) < -
        symptom(object1(O1),symptom(S),time(Per))
```

It indicates that a particular **symptom** is associated with a particular **object**, as in "the disk drive has select lock." The **object1** semantic role would be filled by the disk drive, the subject of the clause, and the **symptom** semantic role would be filled by ***select lock***, the object of the clause. The **time(Per)** is always passed around, and is occasionally filled by a time adjunct, as in ***the disk drive had select lock at 0800***.

In addition to the mapping rules that are used to associate syntactic constituents with semantic roles, there are selection restrictions associated with each semantic role. The selection restrictions for ***have*** test whether or not the filler of the **object1** role is allowed to have the type of symptom that fills the **symptom** role. For example, only disk drives have select locks.

Mapping Rules

The decomposition of ***replace*** is also a very domain specific decomposition that indicates that an **agent** can use an **instrument** to exchange two **objects**.

```
replace(time(Per)) <-
    cause(agent(A),
        use(instrument(I),
            exchange(object1(O1),object2(O2),time(Per))))
```

The following mapping rule specifies that the **agent** can be indicated by the subject of the clause.

```
agent(A) <- subject(A) / X
```

The mapping rules make use of intuitions about syntactic cues for indicating semantic roles first embodied in the notion of case [Fillmore1968, Palmer1981]. Some of these cues are quite general, while other cues are very verb-specific. The mapping rules can take advantage of generalities like "SUBJECT to AGENT" syntactic cues while still preserving context sensitivities. This is accomplished by making the application of the mapping rules "situation-specific" through the use of PREDICATE ENVIRONMENTS. The previous rule is quite general and can be applied to every **agent** semantic role in this domain. This is indicated by the X on the right hand side of the "/" which refers to the predicate environment of the **agent**, i.e., anything. Other rules, such as "WITH-PP to OBJECT2," are much less general, and can only apply under a set of specific circumstances. The predicate environments for an **object1** and **object2** are specified more explicitly. An **object1** can be the object of the sentence if it is contained in the semantic decomposition of a verb that includes an **agent** and belongs to the ***repair*** class of verbs. An **object2** can be indicated by a ***with*** prepositional phrase if it is contained in the semantic decomposition of a replace verb:

```
object1(Part1) <- obj(Part1)/ cause(agent(A),Repair_event)

object2(Part2) <-
    pp(with,Part2) /
```

```
cause(agent(A),use(I,exchange(object1(O1),object2(Part2),T)))
```

Selection Restrictions

The selection restriction on an **agent** is that it must be a field engineer, and an **instrument** must be a tool. The selection restrictions on the two objects are more complicated, since they must be machine parts, have the same type, and yet also be distinct objects. In addition, the first object must already be associated with something else in a **haspart** relationship, in other words it must already be included in an existing assembly. The opposite must be true of the second object: it must not already be included in an assembly, so it must not be associated with anything else in a **haspart** relationship.

There is also a pragmatic restriction associated with both objects that has not been associated with any of the semantic roles mentioned previously. Both **object1** and **object2** are essential semantic roles. Whether or not they are mentioned explicitly in the sentence, they must be filled, preferably by an an entity that has already been mentioned, but if not that, then entities will be created to fill them [Palmer1983]. This is accomplished by making an explicit call to reference resolution to find referents for essential semantic roles, in the same way that reference resolution is called to find the referent of a noun phrase. This is not done for non-essential roles, such as the **agent** and the **instrument** in the same verb decomposition. If they are not mentioned they are simply left unfilled. The **instrument** is rarely mentioned, and the **agent** could easily be left out, as in ***The disk drive was replaced at 0800.***[3] In other domains, the **agent** might be classified as obligatory, and then it wold have to be filled in.

There is another semantic role that has an important pragmatic restriction on it in this example, the **object2** semantic role in ***wait^for^part (awp)***.

```
idiomVerb(wait^for^part,time(Per)) <-
        ordered(object1(O1),object2(O2),time(Per))
```

The semantics of ***wait^for^part*** indicates that a particular type of part has been ordered, and is expected to arrive. But it is not a specific entity that might have already been mentioned. It is a more abstract object, which is indicated by restricting it to being non-specific. This tells reference resolution that although a syntactic constituent, preferably the object, can and should fill this semantic role, and must be of type **machine-part**, that reference resolution should not try to find a specific referent for it (see Section 4).

The last verb representation that is needed for the example is the representation of ***be***.

```
be(time(Per)) <-
```

[3]Note that an elided subject is handled quite differently, as in ***replaced disk drive***. Then the missing subject is

attribute(theme(T),mod(M),time(Per))

In this domain ***be*** is used to associate predicate adjectives or nominals with an object, as in ***disk drive is up*** or ***spindle motor is bad.*** The representation merely indicates that a **mod**ifier is associated with an **theme** in an attribute relationship. Noun phrase semantics will eventually produce the same representation for ***the bad spindle motor,*** although it does not yet.

4. Reference Resolution

Reference resolution is the component which keeps track of references to entities in the discourse. It creates labels for entities when they are first directly referred to, or when their existence is implied by the text, and recognizes subsequent references to them. Reference resolution is called from clause semantics when clause semantics is ready to instantiate a semantic role. It is also called from pragmatic restrictions when they specify a referent whose existence is entailed by the meaning of a verb.

The system currently covers many cases of singular and plural noun phrases, pronouns, ***one***- anaphora, nominalizations, and non-specific noun phrases; reference resolution also handles adjectives, prepositional phrases and possessive pronouns modifying noun phrases. Noun phrases with and without determiners are accepted. Dates, part numbers, and proper names are handled as special cases. Not yet handled are compound nouns, quantified noun phrases, conjoined noun phrases, relative clauses, and possessive nouns.

The general reference resolution mechanism is described in detail in [Dahl1986]. In this paper the focus will be on the interaction between reference resolution and clause semantics. The next two sections will discuss how reference resolution is affected by the different types of semantic roles.

4.1. Obligatory Constituents and Essential Semantic Roles

A slot for a syntactically obligatory constituent such as the subject appears in the intermediate representation whether or not a subject is overtly present in the sentence. It is possible to have such a slot because the absence of a subject is a syntactic fact, and is recognized by the parser. Clause semantics calls reference resolution for such an implicit constituent in the same way that it calls reference resolution for explicit constituents. Reference resolution treats elided noun phrases exactly as it treats pronouns, that is by instantiating them to the first member of a list of potential pronominal referents, the **FocusList.**

assumed to fill the **agent** role, and an appropriate referent is found by reference resolution.

The general treatment of pronouns resembles that of[Sidner1979], although there are some important differences, which are discussed in detail in [Dahl1986]. The hypothesis that elided noun phrases can be treated in much the same way as pronouns is consistent with previous claims by [Gundel1980], and [Kameyama1985], that in languages which regularly allow zero-np's, the zero corresponds to the focus. If these claims are correct, it is not surprising that in a sublanguage that allows zero-np's, the zero should also correspond to the focus.

After control returns to clause semantics from reference resolution, semantics checks the selectional restrictions for that referent in that semantic role of that verb. If the selectional restrictions fail, backtracking into reference resolution occurs, and the next candidate on the FocusList is instantiated as the referent. This procedure continues until a referent satisfying the selectional restrictions is found. For example, in ***Disk drive is down. Has select lock***, the system instantiates the disk drive, which at this point is the first member of the **FocusList**, as the **object1** of ***have***:

```
[event39]
have(time(time1))
 symptom(object1([drive10]),
        symptom([lock17]),
        time(time1))
```

Essential roles might also not be expressed in the sentence, but their absence cannot be recognized by the parser, since they can be expressed by syntactically optional constituents. For example, in ***the field engineer replaced the motor.***, the new replacement motor is not mentioned, although in this domain it is classified as semantically essential. With verbs like ***replace***, the type of the replacement, ***motor***, in this case, is known because it has to be the same type as the replaced object. Reference resolution for these roles is called by pragmatic rules which apply when there is no overt syntactic constituent to fill a semantic role. Reference resolution treats these referents as if they were full noun phrases without determiners. That is, it searches through the context for a previously mentioned entity of the appropriate type, and if it doesn't find one, it creates a new discourse entity. The motivation for treating these as full noun phrases is simply that there is no reason to expect them to be in focus, as there is for elided noun phrases.

4.2. Noun Phrases in Non-Specific Contexts

Indefinite noun phrases in contexts like ***the field engineer ordered a disk drive*** are generally associated with two readings. In the specific reading the disk drive ordered is a particular disk drive, say, the one sitting on a certain shelf in the warehouse. In the non-specific reading, which is more likely in this sentence, no particular disk drive is meant; any disk drive of the appropriate type will do. Handling noun phrases in these contexts requires careful integration of the interaction between semantics and reference resolution, because semantics knows about the verbs that create non-specific contexts, and reference resolution knows what to do with noun phrases in these contexts. For these verbs a constraint is associated with the semantics rule for the semantic role **object2** which states that the filler for the **object2** must be non-specific.[4] This constraint is passed to reference resolution, which represents a non-specific noun phrase as having a variable in the place of the pointer, for example, **id(motor,X)**.

Non-specific semantic roles can be illustrated using the **object2** semantic role in ***wait^for^part (awp).*** The part that is being ***awaited*** is non-specific, i.e., can be any part of the appropriate type. This tells reference resolution not to find a specific referent, so the referent argument of the **id** relationship is left as an uninstantiated variable. The analysis of ***fe is awp spindle motor*** would fill the **object1** semantic role with **fe1** from **id(fe,fe1),** and the **object2** semantic role with **X** from **id(spindle^motor,X),** as in **ordered(object1(fe1),object2(X)).** If the spindle motor is referred to later on in a relationship where it must become specific, then reference resolution can instantiate the variable with an appropriate referent such as **spindle^motor3** (See Section 5.6).

5. Sample Text: A sentence-by-sentence analysis

The sample text given below is a slightly emended version of a maintenance report. The parenthetical phrases have been inserted. The following summary of an interactive session with PUNDIT illustrates the mechanisms by which the syntactic, semantic and pragmatic components interact to produce a representation of the text.

1. disk drive (was) down (at) 11/16-2305.
2. (has) select lock.
3. spindle motor is bad.
4. (is) awp spindle motor.
5. (disk drive was) up (at) 11/17-1236.
6. replaced spindle motor.

5.1. Sentence 1: Disk drive was down at 11/16-2305.

As explained in Section 3.2 above, the noun phrase ***disk drive*** leads to the creation of an **id** ***of the form:*** **id(disk^drive,[drive1])** Because dates and names generally refer to unique entities rather than to exemplars of a general type, their **ids** do not contain a type argument: **date([11/16-**

[4] The specific reading is not available at present, since it is considered to be unlikely to occur in this domain.

1100]),name([paoli]).

The interpretation of the first sentence of the report depends on the semantic rules for the predicate ***be.*** The rules for this predicate specify three semantic roles, an **theme** to whom or which is attributed a **modifier**, and the **time.** After a mapping rule in the semantic component of the system instantiates the **theme** semantic role with the sentence subject, ***disk drive,*** the reference resolution component attempts to identify this referent. Because ***disk drive*** is in the first sentence of the discourse, no prior references to this entity can be found. Further, this entity is not presupposed by any prior linguistic expressions. However, in the maintenance domain, when a disk drive is referred to it can be assumed to be part of a B3700 computer system. As the system tries to resolve the reference of the noun phrase ***disk drive*** by looking for previously mentioned disk drives, it finds that the mention of a disk drive presupposes the existence of a system. Since no system has been referred to, a pointer to a system is created at the same time that a pointer to the disk drive is created.

Both entities are now available for future reference. In like fashion, the propositional content of a complete sentence is also made available for future reference. The entities corresponding to propositions are given event labels; thus **event1** is the pointer to the first proposition. The newly created disk drive, system and event entities now appear in the discourse information in the form of a list along with the date.

id(event,[event1])
id(disk^drive,[drive1])
date([11/16-2305])
id(system,[system1])

Note however, that only those entities which have been explicitly mentioned appear in the **FocusList**:

FocusList: [[event1],[drive1],[11/16-2305]]

The propositional entity appears at the head of the focus list followed by the entities mentioned in full noun phrases.[5]

In addition to the representation of the new event, the pragmatic information about the developing discourse now includes information about part-whole relationships, namely that **drive1** is a part which is contained in **system1.**

Part-Whole Relationships:
haspart([system1],[drive1])

The complete representation of **event1**, appearing in the event list in the form shown below, indicates that at the time given in the prepositional phrase ***at 11/16-2305*** there is a state of affairs denoted as **event1** in which a particular

[5] The order in which full noun phrase mentions are added to the **FocusList** depends on their syntactic function and linear order. For full noun phrases, direct object mentions precede subject mentions followed by all other mentions given in the order in which they occur in the sentence. See [Dahl1986], for details.

disk drive, i.e., **drive1**, can be described as ***down***.

```
[event1]
   be(time([11/16-2305]))
   attribute(theme([drive1]),
   mod(down),time([11/16-2305]))
```

5.2. Sentence 2: Has select lock.

The second sentence of the input text is a sentence fragment and is recognized as such by the parser. Currently, the only type of fragment which can be parsed can have a missing subject but must have a complete verb phrase. Before semantic analysis, the output of the parse contains, among other things, the following constituent list: **[subj([X]),obj([Y])]**. That is, the syntactic component represents the arguments of the verb as variables. The fact that there was no overt subject can be recognized by the absence of semantic information associated with **X**, as discussed in Section 3.2. The semantics for the maintenance domain sublanguage specifies that the thematic role instantiated by the direct object of the verb ***to have*** must be a symptom of the entity referred to by the subject. Reference resolution treats an empty subject much like a pronominal reference, that is, it proposes the first element in the **FocusList** as a possible referent. The first proposed referent, **event1** is rejected by the semantic selectional constraints associated with the verb ***have***, which, for this domain, require the role mapped onto the subject to be classified as a machine part and the role mapped onto the direct object to be classified as a symptom. Since the next item in the **FocusList**, **drive1**, is a machine part, it passes the selectional constraint and becomes matched with the empty subject of ***has select lock***. Since no select lock has been mentioned previously, the system creates one. For the sentence as a whole then, two entities are newly created: the select lock (**[lock1]**) and the new propositional event (**[event2]): id(event,[event2]), id(select^lock,[lock1])**. The following representation is added to the event list, and the **FocusList** and **Ids** are updated appropriately.[6]

```
[event2]
have(time(time1))
symptom(object1([drive1]),
symptom([lock1]),time(time1))
```

5.3. Sentence 3: Motor is bad.

In the third sentence of the sample text, a new entity is mentioned, ***motor***. Like ***disk drive*** from sentence 1, ***motor*** is a dependent entity. However, the entity it presupposes is not a computer system, but rather, a disk drive. The

[6] This version only deals with explicit mentions of time, so for this sentence the time argument is filled in with a gensym that stands for an unknown time period. The current version of PUNDIT uses verb tense and verb semantics

newly mentioned motor becomes associated with the previously mentioned disk drive.

After processing this sentence, the new entity **motor3** is added to the **FocusList** along with the new proposition **event3**. Now the discourse information about part-whole relationships contains information about both dependent entities, namely that **motor1** is a part of **drive1** and that **drive1** is a part of **system1**.

```
haspart([drive1],[motor1])
haspart([system1],[drive1])
```

5.4. Sentence 4: is awp spindle motor.

Awp is an abbreviation for an idiom specific to this domain, ***awaiting part***. It has two semantic roles, one of which maps to the sentence subject. The second maps to the direct object, which in this case is the non-specific spindle motor as explained in Section 4.2. The selectional restriction that the first semantic role of ***awp*** be an engineer causes the reference resolution component to create a new engineer entity because no engineer has been mentioned previously. After processing this sentence, the list of available entities has been incremented by three:

```
id(event,[event4])
id(part,[_2317])
id(field^engineer,[engineer1])
```

The new event is represented as follows:

```
[event4]
idiomVerb(wait^for^part,time(time2))
wait(object1([engineer1]),
   object2([_2317]),time(time2))
```

5.5. Sentence 5: disk drive was up at 11/17-0800

In the emended version of sentence 5 the ***disk drive*** is presumed to be the same drive referred to previously, that is, **drive1.** The semantic analysis of sentence 5 is very similar to that of sentence 1. As shown in the following event representation, the predicate expressed by the modifier ***up*** is attributed to the theme **drive1** at the specified time.

```
[event5]
 be(time([11/17-1236]))
 attribute(theme([drive1]),
 mod(up),time([11/17-1236]))
```

---

to derive implicit time arguments.

5.6. Sentence 6: Replaced motor.

The sixth sentence is another fragment consisting of a verb phrase with no subject. As before, reference resolution tries to find a referent in the current **FocusList** which is a semantically acceptable subject given the thematic structure of the verb and the domain-specific selectional restrictions associated with them. The thematic structure of the verb ***replace*** includes an **agent** role to be mapped onto the sentence subject. The only **agent** in the maintenance domain is a field engineer. Reference resolution finds the previously mentioned engineer created for ***awp spindle motor,*** **[engineer1].** It does not find an **instrument,** and since this is not an essential role, this is not a problem. It simply fills it in with another gensym that stands for an unknown filler, **unknown1.**

When looking for the referent of a spindle motor to fill the **object1** role, it first finds the non-specific spindle motor also mentioned in the ***awp spindle motor*** sentence, and a specific referent is found for it. However, this fails the selection restrictions, since although it is a machine part, it is not already associated with an assembly, so backtracking occurs and the referent instantiation is undone. The next spindle motor on the **FocusList** is the one from ***spindle motor is bad,*** (**[motor1]**). This does pass the selection restrictions since it participates in a **haspart** relationship.

The last semantic role to be filled is the **object2** role. Now there is a restriction saying this role must be filled by a machine part of the same type as **object1,** which is not already included in an assembly, viz., the non-specific spindle motor. Reference resolution finds a new referent for it, which automatically instantiates the variable in the **id** term as well. The representation can be decomposed further into the two semantic predicates **missing** and **included**, which indicate the current status of the parts with respect to any existing assemblies. The **haspart** relationships are updated, with the old **haspart** relationship for **[motor1]** being removed, and a new **haspart** relationship for **[motor3]** being added. The final representation of the text will be passed through a filter so that it can be suitably modified for inclusion in a database.

```
[event6]
 replace(time(time3))
 cause(agent([engineer1]),
     use(instrument(unknown1),
         exchange(object1([motor1]),
                 object2([motor2]),
                 time(time3))))
 included(object2([motor2]),time(time3))
 missing(object1([motor1]),time(time3))
```

Part-Whole Relationships:

```
haspart([drive1],[motor3])
haspart([system1],[drive1])
```

6. Conclusion

This paper has discussed the communication between syntactic, semantic and pragmatic modules that is necessary for making implicit linguistic information explicit. The key is letting syntax and semantics recognize missing linguistic entities as implicit entities, so that they can be marked as such, and reference resolution can be directed to find specific referents for the entities. Implicit entities may be either empty syntactic constituents in sentence fragments or unfilled semantic roles associated with domain-specific verb decompositions. In this way the task of making implicit information explicit becomes a subset of the tasks performed by reference resolution. The success of this approach is dependent on the use of syntactic and semantic categorizations such as ELLIDED and ESSENTIAL which are meaningful to reference resolution, and which can guide reference resolution's decision making process.

ACKNOWLEDGEMENTS
We would like to thank Bonnie Webber for her very helpful suggestions on exemplifying semantics/pragmatics cooperation.

REFERENCES

[Dahl1986]
Deborah A. Dahl, Focusing and Reference Resolution in PUNDIT, submitted for publication, 1986.

[Dowding1986]
John Dowding and Lynette Hirschman, Dynamic Translation for Rule Pruning in Restriction Grammar, submitted to AAAI-86, Philadelphia, 1986.

[Fillmore1968]
C. J. Fillmore, The Case for Case. In ***Universals in Linguistic Theory***, E. Bach and R. T. Harms (ed.), Holt, Rinehart, and Winston, New York, 1968.

[Gundel1980]
Jeanette K. Gundel, Zero-NP Anaphora in Russian. ***Chicago Linguistic Society Parasession on Pronouns and Anaphora***, 1980.

[Hirschman1982]
L. Hirschman and K. Puder, Restriction Grammar in Prolog. In ***Proc. of the First International Logic Programming Conference***, M. Van Caneghem (ed.), Association pour la Diffusion et le Developpement de Prolog, Marseilles, 1982, pp. 85-90.

[Hirschman1985]
L. Hirschman and K. Puder, Restriction Grammar: A Prolog Implementation. In ***Logic Programming and its Applications***, D.H.D. Warren and M. VanCaneghem (ed.), 1985.

[Hirschman1986]
L. Hirschman, Conjunction in Meta-Restriction Grammar. ***J. of Logic Programming***, 1986.

[Kameyama1985]
Megumi Kameyama, Zero Anaphora: The Case of Japanese, Ph.D. thesis, Stanford University, 1985.

[Palmer1983]
M. Palmer, Inference Driven Semantic Analysis. In ***Proceedings of the National Conference on Artificial Intelligence (AAAI-83)***, Washington, D.C., 1983.

[Palmer1981]
Martha S. Palmer, A Case for Rule Driven Semantic Processing. ***Proc. of the 19th ACL Conference***, June, 1981.

[Palmer1985]
Martha S. Palmer, Driving Semantics for a Limited Domain, Ph.D. thesis, University of Edinburgh, 1985.

[Pereira1980]
F. C. N. Pereira and D. H. D. Warren, 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.

[Sager1981]
N. Sager, ***Natural Language Information Processing: A Computer Grammar of English and Its Applications.*** Addison-Wesley, Reading, Mass., 1981.

[Sidner1979]
Candace Lee Sidner, Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse, MIT-AI TR-537, Cambridge, MA, 1979.