In Tomek Strzalkowski (ed.) *Reversible Grammar in Natural Language Processing*, Kluwer Academic, 1994, 257-291.

Reversible NLP By Linking the Grammar To the Knowledge Base

David D. McDonald [1]

March 1993

We present a new reversible architecture for natural language processing (NLP). Separate parsing and generation grammars are constructed from the same representation of the language's linguistic resources as they are linked to objects in the knowledge base. By having two grammars, we are free to use process-specific representations and control techniques, thereby permitting highly efficient processing. At the same time, the single linguistic representation ensures the parsimony of development and competence that make reversible NLP attractive. This architecture is made possible by a construal of parsing that views the process as culminating not in a structural description but in a set of semantic objects—the same objects that the generation process starts from.

1 Author’s address: 14 Brantwood Road, Arlington, MA 02174-8004 (617)646-4124, Internet: MCDONALD@CS.BRANDEIS.EDU

1. Introduction

What is it that people know as native speakers of a natural language? There are many ways to couch an answer to this question, but given the perspective adopted in this paper, we will hold that it is that people have command of a body of *linguistic resources*: words, syntactic constructions, prosodic tunes, fixed phrases, productive morphology, etc. that enable them to generate and comprehend utterances ('texts') of arbitrary length and novelty. This knowledge has many aspects, such as an appreciation of the constraints on how individual resources can be combined, or of the relationships that must obtain among the elements of a text for it to be grammatical. However the overarching knowledge people have is of how these resources relate an utterance to the situation and intentional goals of the speaker, i.e. knowledge of the form–meaning relationship.

Deploying this knowledge either to generate or comprehend a text amounts to manipulating a representation of the linguistic resources in accordance either with the speaker's situation and goals or with a text, depending on the direction of the processing. The question for this volume is whether the identical representation can be used in both directions: Is there a version of the relationship between form (text) and meaning (speaker's situation and goals) that is *reversible*?

On its face, the assumption that a single, uniform representation of linguistic resources can be transparently deployed for either comprehension or generation is a less complex and more parsimonious hypothesis than one that says that different particulars or even different kinds of knowledge are used depending on the direction. The uniform representation is a stronger hypothesis since it constrains both processes and so provides more explanation. Should it prove to be wrong, we will still be able to learn more about the actual state of affairs than if we had started from the very beginning with the assumption that the two faculties use only minimally related resources.

The ultimate goals of the present work are psycholinguistic—to establish the actual representations and processing mechanisms for language that are used by people. However its immediate uses and its methodologies are computational, with the results employed in systems for natural language processing in conjunction with some *underlying application program* such as a knowledge based personal assistant. In such contexts, a reversible approach (sometimes also called a 'bi-directional' approach) has the advantage of efficient and uniform development. In particular, it ensures that the system will be able to understand any word or construction that it knows how to generate and deliberately generate anything that it knows how to understand—a capability that is too often missing in the systems in the early literature.

1.1. Efficiency and the flow of information

These strong methodological points in favor of a single reversible representation of linguistic resources notwithstanding, there are equally strong pressures for having different processing techniques in the two directions once we begin to consider the nature of comprehension and generation as information processing tasks. In comprehension, information proceeds from texts to situation and intentions. The process starts with the wording of the text and its intonation or orthographic markings. From these, the propositional content conveyed by the text and

the probable intentions of the speaker in producing it are deduced and represented. The primary effort is to scan the text's words in sequence, making observations from which the form of the text gradually unfolds. This requirement to scan forces the adoption of algorithms based on the management of multiple hypotheses and predictions that feed a representation that must be expanded dynamically. Major problems are caused by ambiguity and under-specification (i.e. the audience typically receives more information from situationally motivated inferences than is conveyed by the actual text).

In generation, information flows in the opposite direction. Generation proceeds from content to form, from intentions and perspectives to linearly arrayed words and syntactic constructions. A generator starts with its awareness of its intentions, its plans, and the text it has already produced. Coupled with a model of the audience, the situation, and the discourse, this provides the basis for making choices among the alternative wordings and constructions that the language provides—the principal activity in generation. Most generation systems do produce texts sequentially from left to right, just like a comprehension system would scan it; but they do this only after having made decisions about the content and form of the text as a whole. Ambiguity in a generator's knowledge is not possible (indeed one of the problems is to notice that an ambiguity has inadvertently been introduced into the text). And rather than under-specification, a generator's problem is to choose from its over-supply of information what to include and what to omit so as to adequately signal the intended inferences to the audience.

These radical differences in information flow pose a significant problem in the design of bi-directional natural language processing systems—a problem because *efficiency* is as strong a methodological goal as parsimony of representation, and no single uniform algorithm can handle such disparate activities as decision making and hypothesis maintenance with the same efficiency as two specifically tailored algorithms can. This is especially true if we include in our desiderata a requirement that normal processing be deterministic in the sense that all of the structures they build are *indelible*—all of them participate in the final analyses; none are retracted through "backup" (see, e.g., Marcus 1980; McDonald, Meteer & Pustejovsky 1987).

The impact of this problem as we see it is to force the decoupling of representation from algorithm. Sharing the same representation of linguistic knowledge will provide the needed uniformity, while the use of process-specific algorithms provides the efficiency. The question then becomes what kind of representation and what kind of integration between representation and processing will provide the greatest efficiency and most explanatory theory overall.

2. Reversibility through compilation

Most bi-directional NLP systems today do maintain this division between representation and algorithm (see, e.g., Shieber 1988, Wedekind 1988, Shieber et al. 1990). They treat the processes as transducers from expressions to expressions—from a logical form representing a sentence's propositional meaning to the string of words that constitute its terminals. The processes share a grammar that specifies the space of possible mappings between the two kinds of representation as rewrite rules or as constraints on possible structural descriptions. Unfortunately the algorithms these systems use, particularly when the grammar is

couched as constraints, are singularly inefficient when compared to most mono-directional algorithms since they rely on non-deterministic search.

The most efficient mono-directional algorithms involve procedural encodings of the linguistic knowledge in a set of direction-specific rules (e.g. Marcus 1980), and so would appear to be ruled out as candidates for a bi-directional approach. However this need not be the case. For one thing, it is possible to write reversible grammars in a procedural representation such as an augmented transition network provided that one is careful (see, e.g., Webber 1970). Alternatively, and more important for present considerations, one can provide a compiler that can transform one direction-specific representation into another and retain all of the knowledge encoded in the first representation while changing its form into one that can be efficiently processed in the complementary direction. Such a technique preserves the pragmatic benefits of a bi-directional system, namely uniform treatments in both directions, while the development effort is invested in one direction only. Another variation on this is to derive both runtime representations by compilation from a common, process-neutral representation.

For example, the massive systemic grammar for generation, NIGEL, (Mann & Matthiessen 1985) has been adapted for use as a parser (Kasper 1988). The original systemic grammar specifies a network of dependencies among paradigmatic alternatives that are represented as sets ("systems") of features. It is a direction-biased representation, with the generation algorithm following the network of systems from the least constrained alternatives to the most, deciding on one feature from each system as it is reached and posting the corresponding directive for how the feature is to be structurally realized in the text.

Bob Kasper's compiler converts the systemic network into a functional unification grammar representation ("FUG"), where it can then be deployed in a conventional parser for feature-based formalisms with phrase structure backbones such as an active-chart parsing algorithm. With the conceptual basis of the transformation in place, the development of further extensions and modifications is done on the generation grammar, and then that grammar is re-transformed to yield new parsing grammars.

Martin Kay's use of his FUG formalism (e.g. Kay 1980) provides another example of the compilation route to a bi-directional system. In this case both of the representations to be used are compiled from a common neutral representation, though the mapping to the representation used in the generation direction is essentially unchanged from the original as compared with the version used for parsing.

3. The 'pivot-point' between generation and comprehension

The work presented here is also based on compilation from a process-neutral representation, and, as with Kay and Kasper, the representation is grounded in the perspective of generation rather than parsing. What makes it most different from these and other approaches is our assumption that the proper *pivot point* between the two processes—where it is that "parsing" ends and "generation" begins—is quite deep, i.e. much closer to the speaker/hearer's general mental representations than is usually taken to be the case.

To understand this it will be useful to first look at the conventional alternative. In a system like Kay's,[2] the two processes pivot on a structural description ("SD"). Generation begins with a skeletal SD that specifies the intended values of grammatical functions such as subject or main verb. The SD can also encode discourse-level information such as given/new or contrastive focus. The leaves of the SD are words. Parsing ends with a comparable SD, though this one will incorporate a great deal more information (e.g. number concord, surface case, constituent order). This extra information corresponds to what is added to the initial SD in accordance with the demands of the grammar in the course of the generator's processing. Any reference to meaning in the SD (its "semantic" functions) is given as predicate–argument expressions.

These properties of the pivot point are typical of the bi-directional systems in the literature. The starting point for the generator (ending point of the parser) is an expression that is held to encode the text's logical form. This expression incorporates words directly (or uses trivially equivalent logical predicates). Any annotation of non-propositional information is given in terms of surface-oriented functions.

One must ask, however, how such expressions (SDs or logical forms) originate in the course of generation, or, for that matter, why it makes sense for a parser to stop at that point. The best kind of answer is that it is at that point that the character of the computations being performed changes. (Appelt 1989 makes this argument.) The choice of an SD as pivot point is not unreasonable in a mechanical translation task, especially one based on transfer methods. However it is a quite arbitrary choice when the task is communication between goal-oriented intentional agents embedded in situations.

From this point of view, a SD is a very late point in the generation process, corresponding to the start of surface realization. Before the information that a SD encodes can be determined, substantial and little understood subprocesses must operate: The speaker must construe the situation in realizable terms given the linguistic resources available in her language. (This is an especially important task when the speaker's source information is raw numerical data, for example precisely what points of the compass make the wind "*easterly*", see Bourbeau et al. 1990.) The information to be included in the utterance must be selected and a decision made as to whether it is to be mentioned explicitly or left for inference. The selected information must given an organization that reflects the intended rhetorical force and coherence, and it must be distributed into sentences with the appropriate cohesive devices given the context of the prior discourse. During all of this, a mapping of the information to its choice of linguistic resources must be found that make it collectively *expressible* [3] (i.e. has a surface realization, see Meteer 1992).

---

2 For an extensive description of a full-scale, well exercised generation system based on functional unification grammar see McKeown et al. 1990. This system starts considerably deeper than those being contrasted here however, as it incorporates a knowledge base with a model of its world (the maintenance of a complex radio) and it can plan graphical as well as textual output. The comparable portion is its surface realization component, which uses a version of FUG called "FUF".

3 A text plan that is not expressible will lead an incremental generator to talk itself into a corner at the point where the unrealizable element is reached, forcing a restart. One of the reasons why this

One of the ancillary goals of the bi-directional systems in the literature (e.g. Appelt 1989) is to isolate all linguistic knowledge inside the reversible component. This amounts to claiming that the early generation subprocesses just listed can function without knowing anything about the linguistic resources available for their use; but this is simply impossible given how we have characterized them—there must be knowledge of the linguistic resources available to the speaker during the planning stages of the generation process if the decisions we have posited to take place then are to be possible. Note, however, that this early linguistic knowledge need not be as precise or as detailed as what will be used later during surface realization. Lexemes can be selected without accessing their phonetic form. Surface syntactic constructions can be abstracted into generic relations such as head and arguments or matrix and adjunct, and can then be selected and composed without needing to simultaneously impose a linear order or realize their grammatical relationships (see Meteer 1992).

Given these considerations, it seems plausible to assume that linguistic knowledge—albeit in an abstract form—is present in generation as far back as the point where speakers begin to construe their situation in terms of combinations of realizable linguistic resources. Correspondingly, we can try developing a reversible representation of these resources at that level, pivoting now at the earliest level of linguistic categorization, the level where lexical items and basic syntactic functions (predication, attribution, modification) are deployed to realize the concepts and individuals that make up a speaker's mental model of her situation.

Under this construal, the pivot point of a reversible system consists of what we can treat computationally as a set of *objects*: representations of specific individuals, categories of individuals, and the relationships among them, both concrete and rhetorical.

Representationally, we assume that the objects in the speaker's model have a structure that reflects their type and that this structure defines the how they are categorized, their attributes, and their relationships to other objects. The representational philosophy we have adopted reflects the KL-ONE tradition (Woods & Schmolze 1992), where both categories and relations are organized into a specialization hierarchy, and individuals and non-taxonomic facts about them are organized into a separate database that can be traversed in the manner of a semantic net. This is formalized in a representational system known as KRISP (McDonald 1993).

Linguistically, we adopt the theory of Tree Adjoining Grammar ("TAG") as the basis of the surface representation of linguistic resources (Joshi 1985, Kroch & Joshi 1985). TAGs have long been seen as an especially well suited grammatical theory for generation because of their extended domain of locality for grammatical relations, their definition as a predefined set of trees at a strictly surface level of representation, and their implicit definition of what aspects of linguistic

---

could happen would be a choice of lexical paradigm that was missing the needed case, e.g. an Army that is on the *defensive* is said to be *defending* a position, but one cannot analogously refer to an Army on the *offensive* as **offending* the position. People talk themselves into a corner so rarely that we study the cases that occur as "speech errors", suggesting that expressibility may be as intrinsic a property of the generation process as grammaticality, and equally amenable to structural solutions in the generation architecture; see McDonald 1991.

structure are interdependent and must be selected as an entirety versus those aspects that can vary independently and so constitute real choices that a generator must make (Joshi 1987, McDonald & Pustejovsky 1985, Yang et al. 1991).

Lexicalized TAG (Schabes, Abeille & Joshi 1988) provides a set of devices that are a good fit to our construal of the pivot point. Individual words acting as phrasal heads, specifiers, modifiers, etc. are directly associated with the sets of syntactic trees that manifest the syntactic relations the words impose (e.g. a verb or relational noun's subcategorization pattern). Trees are grouped into tree families that reflect the textual variations that a given choice of word and subcategorization permit, collecting together for example the trees for active, passive, various relative clauses, questions, purpose clauses, reduced forms under conjunction, etc. into one structure. Ideosyncratic facts about specific words are readily captured: if, say, a given verb does not passivize, then there is no passive tree in its family. Moreover, TAG provides elegant mechanisms for composition of trees via substitution and its special adjunction operation. This is a crucial property, because the distributed pivot point representation proposed here makes extensive use of composition over minimal conceptual units in the course of comprehending/producing large texts.

A key question in the representational system is what constitutes a minimal, atomic unit of information. This is a theoretical question that will ultimately be decided on empirical grounds as psycholinguistic experiments are devised to probe this level of mental organization. For the present, we have adopted a hypothesis that ties the size of units in the model to the size (information content) of elementary trees in a TAG: *one tree–one unit* (McDonald 1991). Elementary trees are maximal projections of lexical categories (S, NP, Adjp, QP, etc.) that are restricted to not include recursive nodes. This implies among other things that while we will have units corresponding to named individuals, categories and individual instances of stuff (houses, numbers, the color red), and for partially saturated relations ('percent ownership of a company', 'being located in Japan'), we will not have minimal units corresponding to compositions of clauses, e.g. 'planning to leave' or 'establishing a joint venture to manufacture automobiles in Eastern Europe' even though such entities might be useful reifications in specialized reasoning systems.

4. Parsing to objects

It is relatively uncontroversial to assume that the generation process starts with the identification and selection of a set of objects. This is the founding assumption of the bulk of the generation systems in the literature that are designed to communicate information from a knowledge-based system, the closest facsimile to the situation of a human speaker that we have in the AI methodology (see, e.g., Dale 1990, Hovy 1991, Meteer 1992). Given such a starting point, the kind of linguistic knowledge that a generator will draw on most frequently is the options available for realizing these objects. To make this look-up efficient, one is naturally led to an architecture where this knowledge is stored directly with the definitions of the individuals or with the categories that define their properties and behavior, in effect distributing a highly lexicalized grammar over the knowledge base.

The idea that a set of objects and relations is the appropriate stopping point for the comprehension process (parsing) is less obvious (but see Martin & Riesbeck 1986 for just such an architecture). To some extent this is a matter of implementa-

tional perspective, since a formula in conjunctive normal form (such as the output of comprehension systems at SRI, see Hobbs et al. 1990) has already made many of the decompositions assumed here. In other respects, however, the difference is computationally quite significant, since the representation of individuals as logical constants or quantified variables has a quite different notational efficacy when compared with representing individuals as independent first class objects (see Woods 1986 for a discussion of this notion).

Given this disparity in the amount of research on object-centered research in the two directions, this paper will focus on the comprehension side of the problem and discuss generation only as an occasional point of reference. We will start by presenting our process-neutral conception of the form–meaning relationship—how individuals and the categories that define them can be linked to a declarative representation of the linguistic resources that can realize them. We will then show how this generation-biased representation can be reversed for use in the rule-by-rule system of semantic interpretation employed in the SPARSER natural language comprehension system (McDonald 1992). This will involve decomposing TAG tree families into sets of binary phrase structure rules, which will permit the use of SPARSER's highly efficient (constant time) phrase-structure driven chart parser as opposed to high polynomial time TAG parsers. We will walk through some examples of how SPARSER uses such rules, and end with a discussion of where the compiled parsing-biased representation of a lexicalized TAG stands with respect to other versions of TAG.

5. Linking linguistic resources to objects

Let us begin by looking at how a simple individual is linked to its realization. Consider the month December, the object denoted by the word *December* as it is used in "*My wife's birthday is in December*". Note that this is a different object than the actual instance of time denoted by the prepositional phrase in "*He will retire next December*". Rather it picks out a position in a cyclical mapping from named, delimited durations to actual instances of time stuff (Leban et al. 1986).

We want the representation of this object's linguistic realization to permit us to both produce the word *December* when it has been selected as part of an utterance to be generated, and also to engender a parsing rule that will recover this specific object when the word *December* is scanned in the course of a parse. This is what it means for the generator and parser to pivot on a set of objects rather than structural descriptions.

In our semantic model, December falls under the category 'month', and its realization is dictated by a parameterized specification that is given as part of that category's definition, shown below in Figure One. The representational system being used here is known as "KRISP", which stands for "knowledge representation in Sparser". When the Lisp form in the figure is evaluated it creates an object of type category. This object is set within a taxonomic hierarchy where it inherits from ('specializes') three categories as indicated by its 'specializes' field. The categories are 'cyclic-time-unit' (also used for days of the week, holidays, and birthdays; and itself a specialization of 'time-unit', which is used for weeks, hours, etc.), 'sequential' (which means that individuals that fall under this category have a

position in an ordered sequence; licensing sequence-positioning phrases like "*last December*" or "*the December after this one*"), and 'named-individual' (motivating the inclusion of a 'name' as the principal part of its realization).

```
(define-category  month
  :specializes ( cyclic-time-unit sequential named-individual )
  :instantiates  time-unit
  :index (:key name)
  :binds ((name (:v/r word))
          (containing-time-unit  year)
          (time-unit-contained  day)
          (length (:v/r number))
          (position-in-sequence (:v/r number)))
  :realization (:tree-family  NP-proper-name
                :mapping ((name . NP-head))
                :proper-name  name
                :saturation  :complete ))
```

Figure One — the category 'month'

Looking briefly at the next three fields in this definition, the 'instantiates' field indicates how instances of months are to be grouped in the discourse history; 'index' gives the storage properties of months (i.e. keyed lookup); and 'binds' lists the set of lambda variables that go with this category when it viewed as a predicate. While technically quite different, such variables can be seen as 'slots' in the KL-ONE tradition, or simply as fields in a structured object. They are given either a value restriction as indicated by the name of a category in parenthesis with ':v/r' (an actual value will be given with each individual month), or an actual value that is common to all months (distinguishing months from other kinds of cyclic time units).

The 'realization' field is the link that ties months to their linguistic resources. We will go into its mechanics below. Notice here that it points to a family of elementary trees (designated as 'NP-proper-name') and that it associates the word that will be the name of a particular month with the leaf position in those trees that serves as their lexical head (in the 'mapping' field). That word is also indicated to be a proper noun, meaning that it should be capitalized and will not be pluralized. Finally the realization of months via their names is marked as saturated, i.e. all that is needed to realize (or recognize) a particular month is the obligatory lexical head—as compared with the additional arguments needed to individuate events or other kinds of relational information.

To introduce December into the semantic model, the grammar writer uses the form below. Notice that it specifies the values that were left open in the category definition.

```
(define-individual  month
  :name "December"
  :length 31
  :position-in-sequence 12 )
```

Figure Two — defining an individual month

The representation of the resulting object is shown in Figure Three. The object itself is of type 'individual'. But in its capacity to represent objects in the domain being modeled, its 'type' field indicates that the individual it stands for is of type 'month','cyclic-time-unit', 'sequential' and 'named-individual'. Note that since we are at this point dealing with representations of objects rather than with meta-language expressions that define objects, they are given in '#<...>' notation.

```
#<individual
   :type ( #<month> #<cyclic-time-unit> #<sequential>
           #<named-individual> )
   :bindings ( #<binding name = #<word "December">>
               #<binding containing-time-unit = #<year>>
               #<binding time-unit-contained = #<day>>
               #<binding length = #<number 31>>
               #<binding position-in-sequence = #<number 12>> )>
```

Figure Three — the individual 'December'

This object incorporates five objects of type 'binding' that provide first class objects to represent December's attributes. They are the equivalent of filled slots. Each binding records two individuals and the relation between them. The relation is given directionally by the variable, which, as in the more recent KL-ONE -style representational systems, is an independent, first-class object organized taxonomically just as categories are. Bindings are used as the denotations (generation sources) of many copular clauses, e.g. "*there are 31 days in December*".

To realize this individual when it is selected as part of the information to be included in an utterance, we find the most specific of the categories that it falls under ('month') and consult the realization data given there as shown earlier, i.e. the tree family and the mapping of the individual's attribute values to leaf positions in the surface structure trees. That data is deliberately an exact fit to that required for generation in our framework, as it was designed specifically to the interface requirements of the Mumble-86 surface realization component and the Ravel text-structuring component. (See Meteer et al. 1987 and Meteer 1992 respectively for the specifications of these two components of a full natural language generation system.)

When parsing, we want an instance of the word *December* to be understood as denoting the individual in Figure Three since it is what represents the month in the model. That is, we do not want just to recover a description of the month (say a logical constant 'd' given as the argument of the predicate 'month'), rather we want to pick out this very object. This is the essence of what it means for generation and comprehension to pivot on a set of objects rather than an expression like an SD or a logical form: The parser recovers the same categories and individuals as the generator starts with (or the parser creates new objects as it recognizes new individuals in the text).

To facilitate this, in SPARSER the parsing rule that is responsible for *December* is written automatically as part of defining the corresponding individual—this is what what we mean by the notion of compiling the specification of the individual's linguistic resources (its primary category's realization field) into an efficient form for parsing.

Figure Four shows the phrase structure rule that rewrites the word *December* as a proper name with the label 'month'. Since SPARSER employs a rule by rule semantic analysis, this syntactic rule includes a specification of the semantic interpretation of the parse node ('chart edge') that results from the rule's completion, as indicated by the rule's 'referent' field. In this case the interpretation is trivially a pointer to the individual that represents the month December, which is available to the rule-writing procedure because the rule is written as part of the very process that instantiates the individual.

```
#<context-free-rule
   :print-form |month -> "December"|
   :lefthand-side  #<category month>
   :righthand-side  ( #<word "December"> )
   :syntactic-form  #<category proper-name>
   :referent  #<individual:month December> >
```

Figure Four — parsing rule for *December*

6. Summary of the approach

While this example of December is linguistically trivial, it illustrates the main points of this approach. Knowledge of language has two parts. One part is a body of linguistic resources, constituted as the lexicalized, surface level, syntactic, 'elementary trees' in a Tree Adjoining Grammar. The composition of these trees yields all the possible grammatical sentences of the language. The second part is the linking of the resources to the categories, relations, and individuals in the semantic model that comprises the speaker/hearer's conception of the world as construed by language.

The objects (entities) in this model are taken as the pivot point of a reversible NLP system: the source for generation and the target of parsing. For generation, the object–resource link is used essentially as is. For parsing, it is compiled into a set of single-level, immediate-constituent phrase structure rules with accompanying semantic interpretations. This separation of the statement of the resources—the TAG—from the form it takes when deployed provides for efficient processing in both directions, since the form is adapted to fit the pattern of information flow that each process requires.

This approach has nothing to say about how knowledge of linguistic resources or of the form–meaning relationship is acquired. The set of elementary trees is stipulated, not derived from some set of underlying principles that sets out rules for tree formation, a formal vocabulary of linguistic categories, and so on. This is deliberate, since while such elements are a central part of the linguistic enterprise and language learning is a key problem, we believe that these capacities are almost never employed when people comprehend or produce utterances.

In the rest of this paper we will go through a more realistically elaborate example of the compilation of parsing rules from a representation of a category's linguistic realization. This will entail some discussion of the issues in parsing TAGs, and will introduce the notion of an exploded tree family—the device used to define the compilation when relations rather than atomic individuals are involved.

7. Parsing Tree Adjoining Grammars

As they stand, Tree Adjoining Grammars are relatively difficult to parse. One has to manipulate interleaved, multi-level trees of constituents rather than the single level of immediate constituents of context-free grammars. As a result, the amount of state information one must keep in an Earley-type algorithm is dramatically increased (see Schabes et al. 1988, Schabes & Joshi 1988, Earley 1970) since each state now involves eleven terms rather than Earley's original four. For instance because of the possibilities for adjunction one now needs four position indexes into each candidate rewrite rule being considered where Earley had used only one.

Since an Earley style algorithm can run in linear time on some grammars, it is of course an improvement over the earlier algorithms for TAGs that used exhaustive search (CKY), which, because of the extra position indexes needed to accommodate the interleaving of trees, invariably required $O(n^6)$ time. However, the size of the state space that the Earley-type algorithm for TAG requires is always proportional to the number of trees introduced by the words in the string, and its construction and maintenance cannot be neglected in the algorithm's cost.

By contrast, SPARSER's parsing algorithm runs in constant time, is indelible (i.e. every edge that is introduced is part of the final analysis), and maintains no state information beyond its chart of completed edges—that is, its space and time bounds are independent of the size of the grammar (McDonald 1992). This means that reformulating the information in a TAG into the form of grammar rules that SPARSER uses will yield a strikingly improved overall efficiency. This is the motivation behind compiling the TAG representation of linguistic resources into a different representation for parsing, rather than attempting to directly reverse the representation used in generation.

SPARSER's efficiency is achieved through the use of an intricate and linguistically sensitive control structure, the adoption of a semantic grammar (Burton 1976, Burton & Brown 1979), and the predominance of binary rules (these facilitate the incremental composition of the semantic interpretations, and simplify the handling of optional and free-order phrases). In a semantic grammar, one folds together type-based semantic selectional restrictions and a conventional syntactic analysis of English constructions, multiplying the overall number of rules in the grammar accordingly. Rules are stated using constituent labels such as 'person', 'report-verb', or 'from-company', where these labels shadow their conventional counterparts, e.g. NP, verb, PP, and respect the customary conception of how syntactic constituents are partitioned and composed.

Because of this multiplication of syntactic and semantic labels, the number of rules in the system can be quite large.[4] However this has no impact beyond the off-line storage the rules need for their definitions. The parsing algorithm is bottom up, with the completion of binary rules checked for by adding a numerical representation of the labels of adjacent constituents and hashing the result into a table of legal completions—a constant-time operation. For other details, such as how the bottom up algorithm avoids the usual pitfalls of miss-registration and spurious edges, see McDonald 1992.

8. Exploded Tree Families

To discipline and make systematic the compilation of the normal tree-based representations of a lexicalized TAG into the immediate constituent representation used by SPARSER, we introduce schemas that re-encode each family of elementary trees as the requisite number of immediate-constituent rules. We call the result an *exploded tree family*. These schemas symbolically encode the syntactic constituent patterns and semantic interpretations that apply to any lexical head that has the indicated subcategorization and interpretation properties. They are instantiated to actual parsing rules as part of defining the semantic objects that are realized with those lexical heads. We will discuss the relationship between exploded tree families and the trees and tree families of other treatments of Tree Adjoining Grammar in section ten.

Figure Five shows Sparser's current version of the exploded tree family for transitive verbs that take the passive, such as the verb *confirm* as used by the object category definition in Figure Six.[5] It is indentified by its name: "transitive/passive". When generating a reference to a particular event with that category, that name picks out a tree family in Mumble-86's notation which is then used directly. When parsing a text, the rewrite rules this exploded tree family engenders will supply part of the grammar.

4 In an experiment in 1991 with a grammar for short news articles about executive-level personnel changes (the Wall Street Journal's "Who's News" column), about 2,100 context free and context sensitive phrase structure rules were used, achieving an 80% recall and 80% precision on a blind test of 200 new articles in the task of extracting four-tuples of person, event-type, title, and company.

5 At present, an exploded tree family is written by hand by the grammarian just as the families of elementary trees of a TAG are. When some means of defining elementary trees from first principles is developed it can be applied to both.

```
(define-exploded-tree-family  transitive/passive
  :binding-parameters ( agent patient )
  :labels ( s vp vg np-subject np-object )
  :cases
     ((:subject (s  (np-subject vp)
                 :head right-edge
                 :binds (agent left-edge))
      (:direct-object (vp  (vg np-object)
                       :head left-edge
                       :binds (patient right-edge)))
      (:passive (s  (np-object vg+passive)
                  :head right-edge
                  :binds (patient left-edge)))
      (:pos-nominalization (s  (np-object+pos vg+nominalization)
                            :head right-edge
                            :binds (patient left-edge)))
      (:of-nominalization (s  (vg+nominalization of+np-object)
                           :head left-edge
                           :binds (patient right-edge)))))
```

Figure Five — an Exploded Tree Family

Each of the expressions in the 'cases' field of the family corresponds to a phrase structure rule. The key with each rule, e.g. ':subject', is not a part of the rule; its role is to aid the grammarian in being systematic by identifying the grammatical relation of the argument that the rule is composing into the matrix (the head line). The rule proper follows the key. The first rule, for example, indicates that a constituent labeled 'S' —the rule's lefthand side— will be formed when there are two adjacent constituents in the text with labels 'NP-subject' and 'VP' —the rule's righthand side.

As one may already have anticipated from a glance at Figure Six, these are not the actual labels that will go on the actual rewrite rules of the grammar since they are syntactic terms and one of SPARSER's sources of efficiency is its use of a semantic grammar. Instead, the terms in these schemas are substitution variables as spelled out by the 'binding-parameters' and 'labels' fields in Figure Five. These symbolic terms are replaced with the concrete terms to be used in the runtime grammar according to the mappings given with the object definitions.[6] For example the subject rule, when instantiated for 'confirm-in-position' (Figure Six), will come out as two rules:

```
job-event -> board-of-directors job-event\agent

job-event -> company job-event\agent
```

This substitution technique allows the same schemas to be used to produce an arbitrary number of semantically labeled rules according to the lexicalizations and selectional restriction patterns that adhere to this syntactic paradigm.

6 We will not go into the details of the notational conventions that govern this substitution except to note that the '+' character and the keywords (terms with a prefixed colon) have special significance. Interested readers can write to the author for a copy of the SPARSER technical manual.

Continuing this exposition of how an exploded tree family is used as a rule schema, we can now turn to the fairly complex definition of a semantic category shown in Figure Six for the event-type that we call ‘confirm-in-position’. This is the category denoted by the verb *confirm* in a sentence like “*J. Gordon Strasser, acting president and chief executive officer of this gold mining company, was confirmed in the posts …*”, an actual example from the Wall Street Journal that SPARSER handles.

```
(define-category confirm-in-position
  :instantiates  job-event
  :specializes   get-position
  :binds ((agent (:v/r :or board-of-directors company))
          (person (:v/r person))
          (position (:v/r position)))
  :index (:temporary :list)
  :realization
    (:tree-family transitive/passive
     :mapping ((agent . agent)
               (patient . person)
               (s . job-event)
               (np-subject . (board-of-directors company))
               (vp . job-event\agent)
               (vg . :main-verb)
               (np/object . person)
     :main-verb "confirm"
     :saturation (agent person position)
     :additional-rules
        ((:adjunct (job-event  (job-event in-position)
                    :head left-edge
                    :binds (position right-edge)))
         (:adjunct (job-event  (job-event as-position)
                    :head left-edge
                    :binds (position right-edge))) )))
```

Figure Six — the event-type for ‘*confirm*’

This rather specific sense of *confirm* is analyzed here as a specialization of the event-type ‘get-position’ (along with *elect*, *appoint*, etc.), which in turn is a specialization of ‘job-event’, which is also the supercategory of ‘leave-position’ and is itself a specialization of ‘transition-event’. (The model of events used here is taken from Pustejovsky 1991.)

This category specifically binds three variables ‘agent’, ‘person’, and ‘position; as a specialization of more general kinds of events individuals with this category can of course also bind variables for time, location, purpose, etc. The agent of a confirm-in-position is the individual that causes the confirmation; in this domain it is typically a company’s board of directors. In most instances the agent is omitted as uninformative, leading to realization in the passive. The person is the one who gets the position, and that position is itself a composition of a title and a company.

The definition's 'mapping' field is quite long, since it not only identifies how the variables of the category will receive their values—the crucial semantic task for comprehension in this approach—but it also defines the substitutions for the exploded tree family. These substitutions produce the multiplication of basic syntactically-labeled rules by semantic labels that leads to the large number of rules in SPARSER's grammars. Looking just at the first instance of the subject rule (glossed earlier) that will be produced when this definition is executed, we get the following substitutions: the semantic category 'job-event' is subsituted for the symbol 's'; the category 'board-of-directors' is substituted for the symbol 'np-subject' (the annotation 'subject' distinguishing it from the other instance of an np in this schema so that the two can receive different mappings), and 'job-event/agent' is substituted for 'vp'. Spelled out in detail the full rule is shown in Figure Seven. (Note that the schema's symbol for this rule's lefthand-side, 's', has been projected to the corresponding syntactic category and installed as this rule's syntactic form. This category will be the default label on the nodes produced by this rule as discussed below.)

```
#<context-free-rule
    :print-form |job-event -> board-of-directors job-event/agent|
    :lefthand-side  #<category job-event>
    :righthand-side  ( #<category board-of-directors>
                       #<category job-event/agent> )
    :syntactic-form  #<category S>
    :referent
        (:head right-edge
         :binds (#<variable agent> left-edge)) >
```

Figure Seven — a parsing rule from the exploded tree family

Turning to this rule's semantic interpretation—given by its 'referent' field—we see that it is a set of instructions taken from the corresponding case in the exploded tree family (Figure Five, :subject case) with the appropriate substitution, i.e. the symbol 'agent' has been replaced by the variable specified by the 'mapping' field of the realization specification (Figure Six), which by coincidence has the same name. This semantic interpretation means that when the rule completes, the referent of the resulting edge (parse node) is to be the same object as the referent of the edge's head constituent, the verb phrase. (This constituent will be labeled 'job-event\agent', and it will be the right of the new edge's two daughter) Furthermore, this object (an individual with the domain-type 'confirm-in-position') is now to be augmented by binding its 'agent' variable to the individual that is the referent of the left daughter edge, the constituent labeled 'board-of-directors'.

Note that the linguistic resources linked to confirm-in-position (Figure Six) include two additional rules. These identify how its 'position' attribute is realized, namely with either of two different adjunct prepositional phrases, one using *in*, the other *as*. They are analyzed here as attaching to the sentence rather than the verb phrase for compatibility with the way nominalizations are treated. This kind of local augmentation of the general pool of linguistic resources (the set of tree families) is convenient for specifying the often ideosyncratic way that individual verbs can subcategorize for optional complements.

Figure Eight shows the full set of non-terminal parsing rules that are created by the definition of confirm-in-position. They are given here as just their print-forms accompanied by the key for the case that they correspond to in the exploded tree family transitive/passive or in the local cases of the category definition. An example showing how some of these rules are instantiated as edges by the parser and glosses of the objects they recover as referents follows in the next section.

```
(1)  job-event  ->  board-of-directors  job-event\agent
     [ subject ]

(2)  job-event  ->  company  job-event\agent
     [ subject ]

(3)  job-event  ->  job-event  in-position
     [ adjunct ]

(4)  job-event  ->  job-event  as-position
     [ adjunct ]

(5)  job-event  ->  person  confirm-in-position+passive
     [ passive ]

(6)  job-event  ->  person+pos  confirm-in-position+nominalization
     [ pos-nominalization ]

(7)  job-event  ->  confirm-in-position+nominalization  of-person
     [ of-nominalization ]

(8)  job-event\agent  ->  confirm-in-position  person
     [ direct-object ]
```

Figure Eight — the parsing rules for Confirm-in-position

In addition to these rules for non-terminals, a set of rules are created that are analogous to the rule for the word *December*, one for each of the morphological variations on *confirm*, i.e. *confirms*, *confirmed*, etc. Each rule gets as its lefthand-side the word-sense specific label 'confirm-in-position'; the word as its righthand-side; the appropriate syntactic category to indicate its syntactic form label (e.g. '3d-person-singular', 'nominalization', etc.); and will point to the same object as its referent—the object representing confirm-in-position as a category in the speaker/hearer's semantic model of her world (Figure Six).

These syntactic form labels of these rule feed into a common set of default rules for parsing the verbal auxiliary system. For example we can use a general rule for the passive based on the pattern *was* + 'ends-in-ed', rather than having to have one specifically for *was* + *confirmed* and all the other passivizing rules as a conventional semantic grammar would otherwise require.

This cuts down somewhat on the number of rules created overall, since we are not multiplying the 'confirm-in-position' label, which is specific to this semantic model, against all of the syntactic rules needed to handle modals, negation, participles, most adverbs, etc. The potential cost is the fact that the semantic interpretation of these phrases must be the same across the entire vocabulary. The results have been satisfactory thus far, but whether this will continue as SPARSER is applied to significantly more domains remains to be seen.

9. An example of the objects recovered by a parse

Consider the sentence "*J. Gordon Strasser, acting president and chief executive officer of this gold mining company, was confirmed in the posts*", which is the first clause of a five clause sentence from the Who's News column in the Wall Street Journal. Glossing over the parsing of the individual phrases involved because of space limitations, we can jump ahead to the point when the rules for confirm-in-position will apply and look at the edges that SPARSER has formed for those phrases in its chart. Figure Nine is a vertical presentation of that chart state, followed by a list of the edges with their semantic and syntactic labels and their referents. The edges names, e.g., 'e1', are for expository purposes only; the numbers around the excerpted texts are chart positions.

```
    e1   1 J. Gordon Strasser … of this gold mining company 18
    e2   18 was confirmed 20
    e3   20 in the posts 23

[e1: person, np, #<individual:person Strasser> ]

[e2: confirm-in-position+passive, vg, #<individual:confirm-in-
position/past> ]

[e3: in-position, pp, #<individual:position:collection President …> ]
```

Figure Nine — chart edges

At this point SPARSER will apply rule 5 (from the list in Figure Eight) to combine edges e1 and e2 to form a clause (S) labeled 'job-event', edge e4. Following the semantic interpretation that rule 5 has (see Figure Five), the referents of the two daughter edges are combined. The result—the referent of e4—is the same individual as the referent of edge e2, but now the individual is augmented by binding its 'agent' variable to the referent of e1, i.e. the object that represents Mr. Strasser.

The new edge, e4, will then be combined with the adjunct e3 by rule 3 with a similar result. Another edge, e5, is formed as shown in Figure Ten. Its referent is again the same individual we first saw in e2, and again as the referent of e4 (i.e. the individual representing this instance of confirm-in-position). For e5 it gets yet another binding object, this time assigning its 'position' variable to the position individual that is the referent of e3.

```
[e5: job-event, S,
     #<individual
      :type ( #<confirm-in-position> #<get-position> #<job-event>
              #<transition-event> )
      :bindings ( #<binding confirm-in-position-1.person =
                           #<individual:person Strasser>>
                  #<binding confirm-in-position-1.position =
                           #<individual:position:collection President …> )> ]
```

Figure Ten — the final edge, e5

The parse is now finished syntactically since we have recovered a grammatically complete unit, a sentence. Semantically it is still incomplete, since the text did

not explicitly give the agent that did the confirming and so the event-type is not yet saturated. In this genre it would be safe to infer that the agent is the company where the person now holds the position.

10. Is it Still a TAG ?

A defining element of the theory of Tree Adjoining Grammar is that it is based on the definition and manipulation of 'trees'—multi-level structures of sufficient scope to represent the entire syntactic domain over which the grammatical relationships imposed by the lexical head and its subcategorization frame apply. By contrast, the structures defined by the exploded tree family schemas are all single-level, defining one node in terms of its immediate constituents. A grammar comprised of these rules is context free; whereas a TAG has properties not expressible in a context-free formalism—particularly its 'links' and its operation of tree adjunction—that make it closer in power to a context-sensitive grammar (Joshi 1985).

This difference in generative power prompts the question of whether in this framework we can legitimately say that the parsing is being done with a TAG. We will begin to answer this by looking at similarities in the notation of exploded tree families and the distributed representation for trees found in some recent work in the theory of TAGs. That will then take us to our own motivations for that kind of notation, recapitulating why SPARSER gains efficiency by not being implemented directly as a TAG parser. Finally, we will take up the most substantial issue: identifying the counterparts to links and adjunction when an exploded tree family is used, which we will locate in the semantic, rather than the syntactic, realm of the comprehension system.

In Vijay-Shanker & Schabes (1992), the desire for a space-efficient, more easily maintained grammar led to a proposal to restructure the representation of a TAG as a hierarchy of partial descriptions of trees reflecting the portions of each tree that are common among a set of subcategorizations. Any tree headed in a verb, for example, will include the fragment "S -> NP VP, VP dominates V"; transitive verbs will add "VP -> v NP", placing them lower in the hierarchy; di-transitives involving PPs will be lower still and add a linear precedence constraint between the object NP and the PP; and so on. The constituent structure of the tree set is thus decomposed into a set of single level, immediate-dominance/linear-precedence statements—i.e. context-free rewrite rules—accompanied by annotations that specify germaine grammatical relationships and by statements of dominance (where a higher node dominates a lower one via zero to n intermediary nodes along a head line) that introduce ties to some lower node (such as the lexical anchor) where some other nodes may interveen, especially nodes introduced through adjunction.

Vijay-Shankar (1992) is another example of this same notational maneuver, i.e. reformulating the tree set of a TAG as a structure of context-free partial descriptions of trees whose minimal compositions yield the multi-level trees normally attributed to a TAG. His goal is to provide a formulation of TAG that is compatible with unification frameworks by making it possible to preserve the structural relationships between the nodes at a given level once they are established

in the course of a derivation or a parse. In a tree-based approach, the adjunction of, say, a subject-control raising verb within a clause (e.g. "IBM *plans to* lay off more people") will disrupt the subject–predicate relationship of the original tree ("IBM"–"lay off more people") by separating those NP and VP nodes, i.e. after the adjunction has taken place the subject NP is now in construction with a different VP (the one directly dominating *plans*) than it was beforehand. With context-free rules, by contrast, the NP–VP node-pair that is set down to capture the subject-hood of IBM is immutable.

Like these new treatments of TAG, exploded tree families (ETF) are based on representing the grammar in terms of relationships over a set of context-free rules. Crucially, they also share with them the fact that the rules are not defined in isolation but as part of the systematic definition of a set of trees. It is the rule-set as a whole that provides the extended domain of locality characteristic of TAG, which is manifested operationally in the fact that one cannot instantiate just one or two of the rules from an ETF schema, but must introduce them into the run-time semantic grammar as an entirty.

The most salient difference in exploded tree families is the desideratum for what rules are included in each schema, since the rules collectively define not single trees but families of trees, where a family is defined by what surface variations for a given subcategorization frame will be evidenced in texts given linguistically predictable 'transformations' in Zellig Harris's original sense and as elaborated by Abeille (1988). Like Vijay-Shankar & Shabes (1992), each schema factors out the common set of immediate-constituent patterns within the set of trees in the transformational family. Unlike their treatment, however, the ETF schemas do not go further and factor out all the shared constituent patterns in the entire grammar. The need to state a rule of semantic interpretation along with each rule of syntactic form mitigates against such a fully compacted treatment if only because it would make the grammarian's task too difficult because she could not see all of the semantic projections of the terms in a subcategorization frame at the same time.

The idea of factoring trees into their component immediate constituents is independently motivated by considerations of efficient parsing as discussed in the earlier sections of this paper. We can recapitulate that argument here, starting from the simple fact that a text makes certain kinds of information explicit and immediately available, while other kinds of information are available only by inference and the construction of intermediate representations. Efficient processing in parsing comes from attending first to what is explicit, and leaving until later, after the assembly of easily derived intermediate representations, the search for a text's implicit information. From this perspective, the adjacency of a pair of immediate constituents—the facts captured by the binary context-free rules defined schematically by an exploded tree family—is concrete, explicit information, immediately available to a bottom-up parsing algorithm. By contrast, the structure of a text as an assemblage of trees is implicit information, only available by grammar-driven inference once the adjacency facts have been recognized and appreciated.

The Earley-style TAG parsing algorithm developed by Schabes & Joshi (1988) involves the construction of hypotheses as to what trees may be projected from the lexical terminals of the text and of hypotheses about the patterns of substitution and adjunction of trees that could have led to the text. This makes the

parsing task into one of hypothesis maintainance and a controled kind of generate and test. With adjacency-driven binary rewrite rules, on the other hand, a bottom-up algorithm (along with a carefully designed control structure like SPARSER's to avoid spurious and misaligned edges) turns the parsing task into a monotonic, indelible process— one with no hypotheses and no maintanance of state information beyond the chart of already completed edges.

The experience thus far with SPARSER and the semantic labeled, context-free rules generated from exploded tree familes exibits just such highly efficent parsing properties, but it prompts the complementary question of whether the description of a text provided by context-free rules is the correct or most useful description of the relations in the text. Here the criteria for utility, given the goal of parsing to objects, is how effectively the correct semantic structures are identified. This takes us to the question of what are the counterparts of TAG links and adjunction this parsing framework, which we will take up now.

Links in TAG are a device to represent the relationship between gaps and their fillers (Joshi 1985, pg. 214). When we generate with a TAG using MUMBLE-86 (Meteer et al. 1987), such links are explicitly represented and are used to suppress explicit realizations whenever an individual is mapped to a constituent position that the grammar dictates can only be a gap. This forces the realization to be a trace (an orthographically null word) indexed to the entity so that grammatical relationships can be appropriately promulgated to lower constituents later in the sentence. One of the beauties of a TAG is the elegant way that these links are preserved when the filler and gap—always constituents of the same elementary tree—are later separated by the adjunction of new trees to the original, in effect 'stretching' the link over arbitrarily large spans of text depending on how many adjunctions are done.

Traces are explanatory constructs, however, not actual things one can find in a text. Parsing frameworks that try to directly recover traces in their surface structure analyses must hypothesize them at all the possible points where they could occur and then look for evidence to establish which hypothesis is correct. Not surprisingly then, surface structure analyses for SPARSER's grammar do not involve traces.[7]

An analog of stretchable links is still needed for semantic interpretation, however. This is because the correct domain of interpretation for a construction like a clause is (at least) the whole tree containing both subject NP and verb. Since the phrase structure recovered by SPARSER's grammar is not what a TAG would produce (i.e. a set of elementary trees and a derivation structure identifying how they were connected), its pattern of nodes cannot directly reflect the germane

7 Only the relative clause versions of WH-movement have been treated so far, as they are all that appear in our Who's News corpus. The WH of subject-relatives is handled by context-sensitive rules looking at the relative's NP head and rewriting the WH word as a copy of it, e.g., a person or a company. The result is the same patterns of semantic labels as simple declaratives, so no additional case is needed for them in the exploded tree families. The treatment of object relatives and pied-piping is in flux at the time this is written, vacilating between a type-rasing account that allows subject and verb group to combine as in categorial grammars, and an account that postpones the adjudication of the semantic interpretation of the WH phrase until semantic interpretation has occurred and what argument it is that is missing from the clause will be obvious.

semantic domains when, e.g., a subject and predicate have been separated by the interposition of adjoined phrases.

In some frameworks one might use something like an implementation of 'control' or the analog of unwinding the 'raising to subject' transformation to handle this kind of problem. Here we handle it as a semantic operation, since in the semantics we do recover the equivalent of the TAG description of the text.

Following the 'one tree – one object' principle, SPARSER recovers an object for each of the clausal trees in a sentence like "*IBM plans to lay off more people*", i.e. one for IBM laying off more people and for IBM planning to do that. It arranges for their common subject (agent) to flow through from the single surface position of the subject to the two objects that reference it by a semantic device akin to the NP movement chains in Government and Binding Theory. Briefly, it operates as follows:

Phrase construction, and with it semantic interpretation, occurs bottom up.[8] Thus when "*plans*" is composed with "*lay off more people*", the interpretation of that lower clause will have already been done and the fact that it is open in its agent (subject) role will be known. The rule doing that composition will appreciate that "*plan*" is a subject-control verb; accordingly it will mark the binding of the agent of the 'lay off' object as linked to that of the 'plan' object. A moment later, when "*IBM*" is composed with its adjacent VP "*plan to …*", the agent binding of the 'plan' object will be established in the normal manner (see §9), but now, because of the link, the agent binding of the 'lay off' will be established simultaneously. With the parse now finished, we will have one syntactic instance of IBM but two instances semantically, one instance in each of the two event objects corresponding to the two clausal trees that would have been recovered from the text under a standard TAG analysis.

11. Concluding remarks

We have illustrated how a single linguistic representation for linguistic resources, Tree Adjoining Grammar, can be linked to the objects of an application's semantic model and then efficiently deployed either for generation or comprehension. We have focused on comprehension, where the character of the information flow requires a very different runtime representation of the resources if parsing is to be done efficiently. This is accomplished by compiling a set of binary, immediate constituent phrase structure rewrite rules from the elementary trees of the TAG using a set of schemas derived from the TAG's tree families called an "exploded tree family".

As an approach to bi-directional NLP, this architecture is distinguished from others by its choice of 'pivot-point', the representational level at which the comprehension process ends and the generation process begins. We hold that the best choice for this level is the set of objects that make up the speaker/hearer's model of the world. This perspective influences our decision to talk about knowledge of language as "knowledge of a set of linguistic resources and their links to objects in the semantic model" rather than as "knowledge of a grammar", since it emphasizes the distributed nature of the knowledge and invites a strong lexical bias

8 This means that the interpretation is not incrementally left-to-right, which is a problem for further research that will focus on the implications of information structure for semantic interpretation.

as the way to account for the great variations that we see once we move out to study the 'periphery' of linguistic phenomena that dominate what we find in actual, unrestricted texts and go beyond the regularities of the 'core'.

The judgement that the pivot point should be so deep is motivated first by the observation from generation that the decisions made in what is often termed 'text planning' crucially require knowledge of what linguistic resources are actually available if text plans are to be expressible. Generation systems that are constructing utterances for application 'speakers' typically start from objects in the application's model, and they must compose these objects in order to form new utterances; knowledge of what compositions those objects' possible linguistic realizations will permit is thus central to the process.

On the comprehension side, a similar emphasis on the speaker/hearer's model is warranted because, in its natural setting, a text is understood against an enormous, long-term structure of individuals, categories, and relations that represent what the hearer already knows about the world. The text must be rendered into these same terms if its information is to be assimilated and its implications appreciated against this background of expectations. Once one adopts this perspective, the ergonomics (so to speak) of the parsing process shift, and it becomes easier to recover these objects directly, by closely linking the parser's knowledge of grammar to the model, rather than to get to the model by the indirect route of first recovering a logical form or similar kind of expression.

In terms of development, the SPARSER language comprehension system is mature at the time this paper is written, with a well worked out semantic analysis for the job-change domain that is being extended to the domains of commercial joint-ventures and changes in amounts such as earnings or stock prices. The use of the exploded tree families is still rather new, and as a result the more unusual grammar constructions and their semantic interpretations are still being written by hand since it is not yet clear how the generation counterparts of some of these constructions should be handled.

These difficulties are not with the statement of the constructions' form, since that is simple to do when generating with a TAG, but in establishing the reasons why one construction is used rather than another when they appear to carry the same propositional information. The careful reader will have already noticed that the example mappings are underspecified in the generation direction, since they are missing just this criteria that would allow a speaker to choose between, e.g., realizing position information with an *in* adjunct or with *as*.

This is a long-term problem at the heart of generation research. But it may be bi-directional systems that give us the best available methodologies for its study. If we adopt the hypothesis that the situation–utterance relationship is deterministic, i.e. that there are no truly stocastic elements influencing decisions[9], then by parsing a text and seeing what information we have to add to the the parser's output representation in order to regenerate that identical text and not one of its variants, we may be able to arrive at an account of the situational control of varia-

9 Of course if situations have small variations in them that are not consequential to a speaker's accomplishment of her goals, then in practice this may come to the same thing as if the generation procedure had flipped a coin in making some of its decisions.

tion through systematic experiment, which would represent a dramatic advance in the state of the art.

12. References

Abeille, Anne (1988) "A French Tree Adjoining Grammar", technical report, Dept. of Computer & Information Science, University of Pennsylvania.

Appelt, Doug (1989) "Bidirectional Grammars and the Design of Natural Language Generation Systems", in Wilks (ed.) *Theoretical Issues in Natural Language Processing*, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 199-205.

Bourbeau, L., D. Carcagno, E. Goldberg, R. Kittredge & A. Polguère (1990) "Bilingual Generation of Weather Forecasts in an Operations Environment", proc. 13th International Conference on Computational Linguistics (COLING-90), Helsinki, August 1990, pgs. 90-92.

Burton, R. (1976) "Semantic Grammar: An Engineering Technique for Constructing Natural Language Understanding Systems", Ph.D. Dissertation, University of California, Irvine.

_____ & J.S. Brown (1979) "Towards a Natural Language Capability for Computer-Assisted Instruction", in O'Neil (ed.) *Procedures for Instructional Systems Development*, Academic Press, New York, pgs. 273-313.

Dale, Robert (1990) "Generating Recipes: An Overview of Epicure", in Dale, Mellish, & Zock (eds.) *Current Research in Natural Language Generation*, Academic Press, New York.

Earley, Jay (1970) "An Efficient Context-Free Parsing Algorithm", Communications of the ACM 13(2), pgs. 94-102.

Hobbs, Jerry R., Mark Stickel, Douglas Appelt, & Paul Martin (1990) "Interpretation as Abduction", SRI Technical Note 499, SRI International, Menlo Park, California.

Hovy, Eduard (1991) "Approaches to the Planning of Coherent Text" in Paris, Swartout, & Mann (eds) *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, Kluwer, Dordrecht, the Netherlands,, pgs. 83-102.

Joshi, Aravind.K. (1985) How much context-sensitivity is required to provide reasonable structural descriptions: tree adjoining grammars. in Dowty et al. (eds) *Natural Language Processing*, Cambridge University Press.

_____ (1987) "The relevance of Tree Adjoining Grammar to generation", in Kempen (ed.) *Natural Language Generation*, Nijhoff, Dordrecht, the Netherlands, pgs. 233-252.

Kasper, Robert T. (1988) "An Experimental Parser for Systemic Grammars", proc. 12th International Conference on Computational Linguistics (COLING-88), Budapest, August 1988, pgs. 309-312.

Kay, Martin (1980 - 1986) "Algorithm schemata and data structures in syntactic processing", reprinted in Grosz, Sparck-Jones & Webber (eds.) *Readings in Natural Language Processing*, Morgan Kaufmann, Los Angeles, pgs.35-70.

Kroch, Anthony & Aravind Joshi (1985) "Linguistic relevance of tree adjoining grammars", Technical report MS-CI-85-18, Department of Computer and Information Science, University of Pennsylvania.

Leban, Bruce, David McDonald, & David Forster "A Representation for Collections of Temporal Intervals", 5th National Conference on Artificial Intelligence (AAAI), University of Pennsylvania, August 11-15, pgs. 367-371; Morgan-Kaufmann, Palo Alto.

Mann, William.C. & Christian. Matthiessen (1985) "A demonstration of the Nigel text generation computer program", in Benson & Greaves (eds) *Systemic Perspectives in Discourse*, Benjamins, Amsterdam.

Marcus, Mitch (1980) *A Theory of Syntactic Recognition for Natural Language*, MIT Press, Cambridge.

Martin, Charles & Chris Riesbeck (1986) “Uniform parsing and inference for learning”, 5th National Conference on Artificial Intelligence (AAAI), University of Pennsylvania, August 11-15,, pgs. 257-261; Morgan-Kaufmann, Palo Alto.

McDonald, David D. (1991) “One Tree – One Unit: a hypothesis for the conceptual sources underlying generation”, *Computational Intelligence* 7(4), pgs. 361-362.

_____ (1992) “An Efficient Chart-based Algorithm for Partial-Parsing of Unrestricted Texts”, proceedings of the 3d Conference on Applied Natural Language Processing (ACL), Trento, Italy, April 1992, pp. 193-200.

_____ (1993) “KRISP: a representation for the semantic interpretation of texts”, *Mind and Machines*, to appear.

_____ & James Pustejovsky, “TAG's as a Grammatical Formalism for Generation”, 23rd annual meeting of the Association of Computational Linguistics, Univ. of Chicago, July 8-12 1985, pp. 94-103.

_____, Marie Vaughan (Meteer), & James Pustejovsky (1987) “Factors Contributing to Efficiency in Natural Language Generation”, in Kempen (ed.) *Natural Language Generation: Recent Advances in Artificial Intelligence, Psychology, and Linguistics*, Kluwer Academic Publishers, Dordrecht, The Netherlands, in press, pp.159-181.

McKeown, Kathleen, Michael Elhadad, Yumiko Fukumoto, Jong Lim, Christine Lombardi, Jacques Robin, and Frank Smadja (1990) “Natural Language Generation in COMET”, in Dale, Mellish, & Zock (Eds.) *Current Research in Natural Language Generation*, Academic Press, London.

Marie Meteer (1992) *Expressibility and the Problem of Efficient Text Planning*, Pinter Publishers, London.

_____, David McDonald, Scott Anderson, David Forster, Linda Gay, Alison Huettner & Penelope Sibun (1987) *Mumble-86: Design and Implementation*, TR #87-87 Dept. Computer & Information Science, University of Massachusetts at Amherst.

Pustejovsky, James (1991) “The Syntax of Event Structure”, *Cognition*, 41, pgs. 47-81.

Schabes, Yves & Aravind Joshi (1988) “An Earley-type Parsing Algorithm for Tree Adjoining Grammars”, Linc Lab Technical Report 113, Department of Computer and Information Science, University of Pennsylvania.

_____, Anne Abeille, & Aravind Joshi (1988) “Parsing Strategies with ‘Lexicalized’ Grammars: Application to Tree Adjoining Grammars”, proc. 12th International Conference on Computational Linguistics (COLING-88), Budapest, August 1988, pgs. 578-583.

Sheiber, Stuart M. (1988) “A Uniform Architecture for Parsing and Generation”, proc. COLING-88, Budapest , July 1988, pgs. 614-619.

_____, Gertjan van Noord, Fernando Pereira & Robert Moore (1990) “Semantic-Head-Driven Generation, *Computational Linguistics* (16)1, March 1990, pp. 30-42.

Wedekind, Jürgen (1988) “Generation as Structure Driven Derivation”, proc. COLING-88, Budapest July 1988, pgs. 732-737.

Webber, Bonnie (1971) “The Case for Generation”, in Woods (ed.) Papers presented at the Seminar in Mathematical Linguistics, Vol. XIII, Spring 1971, Department of Linguistics, Aiken Computation Laboratory, Harvard University.

Woods, William (1986) “Important issues in knowledge representation”, proceedings of the IEEE 74(10).

_____ & J.G. Schmolze (1992) “The KL-ONE family”, *Computers & Mathematics with Applications*, 23(2-5), pgs. 133-177.

Yang, Gijoo, Kathleen F. McCoy, & K. Vijay-Shankar (1991) “From functional specification to syntactic structures: systemic grammar and tree adjoining grammar”, *Computational Intelligence* 7(4), pgs. 207-219.

Vijay-Shankar, K. (1992) “Using Descriptions of Trees in a Tree Adjoining Grammar”, *Computational Linguistics* 18(4), December 1992, pgs. 481-517.

_____ & Yves Schabes (1992) “Structure Sharing in Lexicalized Tree-Adjoining Grammars”, proc. 15th International Conference on Computational Linguistics (COLING-92), Nantes, August 1992, pgs. 205-211