AN INTEGRATED SYNTACTIC AND SEMANTIC SYSTEM FOR NATURAL LANGUAGE UNDERSTANDING

UN SYSTEME SYNTAXIQUE ET SEMANTIQUE INTEGRE POUR LA COMPREHENSION DU LANGAGE NATUREL

Frédérique SEGOND(1) et Karen JENSEN(2)

(1)Institut National des Télécommunications (Evry, France); email: segond at vaxu.int-evry.fr
(2)Microsoft Corporation (Redmond, Washington State, USA); email: karenje at microsoft.com

RESUME

La stratégie présentée ici est à l'origine d'un système intégré de Traitement Automatique du Langage Naturel - le système PLUS (Progressive Language Understanding System), dans lequel les composantes théoriques traditionnelles, syntaxe, sémantique et discours, sont liées pour former un tout. Le système est écrit dans un seul formalisme, PLNLP (Programming Language for Natural Language Processing; Heidorn 1972), qui fournit une architecture efficace pour unifier les différentes composantes. Le système offre une stratégie élégante pour la compréhension du Langage Naturel à large couverture, et indépendante du domaine d'application. A l'heure actuelle, six composantes constituent le système PLUS/PLNLP; elles peuvent être rapidement décrites de la façon suivante: (1) syntaxe (PEG, la grammaire PLNLP de l'anglais), (2) syntaxe affinée (rattachement des constituants), (3) dérivation d'une forme logique (PEGASUS), (4) désambiguïsation, (5) normalisation des relations sémantiques, (6) modèle du discours au niveau des paragraphes.

Les composantes (1) et (3) sont déjà assez avancées et ont été testées dans le cadre de différentes applications. Les composantes (2) et (4) sont en cours de réalisation. Les techniques pour créer le modèle du discours sont établies mais non encore implémentées. Cet article se concentre sur les composantes (3) et (5) avec une attention plus particulière pour (5) qui pose les fondations grammaticales utilisées par le modèle du discours. Des descriptions des autres composantes peuvent être trouvées dans la littérature (par exemple dans Chanod & al. 1991, Jensen & al. 1992 (à paraître)). La composante (3), PEGASUS, est un passage décisif de la syntaxe à la sémantique - sémantique étant entendu comme impliquant, au minimum, la définition de cas ou rôles thématiques (i.e. structure prédicat-argument). La meilleure illustration en est la différence de représentations en entrée et en sortie. L'entrée est un arbre syntaxique; la sortie est un graphe étiqueté et orienté. Un arbre est en premier lieu une représentation syntaxique dans laquelle l'ordre linéaire et la dominance grammaticale sont porteurs d'informations. Un graphe est une représentation sémantique; l'ordre linéaire n'est plus significatif étant donné que l'information apparaît désormais dans les étiquettes des arcs du graphe ou dans ses attributs. Afin de dériver la forme logique, PEGASUS doit traiter à large échelle des phénomènes d'affectation d'arguments y compris dans des cas difficiles comme les dépendances non bornées (par exemple associer le bon objet au verbe "ate" dans la phrase "What did Mary say that John ate?"), le contrôle fonctionnel (par exemple, trouver les sujets et les objets dans le cas des infinitives), les relations actif/passif (s'assurer que les formes actives et passives ont bien les mêmes arguments sous-jacents), etc... Le programme doit également faire apparaître des relations entre les têtes de syntagmes et leurs modifieurs ou adjoints. Il doit en plus prendre en considération les anaphores - anaphores nominales comprenant les pronoms et les référents de GN définis, anaphores verbales : associer les bons arguments et les bons constituants en cas d'ellipse-. Toute la chaîne d'entrée doit être correctement évaluée. Au stade actuel de son développement, PEGASUS ne prend pas en compte les références définies de GN ni la quantification, mais traite tous les autres phénomènes mentionnés. L'intérêt de PEGASUS est de proposer une méthode de calcul des structures prédicat-argument en post-traitement, ce qui le distingue des procédures couramment employées dans d'autres systèmes de TALN.

La composante (5) porte sur les relations sémantiques. Cette composante fait apparaître les liens sémantiques cachés dans les relations syntaxiques. Pour cela il va falloir, entre autres, rassembler les structures sémantiques paraphrastiques. Pour ce faire on modifie, à l'aide d'une "grammaire de concept" le résultat obtenu avec PEGASUS (la structure prédicat-argument). La tâche de la grammaire de concept est de construire un réseau (bien fondé) dans lequel les relations sémantiques sont établies entre des noeuds de concepts. Cette grammaire est un ensemble de procédures écrites en PLNLP qui accomplissent, sous certaines contraintes, un certain nombre d'opérations sur les graphes. Les arcs de l'analyse sont étiquetés à l'aide de noms de relations eux-mêmes dérivés de manière systèmatique de la combinaison de la syntaxe et de la sémantique du texte d'entrée. Les règles de cette grammaire sont similaires, pour ce qui est de leur forme, aux règles des composantes antérieures, mais elles opèrent sur différents aspects de l'information commune aux structures, analysant les relations entre les noeuds du graphe de la phrase, normalisant les structures sémantiques et les relations lexicales d'une variété de domaines syntaxiques, sans pour autant perdre accès à la structure de surface (contenant les différences syntaxiques). Cet article montre comment, partant de la structure prédicat argument obtenue en sortie de PEGASUS, la grammaire produit des graphes sémantiques tout en préservant la caractéristique du système global : large couverture et indépendance du domaine.

1. Derivation of logical form (PEGASUS)

We present PLUS (Progressive Language Understanding System), an integrated NLP analysis system. In its current state, PLUS consists of six components, roughly described as: (1) syntax (PEG, the PLNLP English Grammar); (2) corrected syntax (reassignment of constituents); (3) derivation of logical form (PEGASUS); (4) sense disambiguation; (5) normalization of semantic relations; and (6) paragraph (discourse) model. The current system architecture is sequential, because this makes it easier to concentrate on developing techniques for processing truly unrestricted input. However, this control structure is expected to become more parallel in the future.

The purpose of the third component, PEGASUS, is to simplify the derivation of a semantic representation, or logical form, for each input sentence or sentence fragment. To do this it computes: (a) the structure of arguments and adjuncts for each clause; (b) NP (pronoun)-anaphora; (c) VP-anaphora (for elided VPs). Simultaneously it must maintain broad coverage (that is, accept and analyze unrestricted input text). More commonly in NLP systems, the computation of such meaning structures is considered impossible unless a particular domain is specified.

Consider the sentence, "After dinner, Mary gave a cake to John." Figure 1 shows the syntactic (tree) representation for that sentence after it has been processed by the first two analysis components, and Figure 2 shows the semantic graph produced by PEGASUS for the same sentence:

```
-----------------------------------------
DECL1|----PP1|-------PREP1-----'After'
     |       |-------NOUN1*----'dinner'
     |       |-------PUNC1-----','
     |----NP1|--------NOUN2*----'Mary'
     |----VERB1*-----'gave'
     |----NP2|-------DETP1------ADJ1*----'a'
     |       |-------NOUN3*----'cake'
     |----PP2|-------PREP2-----'to'
     |       |-------NOUN4*----'John'
     |----PUNC2------'.'
-----------------------------------------
```

Figure 1. Syntactic parse tree

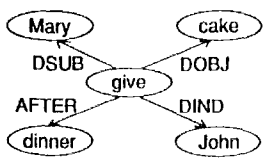


Figure 2. Semantic graph for the sentence in Figure 1

A graph is produced by displaying only those attributes and values that are defined to be semantic. However, the underlying record structure contains all attributes resulting from the parse. In this fashion, all levels and types of information, from morphological to syntactic to semantic and beyond, are constantly available. This principle of accountability holds throughout the PLUS/PLNLP system.

In a NLP system that uses attribute-value pairs, argument structures can be produced (a) by defining, for each node, attribute names that correspond to the desired argument or adjunct types, and (b) by assigning values to those attributes. It is customary to think of argument names like AGENT, PATIENT, etc. However, although these labels are tantalizingly semantic in nature, there is as yet no uniformly acceptable way of relating syntactic structure to them. Therefore we avoid such labels, at least for the time being. We adopt, instead, the notion of "deep" cases or functional roles:

DSUB:	deep subject
DIND:	deep indirect object
DOBJ:	deep object
DNOM:	deep predicate nominative
DCMP:	deep object complement

All deep argument attributes are added to the analysis record structure by PEGASUS. For very simple clauses, deep arguments correspond exactly to the surface syntactic arguments. For example, in "John ate the cake," the NP "John" fills the roles of both surface and deep subject; "the cake" fills the roles of both surface and deep object. In such simple cases, the deep argument attributes could as well have been assigned by the syntax rules; they are assigned by PEGASUS just to simplify the overall system architecture.

Each major class node is examined, and, if it contains more than just one single (head) word, each associated word is evaluated for possible assignment to some deep-structure attribute. In addition to the deep case labels, the following non-syntactic, non-argument attributes define the fully elaborated structure:

PRED:	predicate (basic term) label
PTCL:	particle in two-part verbs
OPS:	operator, like demonstratives and quantifiers
NADJ:	adjective modifying a noun
PADJ:	predicate adjective
PROP:	otherwise unspecified modifier that is a clause
MODS:	otherwise unspecified modifier that is not a clause; also, members of a coordinated structure

And in addition to these, attributes are defined to point to adjunct prepositional phrases and subordinate clauses. The names of these attributes are actually the lemmas of those prepositions and conjunctions that begin their phrases and clauses. In this fashion, a step is taken toward a more semantic analysis of these constituents, without the necessity of going all the way to case labels like "locative" and "durative."

The procedure starts by renaming the surface arguments in all cases, as described previously. Then it calls a set of sub-procedures, each one of which is designed to solve a particular piece of the argument puzzle. Here is an outline of the flow of control taken for the specification of arguments and adjuncts:

1. Assign arguments and modifiers to all VP nodes:
   A. Assign arguments, in this order:
      1) Unbounded dependencies, e.g., in "What did Mary say that John ate?" the DOBJ of "ate" is "What."
      2) Functional control, e.g., in "John wanted to eat the cake," the DSUB of "eat" is "John."

3) Passives, e.g., in "The cake was eaten by John," the DSUB is "John" and the DOBJ is "the cake."

4) Indirect object paraphrases, e.g., the structure for "Mary gave a surprise to John" must be identical to the structure for "Mary gave John a surprise."

5) Indirect object special cases, e.g., in "I told the story," the syntactic object "the story" is the DOBJ; but in "I told the woman," the syntactic object "the woman" is the DIND.

6) Extraposition, e.g., "John ate the cake" is the DSUB of the sentence "It appears that John ate the cake."

B. Assign modifiers (all adjuncts): prepositional, adjectival and adverb phrases; adverbial noun phrases; subordinate clauses; infinitives; comment clauses; participial modifiers; sentential relative clauses; etc.

2. Assign modifiers (including arguments) to all NP nodes.
3. Assign modifiers to all AJP (adjective phrase) nodes.
4. Assign modifiers to all AVP (adverb phrase) nodes.
5. Clean up the attribute-value structure by deleting some unwanted features.

The focus of linguistic interest here is on the assignment of arguments to VP nodes. Ordering of the sub-procedures is important. Long-distance dependencies must be resolved before functional control is assigned, and both of these maneuvers must be performed before passives are handled. The ordering presented here was experimentally determined by parsing sentences that contain more than one of the phenomena noted.

Subcategorization features on verbs are used more strictly here than they are used in the first component, the broad-coverage syntactic sketch. Also, although selectional features were not found to be useful in constructing the syntactic sketch, they are both useful and necessary for defining deep arguments in PEGASUS. With unbounded dependencies, it is important to distinguish the probable subcategorization types of verbs in the sentence, and also some selectional ("semantic") features on nouns, since the argument structure will vary depending on the interplay between these two pieces of information.

The sub-procedure for functional control handles not only infinitive clauses, but also participial clauses, both present and past. These constructions often require argument assignment over long intervening stretches of text. In the sentence "Mary, just as you predicted, arrived excitedly waving her hands," "Mary" is DSUB of the present participle "excitedly waving her hands." In the sentence "Bolstered by an outpouring of public confidence, John accepted the post," "John" is DOBJ of the past participle "Bolstered by an outpouring..."

All of the other sub-procedures for argument assignment are linguistically interesting to various degrees, but none of them is quite so complex as the procedures for unbounded dependency and functional control.

See Segond and Jensen 1991 for an explanation of the assignment of NP- and VP-anaphora, a discussion of advantages to using a post-processor, and a comparison of PEGASUS with other current strategies for deriving predicate-argument structures.

2. Semantic normalization

Semantic relations are represented by a graph. The nodes of the graph contain words; but, since these are linked with dictionary definitions, synonyms, and other related words, it is possible to say that these nodes represent concepts.[1] It is the job of the concept grammar to construct a well-motivated network in which semantic relations are properly drawn among concept nodes.

In order to do this job, one of the important problems that has to be addressed is the problem of showing equivalences between paraphrases. This problem was first approached by PEGASUS, where, for example, both active and passive forms of a clause are provided with the same argument structure. The work is continued by the concept grammar, and expanded to handle a much wider set of paraphrase situations. The basic intuition remains the same, however: different sentences that have essentially the same meaning (truth-value) will have the same semantic graph. And the same principle of accountability applies here as there: the system will always have access to the original surface syntactic variability, so that no nuances of expression need ever be lost.

As an example, all of the following sentences have the same essential meaning, and therefore should be associated with the same semantic graph: "There is a blue block"; "The block is blue"; "The block is a blue block"; "The block is a blue one." These are not classical syntactic variants, like active and passive; but they are variants of the same semantic facts: a block exists, and it is blue.

The sentences are analyzed by the syntax and PEGASUS. (Because our descriptive sentences are purposely kept very simple, we can avoid using the second and fourth components, reassignment and sense disambiguation.) The result is a graph for each sentence, corresponding to the basic arguments and adjuncts of that sentence. The concept grammar examines each sentence graph, checking for certain configurations that signal the presence of common underlying conceptual categories. Here is where the syntactic variants will be normalized.

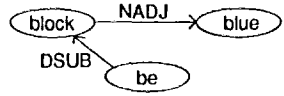
The operation of the concept grammar can be compared to the operation of a syntactic grammar: syntax takes words and phrases, and links them, via common morpho-syntactic relationships, into a structural whole; the concept grammar takes arguments and adjuncts, and links them, via common semantic relationships, into a conceptual whole. Syntax works with syntactic category labels; the concept grammar works with semantic arc labels.

2.1. The "block" sentence paraphrases

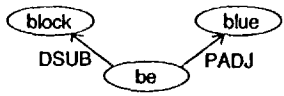
Consider the four "block" sentences above. The argument and adjunct structures (sentential graphs) provided by PEGASUS for these sentences, and shown in Figure 3, use just four semantic arc labels: DSUB, NADJ, PADJ, and DNOM (see above):[2]

[1] See Sowa 1984 for an introduction to conceptual graph structures.

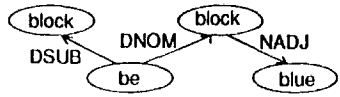
[2] Although only the head lemmas are displayed in the graph nodes, the underlying record structure keeps access to all syntactic details, such as determiners, tense, etc.



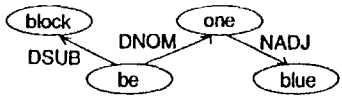
"There is a blue block."



"The block is blue."



"The block is a blue block."



"The block is a blue one."

Figure 3. Sentential graphs for the "block" sentences

These four sentential graphs are quite different; but, since the sentences have the same meaning, there should be just one semantic graph for all of them:

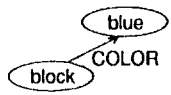


Figure 4. Canonical semantic graph for the sentences in Figure 3

This is a case of paraphrase that requires normalization. In order to achieve it, first of all we delete the node "be" in all graphs. It is well known that the English copula "be" carries very little semantic weight.

RULE 1: Delete the copula "be."

Second, if an adjective carries a lexical feature that marks it as a "color" word, then we change the arc label NADJ to the label COLOR. The effect is to change the name of the relation between the noun and the adjective.

RULE 2: Change NADJ from node with "color" word to COLOR

To achieve the desired semantic graph for "There is a blue block," we apply Rule 1 and Rule 2, deleting the node "be" and changing the name of the relation between the node "block" and the adjective "blue."

When the predicate is an adjective (PADJ), there is, in the argument structure, no direct relation between the subject (DSUB) and the adjective (PADJ). Both of them are attributes of the node "be." In this case, we create a new relation, NADJ, between the subject and the adjective, and delete the relation PADJ. (We will deal later with the difference between predicative (PADJ) and attributive (NADJ) adjectives.)

RULE 3: Create NADJ arc between subject and predicate adjective.

Once this new arc is created, rules 1 and 2 will recognize that the adjective is a "color" word, change the name of the relation NADJ to COLOR, and delete the node "be." These operations will turn the sentential graph for "The block is blue" into the desired semantic graph in Figure 4.

When the predicate is a noun or a noun phrase (DNOM), as in the remaining two "block" sentences, we have to ask if that predicate nominative is the same term as the subject (or is an equivalent empty anaphoric term, like "one"), or if it is different from the subject, and not empty. In the first case we unify the subject and the predicate NPs. All the nodes which point to the first are made to point to the second, and vice versa. Once this is done, the problems of the color adjective and of the empty copula are automatically handled by existing rules, and the sentential graphs for the last two "block" sentences are transformed into the canonical graph in Figure 4.

RULE 4: Unify subject and predicate under appropriate conditions.

In the second case, when there is a DNOM that is different from the subject NP, we create a new relation between the subject and the predicate. In the simplest case, we give this relation the ISA label:

RULE 5: Create ISA link under appropriate conditions.

Hence the sentence "The block is an object" has the following semantic graph:

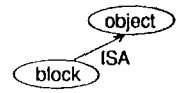


Figure 5. Semantic graph for "The block is an object"

The reader should not conclude from the previous examples that dealing with paraphrases requires a lot of ad hoc solutions. On the contrary, the rules (or procedures) of the concept grammar are general in nature. They identify and represent typical semantic relations in a formal way. A syntactic grammar does the same thing, but at a different level of structure. The concept grammar tries to catch what might be called "the semantics of the syntax." These operations are straightforward, just as the operations that build constituent structure in a syntactic grammar are straightforward. But this simplicity should not obscure the elegance of what is going on here. With minimal effort, using easily accessible parse information, we are automating the creation of a conceptual structure. This conceptual structure will ultimately have a high degree of abstractness, generality, and language independence.

2.2. Locative prepositional phrases

Consider the following set of sentences, which should all have the same semantic graph (Figure 6):

(1)

There is a **blue** block **on** the **red** block.
The blue block is **on** the red block.
There is a red block **under** the blue block.
The red block is **under** the blue block.

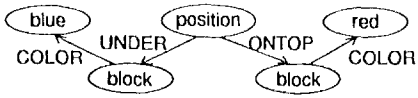


Figure 6. Canonical graph for sentences in (1)

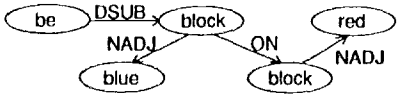


Figure 7. Sentential graph for "There is a blue block on the red block"

Note the graph node labeled "position." This word was never used in the paraphrase sentences, but the concept was implicit in all of them. (The link between preposition names and the word/concept "position" can be validated in dictionaries and thesauri.) One interesting and significant result of setting out to normalize these paraphrases is the emergence of what might be called the essential meaning of the expressions, namely, a statement of the relative position of two objects. In this fashion, the writing of a concept grammar results naturally, and pragmatically, in the emergence of terms that we might want to consider as "semantic primitives." It should be emphasized, however, that we are not committed beforehand to any basic conceptual or semantic primitives. In this example, the relations ONTOP and UNDER appear in the canonical graph of the sentence, but this is just for purposes of the present exposition. What we are interested in is to establish an appropriate link between the two blocks. Instead of ONTOP and UNDER we could have ABOVE (or ON) and BELOW, etc.

It is not necessary to discuss the treatment of each of the paraphrases. The first sentence in (1) will serve as an example. Figure 7, above, shows its sentential graph.

What we want to do is to link the deep subject ("blue block") with the object of the preposition ("red block") by using the relation names ONTOP and UNDER, which spring from the concept POSITION. We delete the copula "be," and create the new node POSITION, motivated by dictionary definitions for locative prepositions. Then we add two attributes, ONTOP and UNDER, to this node (pointing respectively to the subject and the noun phrase object of the preposition), and delete the attribute ON in the list of attributes of the subject. Notice that if the sentence read "above" instead of "on," the treatment would be the same.

Of course, this does not mean that looking at the syntactic relations between words is enough; the semantics of the words themselves are also important. For instance, the kind of relation involved between a subject NP and the NP object of a PP in the case of a locative prepositional phrase (e.g. the cat is *in* the garden, the cat is *under* the table), is not the same as the one involved with the PP which is a part of the sentence "The cat is *in* love." But still, in all these three sentences, what we are interested in is building the relation between "the cat" and the NP object of the PP (garden, table, love). Giving a name to the relation (and, for that purpose, knowing that love is a concept, garden is a place, and table is an object) is the task of the sense disambiguation component, which consults dictionary definitions to find the necessary semantic information.

2.3. Relative clauses

One way of combining propositions (the block is blue, is on the table, etc.) into one sentence is to use a relative clause. We can say:

(2) (a) The block that is blue is on the table.
(b) On the table is the block that is blue.
(c) The block, which is on the table, is blue.

Figure 8 shows the sentential graph for (2a). The attribute PROP points to the semantic structure of the relative clause "that is blue," and the attribute REF identifies the referent of the relative pronoun "that":

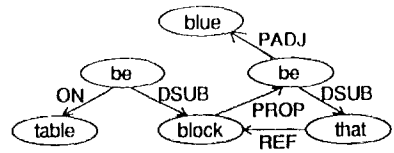


Figure 8. Sentential graph for "The block that is blue is on the table" (2a)

In the sentences of (2), we want to relate the deep subjects of the relative clauses with their predicates. All we have to do, in this case, is to unify the DSUB of the PROP with the REF of the DSUB of the PROP, deleting the REF attribute. The result is a record, pointed to by PROP, which has a DSUB identical to the DSUB of the whole sentence, and therefore possesses both the attributes that it gains from the relative clause, and the attributes of the DSUB of the whole sentence. Now the system is able to handle recursively all the other problems (copula, predicate adjective, and spatial prepositional relationships), and we obtain the same graph as is obtained for sentences such as "The blue block is on the table" or "There is a blue block on the table":

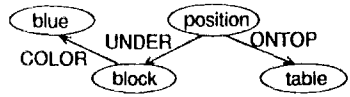


Figure 9. Canonical semantic graph for the sentences in (2)

2.4. Toward the discourse model

Our work also involves normalizing across sentence boundaries. For instance, from (3a–b):

(3) (a) The blue block is on the red block.
(b) The red block is on the black block.

we want to be able to infer (3c–d):

(3) (c) The blue block is above the black block.
(d) The black block is below the blue block.

Inference across sentence boundaries does not differ, in essence, from inference within a single sentence; after

all, two sentences may become one sentence, under coordination:

(3) (a AND b) The blue block is on the red block AND the red block is on the black block.

From an implementation point of view, the strategy is the same. We consider all nodes called "position." There is one such node in the graph for (3a), and another in the graph for (3b). We look at the records for both "position" nodes and obtain two lists: one, a list of all ONTOP attributes; and the other, a list of all UNDER attributes. We look at the intersection of those lists. If they have an element in common (for instance, in the previous example, "red block" will appear in both of them), then we know that we can infer the graph in Figure 10:

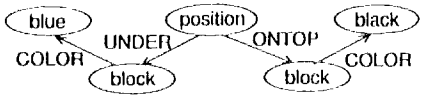


Figure 10. Inferential graph for (3c–d)

Figure 10 displays only the inferences in (3c–d), derived from (3a–b). But the system does not lose access to information about the existence and placement of the red block mentioned in (3a–b).

All the examples given in this paper involve sentences with the verb "be." "Be" and other state verbs comprise a complicated and interesting class. They accept a lot of different constructions (adjectival predicates, nominal predicates, prepositional phrase complements, etc.), and provide a convenient and convincing field for preliminary investigations. At the same time, much of the work done for state verbs (coordination, PP relationships, etc.) can be applied to other verb classes.

3. Conclusion

We hope to have made two substantial contributions in this paper: (1) to suggest a novel method for computing argument structures in a post-processor, in order to simplify the derivation of logical forms for sentences; (2) to show the birth of a concept grammar, which receives syntactic and semantic information from earlier stages of the system, and automatically provides a grammatical foundation for the next stage, discourse. We dealt with some linguistic problems, including different kinds of paraphrases. We also suggested methods for handling logical properties of natural language, such as the spatial properties of prepositions. (See Segond and Jensen 1991 for additional constructions handled by the concept grammar.)

Dealing with locative prepositions is not the same as dealing with the whole of natural language. However, we have tried to avoid specific or ad hoc solutions. The rules of the concept grammar are generic in nature. They express semantic facts about English (and, in some cases, about language in general), just as a morpho-syntactic grammar expresses syntactic facts about English. Therefore they are in no way restricted to a semantic subdomain.

This structure of very general relations is one of the steps leading to an ideal semantic representation of sentences. It provides a universal representation, independent from the surface structure but without losing the information contained in the surface structure.

Another contribution of the paper is to illustrate how this approach leads to an articulated architecture for a natural language understanding system. The architecture provides both modularity and integration of NLP tasks, and allows for a smooth flow from syntax through semantics to discourse. Starting with an initial syntactic sketch, we obtain a conceptual graph step by step, without adding a lot of hand-coded semantic information in the dictionary.

Acknowledgments

We are grateful to all the people who have helped us. Among these, we acknowledge here especially the following: George Heidorn, who provided us with tools and advice; Joel Fagan, who initialized the concept grammar work (see Fagan 1990); and Wlodek Zadrozny, with whom we have had lively and interesting conversations about semantics. Of course, any errors in this work remain our responsibility.

References

Chanod, J.-P., B. Harriehausen, and S. Montemagni. 1991. "Post-processing multi-lingual arguments structures" in *Proceedings of the 11th International Workshop on Expert Systems and Their Applications*, Avignon, France.

Fagan, J.L. 1990. "Natural Language Text: The Ideal Knowledge Representation Formalism to Support Content Analysis for Text Retrieval" in P.S. Jacobs, ed., *Text-based Intelligent Systems: Current Research in Text Analysis, Information Extraction and Retrieval.* GE Research and Development Center, Technical Information Series, 90CRD198, pp. 48–52. (Originally presented at AAAI 1990 Spring Symposium on Text-based Intelligent Systems, March 27, 28 & 29, 1990, Stanford University, Stanford, California, USA).

Heidorn, G.E. 1972. "Natural Language Inputs to a Simulation Programming System." PhD dissertation, Yale University, New Haven, Connecticut, USA.

Jensen, K., G.E. Heidorn and S.D. Richardson. 1992, forthcoming. *Natural Language Processing: the PLNLP Approach.* Kluwer Academic Publishers, Boston, Massachusetts, USA.

Segond, F. and Jensen, K. 1991. "An Integrated Syntactic and Semantic System for Natural Language Understanding." IBM RC 16914, T.J. Watson Research Center, Yorktown Heights, New York, USA.

Sowa, J.F. 1984. *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley Pub. Co., Reading, Massachusetts, USA.