Computers Math. Applic. Vol. 23, No. 6–9, pp. 403–418, 1992
Printed in Great Britain.

0097-4943/92 $5.00 + 0.00

A UNIFICATION-BASED, INTEGRATED NATURAL LANGUAGE PROCESSING SYSTEM

STEVEN L. LYTINEN
Artificial Intelligence Laboratory, The University of Michigan
Ann Arbor, MI 48109, U.S.A.

Abstract—This paper presents a natural language processing (NLP) system called LINK. LINK is unification-based, and incorporates and extends many features which have been emerging from other NLP research in recent years. In particular, the notions of *autonomous syntax* and *compositional semantics*, long staples of NLP systems, have been replaced by a grammar which is much more complex, semantics-oriented, and more reliant on idiomatic constructions; and a semantics which is noncompositional. Processing, also, has been changed from the traditional syntax-driven approach, to an approach which relies much more heavily on semantics and domain knowledge, represented in a semantic net. As a result, LINK is able to efficiently process ungrammatical sentences, as well as nonliteral constructions such as metaphor and metonymy. These tasks have been difficult for more traditional NLP systems.

1. INTRODUCTION

Recently, the status quo in natural language processing (NLP) has been changing from its traditional generative grammar foundation. This foundation carried with it several assumptions, which have turned out not to be optimal. In particular, two of these assumptions were the following:

1. *Autonomy of syntax.* In the traditional approach, syntax was viewed as an autonomous entity, with little or no interaction with semantics/pragmatics. This was true in two senses: first, processing was performed separately, with minimal interactions between parsing and semantic interpretation. Second, grammars were written with little or no reference to semantic information; that is, grammatical categories and rules had little correspondence with semantic categories.
2. *Compositionality of semantics.* Construction of a semantic interpretation of an utterance was relatively straightforward, based on combining the semantic interpretations of lexical items.

This view of language processing dominated NLP for many years, at least until the early 1980's. Good examples of systems which adhere to this approach are Winograd's [1] SHRDLU program, Marcus' [2] parser, and Hirst's [3] PARAGRAM. Even connectionist systems have adopted many of these assumptions. These include Waltz and Pollack's [4] "massively parallel" parser, in which a chart parse of the input was initially performed in order to construct some of the links used in spreading activation; and McClelland and Kawamoto's [5] case analyzer, based directly on Fillmore's [6] case grammar. In some of these systems, such as PARAGRAM, semantic interpretation was interleaved with syntactic analysis so as to prune semantically anomalous parses more efficiently, but grammars still remained largely devoid of semantic information, and syntactic and semantic processing were still performed by separate modules.

Changes in the status quo, however, can be seen in more recent systems and linguistic theories. These include HPSG [7], Cognitive Grammar [8,9], phrasal parsers such as PHRAN [10] and RINA [11], and theories of metaphor (e.g., [12,13]). Although it is not clear that the developers of these programs/theories would lump themselves together, their approaches share a commmon set of assumptions, which are quite different from the traditional paradigm:

1. *Idiosyncratic syntax.* Rather than use a "pure" grammar which contains rather general syntactic rules which are devoid of semantics, these approaches are based on the assumption that grammatical rules must refer to semantic information in order to properly constrain them. The result is a more complex, and idiosyncratic, set of grammatical rules, based largely on phrases, figures of speech, and idiomatic constructions.
2. *Noncompositionality of semantics.* Because of the impure separation between syntax and semantics, and the idiosyncratic nature of many grammar rules, the compositionality of semantics does not hold. Often it is difficult to point to a single lexical item and delineate the contribution which that particular item makes to the semantics of the utterance of a whole.

Thus, in HPSG, semantic interpretation rules are interwoven in the grammar, resulting in quite complex unification-style grammar rules. Likewise, Cognitive Grammar proposes very narrow and complex grammatical rules, often based on semantic categories. In PHRAN and RINA, semantic interpretations are built out of complex phrasal constructions rather than lexical items. And in theories of metaphor, the assumptions of compositionality are violated by the use of mapping rules which construct intended meaning from literal meaning.

Following in this direction, this paper will discuss the LINK system, a unification-based NLP system which extends these notions further. LINK's grammar rules are inextricably intertwined with its semantics and domain knowledge. This is of necessity: semantic constraints must be expressed in grammar rules in order to (a) adequately restrict the domain of these rules; and (b) build a semantic interpretation in tandem with syntactic structures. Grammar rules are also often quite complex, reflecting the idiosyncracies of meaning of various constructions. As a result, semantics in LINK is noncompositional, in that often it is impossible to delineate the contributions to a semantic interpretation made by individual lexical items. Whole constructions, sometimes interacting with particular lexical items, are often responsible for the pieces which combine together to form a complete semantic interpretation.

LINK's semantic and domain knowledge are represented in a semantic net. The network can be used to drive processing, enabling LINK to naturally process ungrammatical texts. Knowledge in the semantic net is used to find plausible semantic attachments between constituents in a sentence. In the case of ungrammatical sentences, the plausible attachments can result in the application of a grammar rule, even when syntactic properties of the constituents do not exactly match the rule. This will be discussed further in Section 4. The semantics-driven approach to processing ungrammatical sentences makes LINK quite distinct from other unification-based parsers, such as PATR-II [14]. We argue that the semantics-driven approach to processing ungrammatical sentences is much more efficient than other syntax-driven approaches to this problem (e.g., [15,16]).

LINK also incorporates the use of mapping rules, similar to those proposed by Lakoff and Johnson, Martin, and others. These rules enable LINK to process nonliteral constructions, such as metaphor and metonymy. Mapping rules in effect propose an alternative, nonliteral interpretation for a constituent whenever its literal interpretation satisfies a set of semantic and/or syntactic constraints. As we will see, these rules are also of use in analyzing the meanings of idioms and figures of speech, which are not normally classified as nonliteral constructions.

2. LINK'S UNIFICATION GRAMMAR

Unification grammar was originally developed to encode syntactic constraints in natural languages, such as subject-verb agreement. However, it has recently been used to encode semantic information also (e.g., [7]). We follow this approach, encoding all linguistic knowledge used by the system in unification-style constraint rules.

2.1. The Basics

As with all unification-based parsers, the data structure which LINK builds to represent its analysis of a sentence is a Directed Acyclic Graph (DAG). DAG nodes may or may not be labeled; arcs must have a label.

```
<constraint-rule> ::= <label>: <equation> ... <equation>

<equation> ::= <labeling> | <unifying>

<labeling> ::= <path> = <label>

<unifying> ::= <path> = <path>

<path> ::= (<arc-label> ... <arc-label>)

<arc-label> ::= <label> | !(<arc-label> ... <arc-label>)
```

Figure 1. Definition of a constraint rule in LINK.

Linguistic knowledge in LINK is expressed mainly in terms of *constraint rules*, which consist of sets of equations. Each equation constrains the DAG in some way, by limiting, for a class of nodes (i.e., any node with a particular label), the set of arcs that can lead from a node of that class, as well as the types of nodes that arcs can lead to. The definition of a constraint rule is shown in Figure 1.

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*.

Constraint rules can also be represented as DAGs, whose nodes and links are labeled. The DAG form of the above rule is given in Figure 2. The root node of this DAG is labeled S, because that is the class of constituent to which this rule applies. For each property specified in the rule, there is a corresponding path in the DAG leading from the root node to a node with the appropriate label (in the case of labeling equations), or there are two paths from the root node which lead to the same node (in the case of unifying equations). Thus, in Figure 2, a node labeled NP is found at the end of the arc labeled 1 coming from the S node.

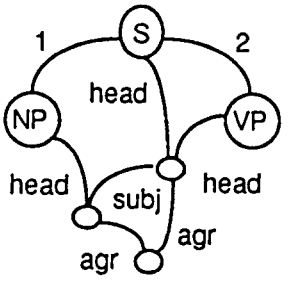

Figure 2. DAG version of the S rule.

```
<lexical-rule> ::= <label>: <label> <equation> ... <equation>

<equation> ::= <labeling> | <unifying>

<labeling> ::= <path> = <label>

<unifying> ::= <path> = <path>

<path> ::= (<label> ... <label>)
```

Figure 3. Definition of lexical rules in LINK.

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. The definition of a lexical rule is shown in Figure 3.

Typical values provided by lexical rules include syntactic feature information, such as the **AGR** feature, as well as semantic information, which causes a semantic representation of the sentence to be built as parsing proceeds. Here is an example of a lexical entry:

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

Equations (6)-(7) specify the word's syntactic features, found under the **AGR** property. Equation (8) provides semantic information about the word, specifying that "eats" means **INGEST**, the conceptual dependency primitive meaning to take a substance into the body [17]. Equations (9) and (10) specify mappings from syntactic to semantic dependencies. Equation (9) states that whatever constituent fills the **SUBJECT** role will also be assigned as the **ACTOR** of the **INGEST**. Similarly, Equation (10) specifies that the syntactic direct object (**DOBJ**) is 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 the **INGEST**). 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:

```
INGEST:
        (actor) = ANIMATE                    <11>
        (object) = FOOD                      <12>
        (through) = BODYPART                 <13>
```

This is an encoding of the sort of case information that many people have proposed, such as Schank's conceptual dependency theory or Fillmore's case grammar [6]. It defines LINK's semantic network, with nodes labeled **INGEST**, **ANIMATE**, **FOOD**, etc., and arcs labeled **ACTOR**, **OBJECT**, etc. Other rules provide similar definitions for fillers like **ANIMATE**, if additional information is required for these fillers.

Because of the mapping provided by "eats" between its subject and the actor of the **INGEST**, the restriction that this constituent's representation must be **ANIMATE** is propagated up to the **NP** which fills the **SUBJ** role specified by Equation (5). Similarly, the **FOOD** restriction on the object of an **INGEST** would propagate to the **NP** assigned as the direct object (**DOBJ**) of "eats," because of Equations (10) and (12). The direct object would be supplied by the following constraint rule:

```
VP:
        (1) = V                              <14>
        (2) = NP                             <15>
        (head) = (1 head)                    <16>
        (head dobj) = (2 head)               <17>
```

2.2. Some Complicating Factors

The rules we have presented so far are too simple, for two reasons. First, we have specified that the actor of an **INGEST** must be **ANIMATE**. However, in other situations we might like to make distinctions between types of animate beings; for example, we might want to define "John" as a **HUMAN**:

```
John: N
        (head rep) = HUMAN                   <18>
        (head rep gender) = MALE             <19>
        (head rep name) = "john"             <20>
```

This suggests the introduction of an inheritance hierarchy among the labels used in the system, as is found in most semantic networks. For example, we want to define **HUMAN** as a type of **ANIMATE**. But since traditional unification requires an exact match of node labels, nodes labeled **HUMAN** and **ANIMATE** will not unify, and we will not be able to successfully parse a sentence such as "John eats."

To accomodate the use of inheritance hierarchies in our semantic network, we follow the approach of Aït-Kaci [18]. Rather than require identical labels on nodes, our unification algorithm allows nodes to have different labels, as long as there is an IS-A relationship between one label and the other. The result of unification of two such nodes is a node with the more specific label. Thus, unifying a node labeled **ANIMATE** with one labeled **HUMAN** results in the label **HUMAN** on the node in the unified DAG.

Another difficulty with the rules as we have presented them thus far is that they are not general enough to allow for processing of passive sentences. We would like to reformulate them so that we do not need a duplicate set of rules for passive sentences, and other constructions. We accomplish this by introducing a level of indirection into paths, indicated by a "!". With this modification, rules are presented in Figure 4 which will process the following sentences:

John ate a sandwich.

A sandwich was eaten by John.

```
S:
   (1) = NP
   (2) = VP
   (head agreement)
          = (1 head agreement)
   (head rep !(head subject-slot))
          = (1 head rep)


VP:
   (1) = VG
   (2) = NP
   (head) = (1 head)
   (head type) = TRANSITIVE
   (head rep !(head object-slot)) =
          (2 head rep)

VP:
   (1) = VG
   (head) = (1 head)
   (head type) = INTRANSITIVE

VP:
   (1) = VP
   (2) = PP
   (head) = (1 head)
   (head !(2 word)) = (2 head rep)

VG: ;for active verbs
   (1) = V
   (head) = (1 head)
   (head type) = (1 type)
   (head subject-slot) =
          (1 subject-slot)
   (head object-slot) =
          (1 object-slot)

VG: ;for passive verbs
   (1) = BE
   (2) = PAST-PART
   (head) = (2 head)
   (2 type) = TRANSITIVE
   (head type) = INTRANSITIVE
   (head rep !(2 object-slot)) =
          (head by)

PP:
   (1) = PREP
   (2) = NP
   (head) = (2 head)
   (word) = (1 1)

NP:
   (1) = DET
   (2) = N
   (head) = (2 head)

NP:
   (1) = N
   (head) = (1 head)


V:
   (1) = eats
   (head) = (1 head)
   (type) = (1 type)

PAST-PART:
   (1) = eaten
   (head) = (1 head)
   (type) = (1 type)

eats: V
   (type) = TRANSITIVE
   (head number) = SINGULAR
   (head person) = 3RD
   (head rep) = INGEST
   (head rep through) = BODYPART
   (head rep through type) = MOUTH
   (subject-slot) = ACTOR
   (object-slot) = OBJECT

eaten: PAST-PART
   (type) = TRANSITIVE
   (head number) = SINGULAR
   (head person) = 3RD
   (head rep) = INGEST
   (head rep through) = BODYPART
   (head rep through type) = MOUTH
   (subject-slot) = ACTOR
   (object-slot) = OBJECT

John: N
   (head rep) = HUMAN
   (head rep gender) = MALE
   (head rep name) = "john"

sandwich: N
   (head rep) = FOOD

a:  DET

by:  PREP
```

Figure 4. Rules for active and passive sentences in LINK.

The indirect paths ("!") are interpreted as meaning that the label of the node pointed to by the path should be used as an arc label in the path at that point. For example, whatever is the label of the node in the (**head subject-slot**) path is used in the S equation as the third arc to follow in the path (**head rep !(head subject-slot)**).

In these rules, we have introduced verb groups to include the use of auxiliaries such as in passive sentences. If the verb does not have the passive auxiliary, then the verb group's SUBJECT-SLOT and OBJECT-SLOT are passed up from the verb. These slots are then used as variables, so that the values of the SUBJECT-SLOT and OBJECT-SLOT are used as the link names under which to place the representations of the subject and direct object of the sentence, respectively (the final equations for S nodes and VP nodes). However, if the sentence does contain the passive auxiliary, then the second set of VG equations will be satisfied, equating the SUBJECT-SLOT of the VG with the OBJECT-SLOT of the verb, and the old SUBJECT-SLOT of the verb with the object of the preposition "by." The latter is accomplished through the equation unifying this slot with the node at the end of the BY link on the VP, which is in turn unified with the object of the preposition "by" through the prepositional phrase constraint rules given above. These work by creating a link labeled WORD from the PP node to the actual prepostion in the sentence (i.e., "by"). Then, this word is used in the final VP rule above, in this case resulting in the semantic representation of the object of "by" to be pointed to by a BY link coming off of the head of the VP.

2.3. The Line Between Semantics and Pragmatics

So far, we have seen the encoding of phrase structure information, syntactic dependencies, and semantic dependencies encoded in our unification framework. A comment is in order about semantic information. It is difficult to make a clear distinction between "semantics" and "pragmatics" in this approach. Notice that in the lexical entries of verbs such as "eats," we only have a mapping between syntactic and semantic/pragmatic dependencies. There is no use of selectional restrictions or semantic features *à la* [19]. Instead, this information is provided by our separate entry for what an INGEST ought to look like. We can think of this information as "world knowledge," since it is separate from our lexicon, and describes restrictions about what happens in the world. Transfers of control are done between animate agents, and the transfers are performed on physical objects. This is not a linguistic fact, as it could easily be used in non-linguistic activities such as planning or problem solving.

This is an important point, and illustrates another advantage of abandoning the modular approach of previous natural language systems. If we compare our knowledge base in our system, in which all knowledge is encoded in the same way, with rules in modular systems, we discover that knowledge in the modular systems was duplicated from module to module. The information about INGEST's having animate agents performing an act upon a physical object would be duplicated several times: perhaps in the syntactic module, so that syntactic interpretations would not be found which violated these constraints; certainly in the semantic component, so that ambiguities could be resolved; and finally in the pragmatic component, where this information would be necessary to perform inferences. In our approach, however, it is only necessary to encode this knowledge once, in a "world knowledge" (i.e., pragmatic) rule. Since this information is readily integrated in with syntactic and semantic information through our unification-style mapping rules, the duplication is no longer necessary.

3. PARSING

A grammar in LINK consists of a set of lexical and constraint rules.[1] If a label used in the grammar has no rules explicitly defined for it, then the grammar is assumed to contain a rule for that label with no constraints; i.e., the following rule:

<label>:

[1] A grammar can also contain a set of mapping rules, which will be discussed in Section 5.

LINK succeeds in finding an interpretation of a sentence when it has produced a DAG that satisfies the following conditions:

1. The root node is labeled S.
2. For each lexical item in the sentence, the DAG contains a node whose label is that lexical item. If a lexical item appears multiple times in the sentence, there must be multiple nodes in the DAG.
3. Every node corresponding to a lexical item can be reached from the root node via a path whose arc labels are all numeric.
4. For any two nodes corresponding to two adjacent lexical items, if a single node has numerically labeled arcs which lead to both lexical items, then the numbers on the labels of these arcs must reflect the ordering of the words; i.e., the word on the left has a smaller-numbered arc.
5. Every node in the DAG *satisfies* a lexical or constraint rule.

A node satisfies a constraint rule if it is *subsumed* by the subdag whose root is that node; that is, if the unification of the DAG representation of the constraint rule and the subdag whose root is that node is that subdag. Lexical rules are satisfied in much the same way. A lexical rule is really a shorthand for two constraint rules:

```
<label1>:<label2> <equation>...<equation>
```

is equivalent to

```
<label1> :
   (1) = <label2>
   (head) = (1 head)

<label2>:<equation>...<equation>
```

In order to produce a DAG which satisfies these conditions, LINK is implemented in the style of a bottom-up chart parser.[2] In this way, it is similar to the PATR-II system [14]. However, in LINK, unification is completely integrated with chart parsing. This was not the case with PATR-II, in which the chart parsing and unification were performed by separate modules. As we will see in Section 4, this is especially important to the ability of LINK to process ungrammatical inputs.

Whereas links in a standard chart are labeled with a syntactic category, in LINK's chart they are labeled with a DAG, representing all the syntactic and semantic features which have been assigned to each constituent. New links are added to LINK's chart by applying unification rules. These rules are indexed by subconstituents; i.e., by the values of paths which have numeric labels. Whenever a sequence of constituents in the chart is found which unifies with the corresponding sequence in a rule, that rule is applied. Unification guarantees that each constituent from the sentence has the syntactic and semantic features specified by the rule, or at least is compatible with those features.

To see more clearly how this works, consider the noun phrase "a man." Let us assume the following lexical entries for these words:

```
a: DET
   (head number) = sing                    <21>
   (head rep ref) = indefinite             <22>

man: N
   (head number) = sing                    <23>
   (head rep) = HUMAN                      <24>
   (head rep gender) = MALE                <25>
```

[2] See [20] for a description of chart parsing.

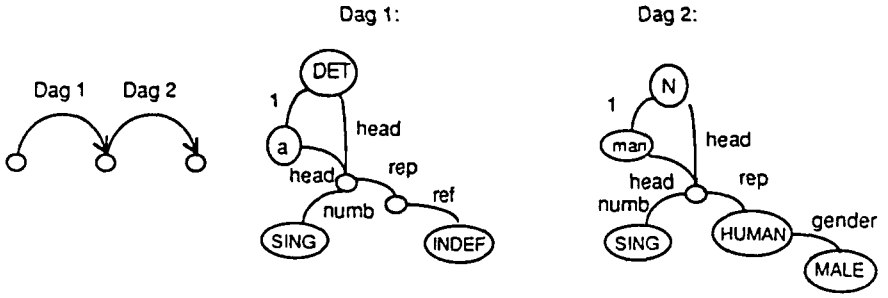

Figure 5. Initial chart for "a man."

When parsing begins, LINK constructs the chart in Figure 5.[3] Thus, the initial chart contains nodes which satisfy the lexical rules for "a" and "man." Let us assume the following constraint rule for noun phrases:

```
NP:   (1) = DET                          <26>
      (2) = N                            <27>
      (head) = (2 head)                  <28>
      (head rep) = (1 head rep)          <29>
      (head common) = (1 head common)    <30>
```

Assuming the existence of this rule, the parser would construct an NP node which satisfies it, with an arc labeled 1 leading to the DET node from above, and an arc labeled 2 leading to the N node. This DAG satisfies the NP rule, since the DET and N nodes from above unify with the values of the 1 and 2 properties in the rule. Thus, the rule is applied, and a new link is added to the chart, with the result shown in Figure 6.

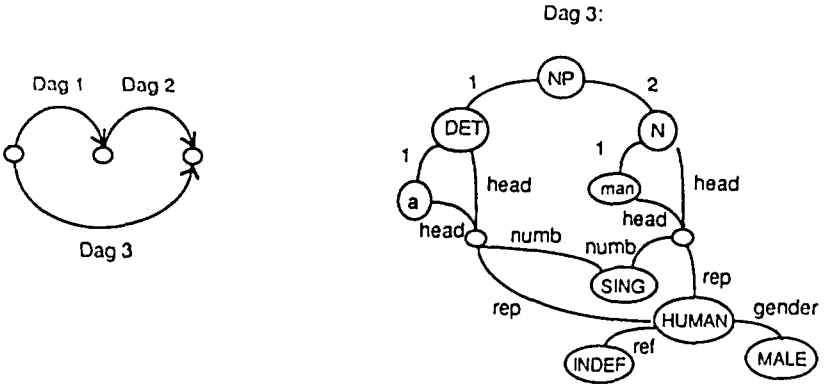

Figure 6. Chart after application of NP rule.

Eventually, assuming a grammatical sentence to begin with, this process will lead to the construction of an S node which satisfies the requirements stated earlier.

In this approach to parsing, processing of syntax, semantics, and pragmatics are completely integrated. This is because all properties are represented in the rules which must unify with constituents found in the sentence. Thus, a mismatch between semantic expectations and the semantics of constituents in the sentence would result in the failure to apply a rule.

Since processing is integrated, a semantic representation of the sentence is constructed as a by-product of the rule application process. Higher level nodes gather all of the semantic information from the nodes they span. In the above example, the fact that the reference to the HUMAN is INDEFINITE is added as a result of Equation (29).

[3] For ambiguous words, a separate link would be entered in the chart for each meaning.

4. PROCESSING UNGRAMMATICAL INPUTS

LINK's parsing algorithm has been extended to enable the system to process ungrammatical sentences; that is, texts which do not conform to the system's grammar rules. In this section, we will discuss the modifications to the parsing algorithm. We would argue that techniques for robust processing, such as ours, emerge from an integrated processing approach much more naturally than they would in traditional NLP systems.

LINK uses its semantic net to drive processing of ungrammatical inputs. This is possible because of LINK's integration of syntactic and semantic knowledge. To see how this is done, consider the following simple example:

John ate the sandwich quick.

"Quick" should be an adverb, "quickly," in order for this sentence to be grammatical. However, LINK is able to process this sentence, even if its grammar is written so as to only accept sentences to be modified by adverbs rather than adjectives.

Some lexical and constraint rules which are relevant to this input are the following:

```
S:
   (1) = S                              <31>
   (2) = ADV                            <32>
   (head) = (1 head)                    <33>
   (head rep) = (2 head rep)            <34>

quick:  ADJ
   (head rep speed) = RAPID             <35>

SPEED:
   (slot-of) = ACTION                   <36>
   (filler) = SPEED-QUALIFIER           <37>
```

Semantic information provided by modifiers such as adverbs are added to what they modify by unifying the REP of the two constituents. The modifiers themselves are defined as having a nonlabeled node in the (HEAD REP) path, with information hanging off of this node. Thus, unifying the two (HEAD REP) nodes, as in Equation (34), adds this information to the constituent which the adverb modifies.

The word "quick" is defined as modifying the SPEED property of whatever the word attaches to. This property is defined by a special type of constraint rule, which specifies the paths SLOT-OF and FILLER. These are special paths, which enforce constraints on the nodes which an arc connects together. The SLOT-OF arc is the node from which the arc originates, and the FILLER is the destination node of the arc. Thus, any arc labeled SPEED should lead from an ACTION to a SPEED-QUALIFIER. RAPID is a type of SPEED-QUALIFIER.

Processing of this input proceeds as follows: first, subconstituents are built up, using the system's normal grammar rules. LINK can identify "John ate the sandwich" as an S, and "quick" as an ADJ. At this point, parsing cannot proceed further.

When LINK fails to complete a parse, it tries to find desirable semantic attachments between adjacent constituents in the input, as dictated by the knowledge in its semantic network. In this case, attachments are considered between "John ate the sandwich" and "quick," as well as "the sandwich" and "quick" (since the NP "the sandwich" is an embedded constituent which is also adjacent to "quick"). Attachments are proposed using domain knowledge from the semantic net about INGEST, FOOD, and RAPID, the semantic interpretations of these constituents. In this case, there is a desirable attachment that can be made between "John ate the sandwich" and "quick," since INGEST and RAPID can be connected together by a SPEED arc (according to the definition of SPEED). Thus, based on its semantic net, LINK would like to attach "quick" to the complete sentence rather than just to "the sandwich."

Given this desired attachment, LINK searches its grammar for potentially relevant rules which would perform the attachment. One relevant rule is the adverb attachment rule, Equations (31)–(34). There is a "near match" between this rule and the constituents in the input: "John ate the sandwich" is an S, but "quick" is an ADJ rather than an ADV. Because adjectives and adverbs are functionally similar, though, this rule is applied, with the constraint from Equation (32) eliminated. The parse can then proceed, and the input is successfully processed.

This approach to processing of ungrammatical inputs is highly semantics-driven, since rules are selected on the basis of preferred semantic attachments. This is in sharp contrast to most other work on this problem, in which purely syntactic techniques have been explored (e.g., [16,21]). For limited domains, we believe that the semantics-driven approach is much more promising, since domain knowledge can often lead the system to the correct interpretation with very little search.

We have applied our approach to processing ungrammatical texts in a project called TASLINK [22], which processed terse free-form text descriptions of automobile stalling problems, and the repairs that fixed them. The TASLINK system successfully processed approximately 70% of texts in this domain. Over half of the texts were ungrammatical in some way, illustrating the success of this approach in a limited domain.[4]

5. LINK AND NONLITERAL CONSTRUCTIONS

5.1. LINK's Mapping Rules

Many types of constructions do not lend themselves well to the traditional compositional semantics which is prevalent in most NLP systems, and which LINK uses in what we have described thus far. Metaphor is one such phenomenon. Consider the following:

The stock market rose today.

Essential to the compositional approach to semantics is the notion that the meaning of a sentence can be constructed from the meanings of its pieces (either lexical items or phrases). However, if we combine the literal meanings of "stock market" and "rose," we obviously will not arrive at the correct interpretation of this sentence. We could define one or both of these terms as ambiguous, with "literal" and "metaphorical" senses, but it is not clear that we can easily separate the meaning of the metaphor into components to be associated with individual lexical items, as the compositional approach requires. For example, "rose" could be defined ambiguously, as either referring to an increase in altitude or an increase in a numeric indicator, but this does not capture the generalization that almost any word which refers to a change in altitude can be used in the same way. Consider the following examples:

The stock market plummeted today.

My fever has gone up.

Computer science enrollments are leveling off.

The 10% raise in property taxes this year was outrageous.

If we defined "rose" as ambiguous, we would have to define every other verb which referred to a change in altitude ambiguously, also. Even nouns would be ambiguous, as the final example illustrates.

This sort of phenomenon is very widespread in natural language. We will not attempt to argue this point here, as many others have presented convincing arguments as to the prevalence of metaphors and other nonliteral constructions (e.g., [8,12]).

The generality of the above metaphor, and the prevalence of the phenomenon in general, suggests that an alternate mechanism is needed to properly account for it. Syntactic flexibility and independence from particular lexical items suggests an approach based on *mapping rules*, or rules which map one semantic interpretation (the literal one) to another (the metaphorical

[4] See [22] for details.

one). In the case of this metaphor, the mapping is from the change in altitude of an object which has an associated numerical indicator, to the change in value of the indicator. The direction of the change in value is "the same" as the direction of the change in altitude: that is, increase in altitude means increase in the numerical indicator.

The use of mapping rules to account for metaphors (and other constructions, as we will see shortly) is also motivated by the ease with which people can understand many novel metaphors. This ease of comprehension is difficult to explain in the standard compositional paradigm. For example, in the last few years economists have begun talking about a "soft landing" of the economy. This conveys a rather complex process of economic growth slowly decelerating until it levels off at around zero (little or no growth), without slipping into recession (negative growth). This is a very complex concept. Yet despite its complexity, when this term was introduced into the American vocabulary, it seemed to require no explanation. A theory of metaphor comprehension based on lexical ambiguity cannot explain this, since "soft landing" previously had no metaphorical definition.

The definition of a mapping rule in LINK is shown in Figure 7. A <label> is any syntactic or semantic category used in the grammar. A <var> (variable) by convention begins with a '?', and the same set of variables appear in the left and right hand sides of a rule.

```
<mapping-rule> ::= <dag-spec> => <dag-spec>
<dag-spec> ::= <label> <equation> ...  <equation>
<equation> ::= <labeling>|<unifying>
<labeling> ::= <path> = <label>|<path> = <var>|<path> = (<var> <label>)
<unifying> ::= <path> = <path>
```

Figure 7. Definition of a mapping rule.

Mapping rules are interpreted to mean the following: whenever the parser builds a node with the appropriate label, and that node satisfies the constraints specified in the left-hand-side constraint list, then an alternate interpretation can be built; namely, a node satisfying the description on the right hand side. Variables indicate mappings between the values of properties of the original node and properties of the alternate node.

To illustrate, here is the mapping rule for the change-in-altitude metaphor:

```
CHANGE-ALTITUDE:                ⇒      CHANGE-VALUE:
 (object) = PHYS-OBJ                    (object) = ?X
 (object num-value) = ?X
```

Mapping rules have the effect of introducing an alternate interpretation, given a particular construction. Thus, the effect is the same as an ambiguous word definition, but it is accomplished using much more general rules. Alternate interpretations are introduced by adding an additional link to the chart, which is an appropriately modified copy of the link to which the original node is attached. Thus, in the case of the rule above, whenever LINK constructs a semantic interpretation consisting of the predicate CHANGE-ALTITUDE, whose object is a PHYS-OBJ, then an alternate interpretation is the predicate CHANGE-VALUE, with its object the numerical-value associated with the PHYS-OBJ.

5.2. Mapping Rules and Metonymies

Metaphors are not the only construction for which mapping rules are useful. They are also useful in specifying the interpretation of metonymies. Metonymy is another example of a phenomenon in which the underlying representation of a sentence is different from the "literal" meaning which might be computed directly from the meanings of words and their respective syntactic/semantic roles. For example:

Iran said today that its warships had shelled a Dutch supertanker.

It is not actually the country of Iran that performed the act of communication referred to in the sentence. Rather, it is probably a spokesperson for the country who performed this act. LINK's final representation of this sentence ought to reflect this inference. In fact, if LINK did not make the inference, it would fail to parse the sentence, since "Iran" would not meet the semantic requirements of the ACTOR of an MTRANS (the conceptual dependency primitive representing acts of communication).

To process this sentence, LINK uses a mapping rule which describes the general situation in which this inference should be made. The rule is the following:

```
MTRANS:                          ⇒   MTRANS:
 M(actor) = (?X NATION)               (actor) = HUMAN
                                      (actor type) = GOVERNMENT-OFFICIAL
                                      (actor nationality) = ?X
```

This rule specifies that whenever LINK attempts to build an MTRANS whose actor is a NATION, an alternate interpretation is also built with the actor a HUMAN whose nationality is the specified nation. In this case, the original literal interpretation is immediately rejected as being semantically anomalous, leaving the interpretation resulting from the application of the mapping rule as the only alternative.

Mapping rules do not always operate exclusively in the domain of semantics. Sometimes a certain syntactic construction is required for a mapping rule to apply. For example:

John drank his can of beer.

In this sentence, even though the direct object of "drink" is "can," the interpretation of the sentence should not be that John INGESTed a metal object. However, we need a more restrictive rule for this instance of metonymy, as it appears that it is only allowed for certain syntactic constructions:

*John drank his beer can.

*John drank his can.

These examples illustrate that the metonymy only applies when the substance inside the container is explicitly stated, and appears as the object of the preposition "of." The following equations specify the appropriate conditions for application of the mapping rule:

```
NP:                                   ⇒   NP:
 (head rep) = CONTAINER                    (head rep) = ?X
 (head rep contains) = ?X
 (head of) = (head rep contains)
```

We specify the contents of the container as the variable ?X. The final equation on the left hand side requires that the contents also be the object of the preposition "of." A set of equations attaching PP's to NP's, similar to the equations discussed earlier for the attachment of PP's to VP's, results in the building of a link labeled OF from the head of the NP to the FOOD node representing "beer." Thus, the existence of this equation in our mapping rule guarantees that the correct syntactic construction was used in order for this rule to apply. This time, the modification causes the two nodes labeled FOOD (one from the word "beer" and the other from the OBJECT restriction on an INGEST) to be unified, and placed in the role in which the CONTAINER was to be placed, since the (HEAD REP) path is set to be both of the FOOD nodes.

Once again, this rule introduces an alternate interpretation: it is still possible that the noun phrase refers to the container itself. Both interpretations are constructed, and then one or the other is eliminated by the surrounding context, or else LINK analyzes the sentence as being semantically ambiguous.

Adding rules for handling metonymy to our framework results in an additional advantage to the elimination of selectional restrictions, as discussed in Section 2.3. Traditionally, metonymy is

a difficult phenomenon to handle with selectional restrictions, since a restriction may very well be violated by the indirect reference. In a modular system, this presents problems, because the knowledge for resolving the reference is most likely in a module further along the processing line from the module which uses selectional restrictions. The only possible solution is to relax the selectional restrictions, letting some semantically unacceptable parses slip through until later so that instances of metonymy are not rejected as being ungrammatical. However, in LINK, since rules for handling metonymy are applied at the same time as all other rules in the system, this is not a problem. As soon as the failure in unification is found, the situation is resolved by failure equations. Thus, there is no need to adjust the restrictions on semantic roles in the same way that selectional restrictions would have to be loosened.

5.3. Mapping Rules and Idiomatic Expressions

Mapping rules are also used in LINK to process some types of constructions not usually thought of as nonliteral. Nonetheless, the mapping rule paradigm seems most appropriate for these constructions, because their level of generality requires a rule which operates on intermediate (i.e., "literal") semantic interpretations.

Consider the following examples:

John gave Mary a kiss.
John gave Mary a lecture.
John was given a beating by Mary.
John got a big welcome at the airport.

These are all examples of a common expression in English, in which the doing of some action to a recipient is expressed as "giving" the recipient that act. Just as with metaphors, this expression is quite flexible, independent of syntactic construction and relatively independent of the particular lexical item. Because of this flexibility, the use of mapping rules is suggested. The mapping rule for this expression is the following:

```
ATRANS:                    ⇒   ?X:
 (object) = (?X ACTION)          (!(?X object-slot)) = ?Y
 (recip) = (?Y PHYS-OBJ)
```

Recall the use of indirect paths as discussed in Section 2.2. Here it is assumed that the word expressing the main action (which is the direct object of the "giving" verb) will have an OBJECT-SLOT property, similar to the definition of "eats" in Section 2.2. This is necessary because the semantic case of the surface indirect object differs depending on the particular action. For example, in "John gave Mary a kiss," Mary is the OBJECT of the kissing, but in "John gave Mary a lecture," Mary is the RECIPIENT of the MTRANS (that is, Mary is not the topic of the lecture, but rather the recipient of it). Thus, the above mapping rule would interact with lexical rules such as the following:

```
kiss: N
 (head rep) = *KISS*
 (subject-slot) = ACTOR
 (object-slot) = OBJECT

lecture: N
 (head rep) = MTRANS
 (subject-slot) = ACTOR
 (object-slot) = RECIP
```

6. CONCLUSION

LINK is part of an emerging new paradigm in natural language processing, in which syntax and semantics are interconnected in complex and idiosyncratic ways. Rather than use a "pure," elegant set of grammar rules, followed by semantic interpretation, LINK's grammar rules are often inelegant, reflecting the idiosyncracies of idioms, figures of speech, metaphors, and other complex linguistic phenomena. Semantic interpretation is a by-product of syntactic analysis in LINK, the result of the need to use semantic constraints to express the limitations of idiomatic syntactic constructions.

This general approach to language processing has the following functional advantages:

- efficient processing through simultaneous application of syntactic and semantic constraints,
- natural extensions to robust processing, enabled by semantic indexing of grammar rules,
- ability to process metaphor, metonymy, and figures of speech, via mapping rules.

As was discussed in the introduction, many of the characteristics of LINK are derived from other systems which follow in this general paradigm. LINK further extends these characteristics in several ways. First, processing in LINK is completely integrated. Because of this, domain knowledge, stored in a semantic network, can be used to drive processing. This is highly useful in the case of processing ungammatical inputs. Intuitively, a semantics-first approach seems to be the most efficient way to search for an interpretation of an ungrammatical text, since the search space for possible interpretations based on syntactic information alone is likely to be much larger than based on desirable semantic attachments. This approach was successfully used in the highly limited domain of automobile repair descriptions, and should be applicable to many domain-restricted natural language processing tasks.

Second, LINK is the first system to combine the notion of mapping rules with phrasal grammars and idiosyncratic syntactic constructions. Other phrasal parsers, such as RINA and PHRAN, as well as unification-based systems, such as HPSG and PATR-II, have not used mapping rules; while systems using mapping rules, such as Martin's and Carbonell's [23], have been based on more traditional grammatical approaches. The use of mapping rules in LINK is also different from most previous systems, in that it is not *failure-driven*. Other systems using mapping rules, which generally accounted for the processing of metaphors, applied them only when literal semantic interpretation failed (although Martin's system is an exception to this). LINK's mapping rules, on the other hand, always apply, acting to introduce an alternative interpretation. Normal semantic processing then determines which interpretation is the correct one for a given context, just as is the case for a lexical ambiguity. As discussed in [24], this approach to the use of mapping rules is more compatible with the psychological data on processing of metaphor and other nonliteral constructions, which indicates that nonliteral interpretations take no longer for people to compute than literal meanings (e.g., [25–27]).

REFERENCES

1. T. Winograd, *Understanding Natural Language*, Academic Press, New York, (1972).
2. M. Marcus, *A Theory of Syntactic Recognition for Natural Language*, MIT Press, Cambridge, MA, (1980).
3. G. Hirst, Semantic interpretation and ambiguity, *Artificial Intelligence* 34 (2), 131–178 (1988).
4. D. Waltz and J. Pollack, Massively parallel parsing: A strongly interactive model of natural language interpretation, *Cognitive Science* 9 (1), 51-74 (1985).
5. J. McClelland and A. Kawamoto, Mechanisms of sentence processing: Assigning roles to constituents of sentences, In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 2, (Edited by D. Rumelhart and J. McClelland), pp. 272–325, The MIT Press, Cambridge, MA, (1986).
6. C. Fillmore, The case for case, In *Universals in Linguistic Theory*, (Edited by E. Bach and R. Harms), pp. 1–88, Holt, Rinehart and Winston, New York, (1968).
7. C. Pollard and I. Sag, *Information-Based Syntax and Semantics*, Center for the Study of Language and Information, Menlo Park, CA, (1987).
8. R. Langacker, *Foundations of Cognitive Grammar*, Vol. 1, Stanford University Press, Stanford, CA, (1987).
9. R. Langacker, A usage-based model, In *Current Issues in Linguistic Theory*, Vol. 50, (Edited by B. Rudzka-Ostyn), pp. 127–161, John Benjamins Publishing, Amsterdam, (1988).
10. R. Wilensky and Y. Arens, PHRAN: A knowledge-based natural language understander, In *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA, pp. 117–121 (1980).

11. U. Zernik and M. Dyer, The self-extending phrasal lexicon, *Computational Linguistics* **13**, 308–327 (1987).
12. G. Lakoff and M. Johnson, *Metaphors We Live By*, The University of Chicago Press, Chicago, IL, (1980).
13. J. Martin, Understanding new metaphors, In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, pp. 137–139, (1987).
14. S. Shieber, *An Introduction to Unification-Based Approaches to Grammar*, CSLI, Stanford, CA, (1986).
15. R. Weischedel and N. Sondheimer, Meta-rules as a basis for processing ill-formed input, *American Journal of Computational Linguistics* **9** (3/4) (1983).
16. C. Mellish, Some chart-based techniques for parsing ill-formed input, In *Proceedings of the 1989 Conference of the Association of Computational Linguistics*, Vancouver, B.C. (June, 1989).
17. R. Schank, *Conceptual Information Processing*, Lawrence Erlbaum Associates, Hillsdale, NJ, (1975).
18. H. Aït-Kaci, A new model of computation based on a calculus of type subsumption, Ph.D. Thesis, University of Pennsylvania (1984).
19. J. Katz and J. Fodor, The structure of a semantic theory, *Language* **39**, 170–210 (1963).
20. T. Winograd, *Language as a Cognitive Process, Vol 1: Syntax*, Addison-Wesley Publishing, Reading, MA, (1987).
21. W. Woods, Optimal search strategies for speech understanding control, *Artificial Intelligence* **18** (3) (1982).
22. S. Lytinen, Robust processing of terse text, In *Proceedings of the 1990 AAAI Spring Symposium on Text-based Intelligent Systems*, Palo Alto, CA, pp. 10–14 (March, 1990).
23. J. Carbonell, Metaphor: An inescapable phenomenon in natural language comprehension, In *Strategies for Natural Language Processing*, (Edited by W. Lehnert and M. Ringle), pp. 415–434, Lawrence Erlbaum Associates, Hillsdale, NJ, (1982).
24. S. Lytinen, S. Huffman and J. Kirtner, Towards a psychologically plausible computational theory of metaphor processing, Dept. of EECS, University of Michigan, Research report, Computer Science and Engineering Division.
25. R. Gibbs, Contextual effects in understanding indirect requests, *Discourse Processes* **2**, 1–10 (1979).
26. R. Gibbs, Do people always process the literal meanings of indirect requests?, *Journal of Experimental Psychology: Learning, Memory, and Cognition* **9** (3), 524–533 (1983).
27. A. Ortony, D. Schallert, R. Reynolds and S. Antos, Interpreting metaphors and idioms: Some effects of context on comprehension, *Journal of Verbal Learning and Verbal Behavior* **17** (4), 465–477 (1978).