Semantics-first Natural Language Processing

Steven L. Lytinen
Artificial Intelligence Laboratory
The University of Michigan
Ann Arbor, MI 48109

Abstract

There is no consensus on how syntax and semantic/pragmatics should interact in natural language processing. This paper focuses on one issue concerning interaction: order of processing. Two approaches are compared empirically: an interleaved *syntax-first* approach, in which semantic interpretations is performed at intermediate points during parsing; and a *semantics-first* approach, in which semantic considerations drive the rule selection process during parsing. The study provides empirical evidence that the semantics-first approach is more efficient than the syntax-first approach in processing texts in narrow domains.

Introduction

Over the last decade, many researchers in Natural Language Processing (NLP) have begun to realize that semantic and pragmatic information must have more influence on parsing. The reason for this is that many syntactic ambiguities cannot be resolved without reference to semantics or pragmatics. As a result, there have been many recent efforts to integrate higher-level analysis into parsing. Although the way in which integration is achieved in different systems varies widely, perhaps the dominant approach has been one in which syntax and semantics[1] are *interleaved*; i.e., semantic analysis is performed at intermediate points during the parse, thus allowing it to filter out semantically amomalous constituents before syntactic processing is complete. The purpose of doing this is to make processing more efficient, since semantics prunes those parses with no semantic interpretation before syntax can build further on them. Examples of systems using the interleaved approach include SHRDLU (Winograd, 1972), PARAGRAM (Hirst, 1988), TRUMP (Jacobs, 1987), Grishman and Sterling's (1989) system, and unification-based systems such as PATR-II (Shieber, 1986)[2].

In this paper, I will present an empirical comparison between the interleaved syntax-first approach and an alternative, which I will call the *semantics-first* approach. The semantics-first approach is also interleaved, in that syntactic and semantic processing alternate. However, this time semantic processing is done first: semantic attachments between constituents are proposed using world knowledge, and then grammar rules are searched for which can realize the desired semantic attachments. This approach was originally implemented in a system called MOPTRANS (Lytinen, 1986), and has been re-implemented for the purposes of running the empirical comparison in a unification-based NLP system called LINK (Lytinen, in press).

From a functional standpoint, the syntax-semantics interaction issue is essentially one of efficiency. Given the same linguistic knowledge, different methods of integration should, at least in theory, produce the same results given the same input. However, the speed at which these results are produced might vary widely depending on the organization of this knowledge, and the order in which it is applied. As is often the case in AI, efficiency is not just an implementational detail, as different strategies may result in algorithms which are orders of magnitude different in their performance. This could mean the difference between a working system and one which takes hours or days to process a single sentence.

The claim of this paper is that the semantics-first approach is more efficient than the interleaved syntax-first approach. This claim is supported by the empirical study, in which the semantics-first implementation of LINK and an interleaved syntax-first version were

[1] For the sake of brevity, I will use *semantics* to refer to the traditional linguistic concept of semantics, or knowledge about the meanings of words, as well as *pragmatics*, or knowledge about the world and about how language is used.

[2] The division between modules in PATR-II is somewhat different than in other systems: phrase structure information remains in the "syntactic" module and is used to guide parsing, while other syntactic information resides in the unification module. Although Shieber did not address semantics in (Shieber, 1986), others have used this framework to incorporate semantic information into the unification module (e.g., HPSG, Pollard and Sag, 1987).

both used to process randomly selected sentences from two different corpora.

The paper is organized as follows: first I present a brief discussion of the unification grammar which is used in LINK. Then I discuss the two implementations of the system, and how they differ from one another. Finally, the empirical comparison is presented.

LINK's Unification Grammar

The grammar rules used in LINK are very similar to those in PATR-II (Shieber, 1986). They are encoded as unification *constraint rules*, each of which consists of a set of equations. Here is a simplified example of a constraint rule:

```
S: (1) = NP                              <1>
   (2) = VP                              <2>
   (head) = (2 head)                     <3>
   (head agr) = (1 head agr)             <4>
   (head subj) = (1 head)                <5>
```

Each equation in this rule specifies a property which any node labeled S must have. A property consists of a *path*, or a sequence of arcs with the appropriate labels starting from the node in question; and a *value*, which is another node to be found at the end of the path. Equations specify the values of properties in one of two ways. They may specify the label of the node to be found at the end of the path, as in equations 1 and 2 (i.e., the arc from an S node labeled 1 leads to a node labeled NP). We will call these *labeling equations*. Or, they may specify that two paths must lead to the identical node, as in equations 3-5. Identity here is defined by the *unification* operation; i.e, if two paths must lead to the identical node, then the nodes at the end of the two paths must unify. Unification merges the properties of two nodes; thus, two paths can unify if their values have no properties which explicitly contradict each other. These equations will be called *unifying equations*.

Functionally, the above rule encodes information about English sentences as follows. Equations 1 and 2 specify that a sentence is made up of two subconstituents: a NP and a VP. Ordering of these constituents is implicit in the numbering of the paths. Equation 3 assigns the HEAD of the sentence to be the VP, by unifying the VP's HEAD path with the HEAD path of the S. This will be discussed further shortly. Equation 4 specifies that the NP and the VP must agree in number and person. These syntactic properties are found under the AGR (agreement) feature of each constituent. Finally, equation 5 assigns the NP to be the subject of the sentence.

The HEAD property referred to in equations 3-5 is used to propagate information up and down the parse structure. This is accomplished by unification of HEAD links, as in equation 3. Because of this equation, any information on the HEAD of the VP is accessible from the S node. Similar equations would assign the heads of other constituents, such as a verb (V) to be the HEAD of the VP, and a particular lexical item to be the HEAD of the V.

Lexical items typically provide the values which are propagated by HEAD links. They are encoded using *lexical rules*, which look slightly different from constraint rules. Here is an example of a lexical entry:

```
eats: V
   (head agr number) = sing                    <6>
   (head agr person) = 3rd                     <7>
   (head rep) = EAT-FOOD                       <8>
   (head subj rep) = (head rep actor)          <9>
   (head dobj rep) = (head rep object)) <10>
```

Typical values provided by lexical rules include syntactic feature information, such as the AGR feature (eqs. 6-7); as well as semantic information, which causes a semantic representation of the sentence to be built as parsing proceeds (eq. 8). Lexical entries also can contain mappings from syntactic to semantic dependencies (eqs. 9-10). In this case, "eats" specifies that whatever constituent fills the SUBJECT role will also be assigned as the ACTOR of the EAT-FOOD, and that the syntactic direct object (DOBJ) will be assigned as the semantic OBJECT.

One more type of knowledge remains to be specified. Equations 9 and 10 are used in conjunction with the system's domain knowledge, to impose restrictions on the semantic properties (i.e., the values of the REP path) of the subject and direct object of "eats" (i.e., the ACTOR and OBJECT of EAT-FOOD). This type of knowledge is also encoded in constraint rules. In this particular case, the rule which encodes the relevant world knowledge is the following:

```
EAT-FOOD:
   (actor)  = ANIMATE                      <11>
   (object) = FOOD                         <12>
   (instrument) = UTENSIL                  <13>
```

Because of the mapping provided by "eats" between its subject and the actor of EAT-FOOD, the restriction that this constituent's representation must be ANIMATE is propagated to the NP which fills the SUBJ role specified by equation 5. Similarly, the FOOD restriction on the object of EAT-FOOD would propagate to the NP assigned as the direct object (DOBJ) of "eats."

Syntax-first vs. Semantics-first

The syntax-first and semantics-first versions of LINK share a great deal. Both implementations use the same grammar rules, as outlined above. Both are also implemented as bottom-up chart parsers[3], which perform the unification operations specified by a grammar rule whenever that rule is applied. Thus, the system builds,

[3] see (Winograd, 1987) for a good description of chart parsing.

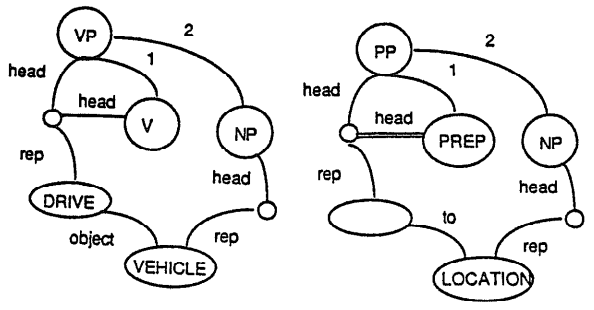

Figure 1: Consituents constructed at an intermediate point during the parse of "John drove the car into the city."

in parallel, a chart which represents the syntactic constituents found thus far in the sentence, as well as directed acyclic graph (DAG) representations of semantic and additional syntactic information about each constituent, as specified by the unification equations in the grammar.

The difference between the syntax-first and semantics-first versions of LINK is the way in which it is determined which grammar rule should be applied next during parsing. As the names would imply, syntactic information is used to select a rule in the syntax-first version, whereas semantic information is used in the semantics-first version. To illustrate, let us consider a simple example:

John drove the car into the city.

Assuming that processing has proceeded to the point where the constituents in figure 1 have already been constructed, we will compare the rule selection process for the syntax-first and semantics-first approaches at this point.

In the syntax-first version of LINK, processing is virtually identical to the PATR-II system (Shieber, 1986). The labels of the root nodes of DAGs are the only information used to select a new rule. These labels are syntactic categories, such as NP, VP, etc. A grammar rule is applied whenever a sequence of labels is found in the chart which matches, in the right order, all of the constituents specified in the phrase structure component of that rule; that is, all of the constraint equations whose left hand sides are numbers. In our current example, two rules are found which can be applied:

```
VP:
   (1) = VP                          <14>
   (2) = PP                          <15>
   (head) = (1 head)                 <16>
   (head rep) = (2 head rep)         <17>

NP:
   (1) = NP                          <18>
   (2) = PP                          <19>
   (head) = (1 head)                 <20>
   (head rep) = (2 head rep)         <21>
```

This is because the constituents "drove the car" and "into the city" match equations 14 and 15, and "the car" and "into the city" match equations 18 and 19. Syntax-first LINK attempts to apply both of these rules. The VP rule succeeds, because of the following information about driving:

```
DRIVE:
   (actor)    = HUMAN                <22>
   (object)   = VEHICLE              <23>
   (from)     = LOCATION             <24>
   (to)       = LOCATION             <25>
```

Since "city" qualifies as a LOCATION, "into the city" can be attached to "drove the car." However, since physical objects (such as VEHICLE) cannot have a TO relation attached to them, the NP rule fails. Thus, syntax-first LINK attempts to apply two rules, one of which succeeds.

In the semantics-first version of LINK, the rule selection process is driven by semantics. Desirable semantic attachments between adjacent constituents in the chart are identified, using the system's domain knowledge (encoded in rules such as equations 11-13 and 22-25). In this particular example, only one desirable attachment is found: the one between DRIVE and LOCATION ("city"). The LOCATION can fit as either the FROM or the TO of DRIVE, according to equations 24-25. After desirable attachments are identified, LINK tries to find any grammar rules which would cause the attachment to take place. In order for this to happen, a rule must be found which contains a unifying equation which would unify the appropriate paths. In this case, there are two possible paths, because of the two possible attachments:

Path associated w/VP	Path associated w/PP
(head rep from)	(head rep to)
(head rep to)	(head rep to)

This is because LOCATION (the representation of "city") is in the (head rep to) path of the PP (see figure 1). A rule must be found which contains a unifying equation which would cause one of these pairs of paths to be unified. The VP rule (eqs. 14-17) is found. This is because equation 17 will unify the (head rep to) path of the VP with the (head rep to) path of the PP, the second possibility from above. The rule is applied, and "into the city" is attached to the VP "drove the car."

Notice that semantics-first LINK never considers the potential attachment of "into the city" to the NP "the car." This is because semantics did not find any potential connections between these two constituents. In

Automobile Repair domain

VEHICLE STALLS ON HOT DAYS.
CEL FLASHES ON ACCEL, BOTH HOT AND COLD, BACKFIRES, WANTS TO STALL.
ENGINE CUTS OUT/STALLS, NO RESTART HOT INTERMITTENT, CODE 42 STORED, ALSO DIFFICULT COLD STARTING.
ENGINE QUITS AT CRUISING SPEEDS, FUEL PUMP FUSE IS BLOWN.

Assembly Line domain

WALK TO FRONT OF JOB
GET & READ BODY TAG TO VERIFY SERIAL #
WALK TO FRONT DOOR, TOSS INSPECTION RECORD IN JOB
CHECK NEXT JOB FOR STYLE, COLOR, CF5

Figure 2: Examples of texts from the two corpora

a syntax-first parser, both possible syntactic interpretations must be constructed, upon which the interpretation in which the PP is attached to the NP would immediately be discarded as semantically anomalous. This illustrates why, intuitively, one might expect the semantics-first approach to be more efficient than the syntax-first approach.

The Empirical Comparison

An empirical test was conducted which compared the syntax-first and semantics-first versions of LINK. In the test, both versions were used to process single-sentence texts from two different corpora. These corpora were selected because we had already developed extensive sets of grammar rules and lexical entries for them. One corpus consisted of descriptions of symptoms displayed by automobiles in need of repair. The other corpus consisted of short descriptions of sequences of actions to be performed on an assembly line. Examples of descriptions from both corpora are shown in Figure 2. Our work on the automobile repair domain is described in detail in (Lytinen, 1990).

For both domains, the coverage provided by the grammar and lexicon was quite good. In (Lytinen, 1990), we reported that LINK was able to correctly process about 70% of new descriptions, provided they did not contain new vocabulary items. We have achieved approximately the same success rate in the assembly line domain. Thus, coverage the grammar and lexicon for the two corpora are such that arbitrary sentences from each can be processed with reasonable accuracy by the knowledge bases which we developed.

For the empirical comparison, 50 sentences were chosen at random from each corpus and processed by the syntax-first and the semantics-first versions of LINK. Processing time was compared both in terms of number of grammar rules that the system tried to apply, and CPU time necessary to process each sentence.

Auto Repair Domain		
Approach	# of rules	CPU time[4]
Syntax-first	63	10308
Semantics-first	46	8143
Assembly Line Domain		
Approach	# of rules	CPU time
Syntax-first	52	4973
Semantics-first	36	5166
Grammatical Auto Repair Examples		
Approach	# of rules	CPU time
Syntax-first	101	20392
Semantics-first	61	13017

Figure 3: Average performance of two approaches on the three test corpora

Approach	Best fit polynomial	R^2
Automobile repair corpus		
Syntax-first	$y = -6.3526 + 1.8297x + 0.17007x^2$	0.652
Semantics-first	$y = -20.980 + 5.3412x$	0.630
Assembly line corpus		
Syntax-first	$y = -7.2023 + 8.0611x$	0.542
Semantics-first	$y = 5.2928 + 4.0449x$	0.570
Grammatical automobile repair corpus		
Syntax-first	$y = 14.499 - 2.4595x + 0.39376x^2$	0.766
Semantics-first	$y = -5.2954 + 3.8007x$	0.645

Figure 4: Best fit polynomial for # of rules applied during processing vs. sentence length for the three corpora

Approach	Best fit polynomial	R^2
Automobile repair corpus		
Syntax-first	$y = -449.93 + 827.41x - 130.82x^2 + 8.8058x^3$	0.676
Semantics-first	$y = -514.34 + 25.987x + 47.863x^2$	0.701
Assembly line corpus		
Syntax-first	$y = -1530.4 + 859.69x$	0.517
Semantics-first	$y = 435.37 + 666.72x$	0.437
Grammatical automobile repair corpus		
Syntax-first	$y = -2836.3 + 1375.0x - 117.41x^2 + 5.2987x^3$	0.878
Semantics-first	$y = 160.45 + 57.638x + 34.201x^2$	0.830

Figure 5: Best fit polynomial for CPU time vs. sentence length for the three corpora

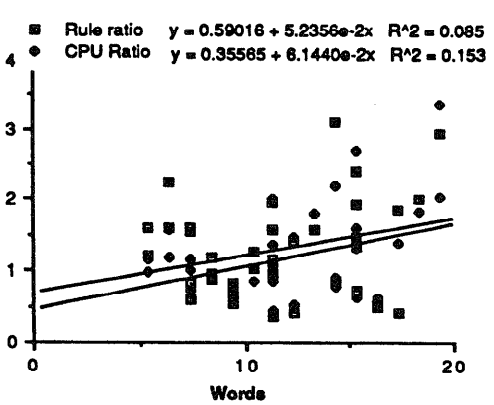

Figure 6: Processing ratios vs. sentence length for the automobile repair corpus

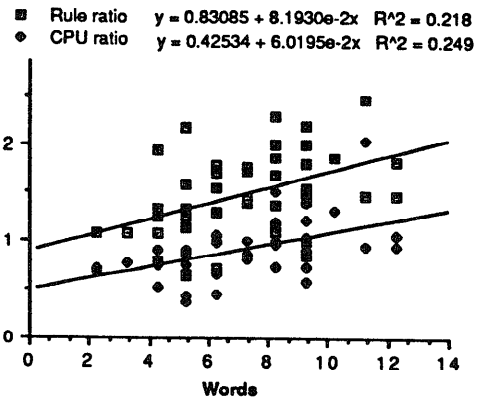

Figure 7: Processing ratios vs. sentence length for the assembly line corpus

As can be seen from examining examples in figure 2, both corpora consist mainly of text which does not conform to standard English grammar. The texts are terse, with determiners missing from noun phrases, incomplete sentences, or lists of incomplete sentences separated by commas. To ensure that the peculiarities of our corpora did not affect the results of the comparison, a third test was conducted on more standard sentences. A native English speaker was given the set of 50 examples from the automobile repair domain and asked to rewrite the examples using standard English grammar. He did not know for what purpose this was being done. LINK's grammar for this domain was also modified, so as to make it more standard. Then the comparison between the syntax-first and semantics-first approaches was run again.

Average results for the three tests are presented in Figure 3. The differences in average number of rules and CPU times is significant for the automobile repair corpus (# of rules: $t(48)=1.78$, $p<.05$; CPU: $t(48)=1.69$, $p=.05$). Number of rules is also significantly higher for the syntax-first approach in the assembly line domain than the semantics-first approach ($t(49)=5.44$,$p=.0001$), though CPU time is significantly lower for the syntax-first approach ($t(29)=-1.70$, $p=.05$). This may be a reflection of the short average sentence length of this domain. Finally, for the grammatical examples, average number of rules and CPU times were significantly higher for the syntax-first approach (# of rules: $t(45)=3.95$, $p=.0003$; CPU time: $t(45)=2.70$, $p=.01$).

Results also indicate that the average-case complexity of the syntax-first approach is significantly worse than for the semantics-first approach. First, the number of rules applied during processing of a sentence was plotted against sentence length for each of the three corpora. Figure 4 shows the best polynomial curve fit for each corpus, according to a weighted R^2 analysis. Figure 5 shows the best fits for plots of CPU time vs. sentence length for each corpus. Although the data are very noisy, in general a higher-order polynomial is required to fit the syntax-first data than the semantics-first data.

Average-case complexity of the two approaches was also compared by plotting the ratio of syntax-first processing to semantics-first processing vs. sentence length, for both number of rules appied and CPU time. If the two algorithms had the same complexity, this ratio should remain constant. However, as the graphs in figures 6-8 indicate, the ratio tends to increase with sentence length, indicating a higher-order complexity for the syntax-first approach. Least squares analysis of each dataset indicates a straight line with increasing slope in all instances. The largest effect was seen on the grammatical automobile repair corpus. For this test, the slopes of the best-fit lines were highly significantly greater than 0 (Number of rules: $F(1,43) = 34.44$, $p<<.0005$; CPU time: $F(1,43) = 53.33$, $p<<.0005$). Slopes were also significantly greater than 0 for the assembly line corpus (Number of rules applied: $F(1,48)=13.08$, $p=.0007$; CPU time: $F(1,48) = 15.50$, $p=.0003$). For the original automobile repair corpus, slopes were positive, but the deviation from 0 was not significant (Number of rules applied: $F(1,44)=1.64$, $p=.207$; CPU time: $F(1,44)=2.77$; $p=.103$).

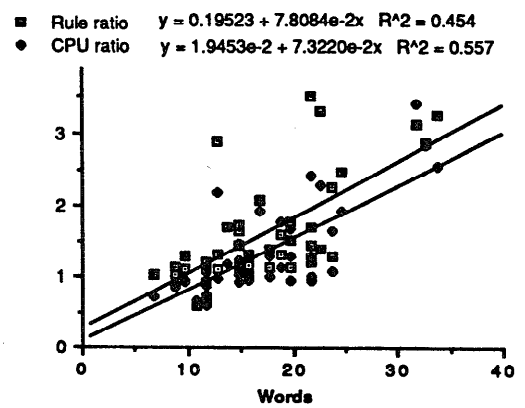

Figure 8: Processing ratios vs. sentence length for the grammatical automobile repair examples

Conclusion

The results of the empirical comparison indicate that, at least for the domains selected for the test, the semantics-first version of LINK was significantly more efficient than the syntax-first version. Not only was the semantics- first version an average of 26% faster over the three tests, but analysis also suggested that the average-case complexity of the syntax-first approach was worse than for the semantics-first approach.

The corpora selected for the comparison contained material from very limited domains. One might question if the results of the comparison were affected by the narrowness of the domains. It is possible to think of reasons why expanding domains might affect the results in either direction. Favoring the semantics-first approach would be the fact that the variety of syntactic constructions encountered in these narrow domains was quite limited. Even the grammatical texts produced by our native English speaker did not exercise the full variety of syntactic constructions available in English. Working with a broader domain would likely widen the variety of syntactic constructions encountered, thus increasing the number of potential syntactic ambiguities that would need to be resolved. This could greatly detract from the efficiency of the syntax-first method, by increasing the number of meaningless syntactic interpretations constructed. On the other hand, the semantics-first approach might be hurt by broadening the domain by increasing the number of candidate semantic attachments which would be found. This factor would be limited by the fact that the semantics-first approach only considers attachments between adjacent constituents; however, an increase in attachments found could possibly narrow the gap between the semantics-first approach and the syntax-first version.

In any case, the question of whether these results would hold for wider domains is somewhat of a moot point. Presently available theories of natural language semantics are taxed to their limit just to handle arbitrary texts from very narrow domains. Whatever the approach, it is beyond the ability of the field to build NLP systems which can understand arbitrary texts from relatively unrestricted domains to any degree at all. The results reported in this paper indicate that for applications in narrow domains, the semantics-first approach appears to be significantly more efficient than the interleaved syntax-first approach.

References

Hirst, G. (1988). Semantic interpretation and ambiguity. *Artificial Intelligence, 34*, pp. 131-178.

Grishman, R., and Sterling, J. (1989). Towards robust natural language analysis. In *Proceedings of the AAAI Spring Symposium on Text-based Intelligent Systems*, Palo Alto, CA, March 1990, pp. 106-108.

Jacobs, P. (1987). A knowledge framework for natural language analysis. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 675-678.

Lytinen, S. (in press). A unification-based, integrated natural language processing system. To appear in *Computers and Mathematics with Applications.*

Lytinen, S. (1990) Robust processing of terse text. In *Proceedings of the 1990 AAAI Symposium on Intelligent Text-based Systems*, Stanford CA, March 1990, pp. 10-14.

Lytinen, S. (1986). Dynamically combining syntax and semantics in natural language processing. In *Proceedings of the Fifth National Conference on Artificial Intelligence,* Philadelphia PA, pp. 574-578.

Pollard, C., and Sag., I. (1987). *Information-based Syntax and Semantics.* Menlo Park, CA: Center for the Study of Language and Information.

Shieber, S. (1986). *An Introduction to Unification-based Approaches to Grammar.* CSLI, Stanford CA.

Winograd, T. (1972). *Understanding Natural Language.* New York: Academic Press.

Winograd, T. (1987). *Language as a Cognitive Process. Vol. 1: Syntax.* Reading, MA: Addison-Wesley Publishing.