AI Magazine Volume 19 Number 3 (1998) (© AAAI)

IJCAI-97 Research Excellence Award Acceptance Lecture

Relationship between Natural Language Processing and AI

Role of Constrained Formal-Computational Systems

Aravind K. Joshi

■ Modeling various aspects of language—syntax, semantics, pragmatics, and discourse, among others—by the use of constrained formal-computational systems, just adequate for such modeling, has proved to be an effective research strategy, leading to deep understanding of these aspects, with implications for both machine processing and human processing. This approach enables one to distinguish between the universal and stipulative constraints. This is in contrast to an approach where we start with the most powerful formal-computational system and then model the phenomena by making all constraints stipulative in a sense. The use of constrained systems for modeling leads to some novel ways of describing locality of structures and brings out the relationship between the complexity of description of primitives and local computations over them. These ideas serve to unify theoretical, computational, and statistical aspects of natural language processing in AI. It is expected that this approach will also be productive in other domains of AI.

It is indeed an honor to be selected by the International Joint Conferences on Artificial Intelligence for the Research Excellence Award. I am very grateful to all of you, my colleagues in AI, and my collaborators over many years, which include, most importantly, all my wonderful students. To all of you, I am immensely indebted.

I have always had a nagging doubt about whether I was truly an AI person or not. On two occasions, for IJCAI-75 and AAAI-97, I received reviews of my papers with comments such as, "This work is very interesting and wonderful but it is not quite AI!" I have always wondered about that, but now with this award, I guess, I have the official stamp of AI. Thank you again for selecting me for this award.

My talk will not be a survey of the field of natural language processing. It is not even a survey of all my own work. I will focus on only a few topics that illustrate an approach that has influenced my own work in a significant way. The particular set of ideas that characterize these efforts could best be described as the use of constrained formal-computational systems for describing various aspects of language—syntax, semantics, pragmatics, and discourse, among others. The use of constrained computational systems is in sharp contrast to starting with the most general and most powerful computational systems and then making all constraints, necessary for description, stipulative in a sense. The use of such systems allows one to distinguish between the universal and stipulative constraints. *Universal constraints* are properties of languages that are (or claimed to be) valid across languages and not language particular. All other properties that may be language particular are then stipulative. In this approach one tries to capture the universal properties as properties of the constrained system itself. On the other hand in the approach where the underlying system is unconstrained, all constraints introduced in the descriptions are stipulative by definition. Ideally we want a constrained system that captures all and only the universal properties. Of course, this is not achievable at present and perhaps may never be achievable. However, this is the goal of the constrained systems approach. This approach has proved to be quite successful in computational modeling of language and has given deep insights into the structure of languages. Moreover, it has also led to efficient processing techniques.

I have listed below five topics of my research that fall under the characterization of the approach using constrained formal-computational systems: (1) cascaded finite-state transducers for parsing; (2) lexicalized grammars—lexicalized tree-adjoining grammars (LTAGs); (3) some aspects of bilingual processing; (4) computation of certain classes of inference, for example, presupposition and entailments; and (5) local structure of discourse—centering.

I will only discuss three of these items, items 1, 2, and 5. These examples will serve to illustrate the main point of my talk—the role of constrained computational systems.

My first topic or example is the use of *cascaded finite-state transducers (*FSTs*)* for parsing. It also happens to be the very first work I did in natural language processing. As far as I know this is the first use of FSTs for parsing. This work (carried out during the period 1958–1959) was part of a project called Transformations and Discourse Analysis Project (TDAP), directed by Professor Zellig Harris at the University of Pennsylvania.[1]

The FST parser consists of a cascade of finite-state transducers corresponding to the following computations: (1) dictionary lookup and computation of the so-called *grammatical idioms*, that is, word clusters that behave as a single part of speech; (2) part-of-speech disambiguation; (3) computation of simple noun phrases, prepositional phrases, and verb clusters; and (4) computation of clauses (strictly not an FST computation). Rather than describing these computations, I will give an example, which is an actual output from the original program:

> [We] {have found} /that [subsequent addition] (of [the second inducer]) (of [either system]) < after {allowing} [single induction] {to proceed} + > (for [15 minutes]) (also) {results} (in [increased reproduction]) + \ + (of [both enzymes])

Here [...] denotes a simple noun phrase, (...) denotes a simple adjunct, and {...} denotes a verb cluster. Both < ... > and /... \ denote clauses. + denotes the end of a verb complement. After the dictionary lookup, grammatical idiom computation, and part-of-speech disambiguation, the simple noun phrases are computed by an FST scanning from right to left, then the prepositional phrases by a left-to-right FST, and the verb clusters by a left-to-right FST. The computation of clauses is done by a pushdown store with a depth-first strategy.

There are several reasons for mentioning this very early work. First FSTs are an example of a constrained computational system, but more importantly, FSTs are once again playing a very significant role in natural language processing, and many of the techniques used in this early work have close connections to some very recent work on FSTs. This resurgence of FST technology in natural language processing is because of our substantial knowledge of finite-state calculi; the new techniques for handling enormous sizes of FSTs and their determinateness and minimization; and, of course, techniques for handling stochastic and weighted FSTs. Some of the key efforts in this area are Koskenniemi, Tapanainen, and Voutilainen (1992), Karttunen (1996), Hobbs et al. (1992), and Mohri, Pereira, and Riley (1997).[2] FSTs are an example of a constrained system, but they also serve as an example of a computational system that is finite state (thus local) but where the state descriptions can be complex; so, it is an example (no doubt a simple one) of a computation that I call *local computation on complex structures*. I will return to this theme repeatedly.

Now I will turn to my second example—lexicalized grammars—which also illustrate constrained computational systems. A lexicalized grammar consists of a finite set of elementary structures (for example, strings, trees, or directed acyclic graphs), each structure associated with a lexical anchor. Each anchor may have more than one associated structure. Further there is a finite set of composition operations, which are universal, that is, language independent. Thus in a lexicalized grammar, in a sense, grammar and lexicon are the same thing; in other words, grammar = lexicon.

A particular example of a lexicalized grammar is the LTAG,[3] where each lexical item is associated with one or more elementary trees (or directed acyclic graphs). Each elementary tree encapsulates syntactic and semantic information associated with the lexical anchor.[4] In LTAG the elementary trees localize all dependencies including the so-called long-distance dependencies within the domain of the elementary trees. The two composition operations are substitution and adjoining. *Substitution* is the obvious operation—substituting a tree at a

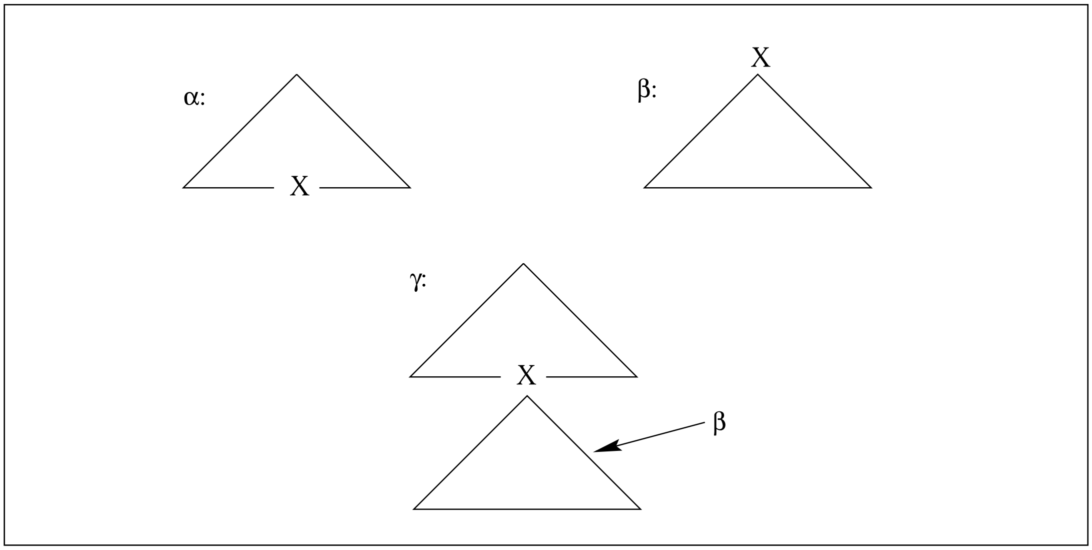


Figure 1. Lexicalized Tree-Adjoining Grammar Substitution.

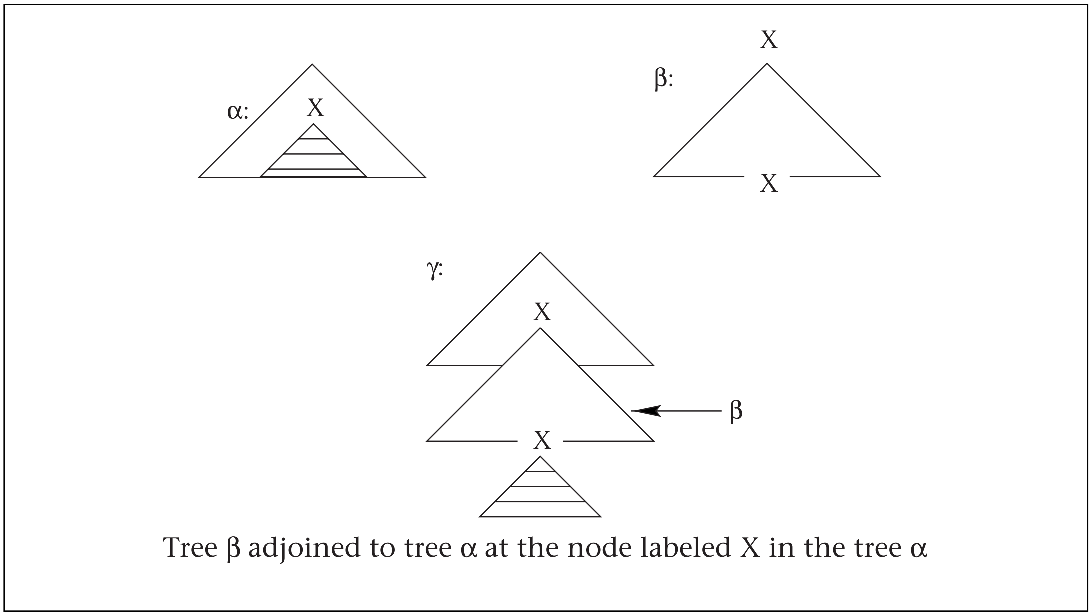

α:
X
β:
X
X
γ:
X
β
X
Tree β adjoined to tree α at the node labeled X in the tree α

Figure 2. Lexicalized Tree-Adjoining Grammar Adjunction.

frontier node of another tree. *Adjoining* is a more complex operation—splicing a tree into the interior of another tree. In figure 1, the tree β is substituted at the node *X* on the frontier of the tree α resulting in the tree γ. In figure 2, the tree β with root node labeled *X* and a frontier node also labeled *X* is spliced into, or adjoined into, the tree α at the node *X*, resulting in the tree γ. These two composition operations are language independent. Thus the entire linguistic or grammatical information is contained in the set of elementary trees. The linguistic theory then consists of a specification of this finite set of elementary structures.

For a long time I thought that tree adjoining was really a new idea, but David Weir (now at the University of Sussex, Brighton, United Kingdom) pointed out to me some time back that perhaps Vasily Kandinsky may have already known something about tree adjoining, as is clear from his watercolor, *Ins Dunkel,*[5] in figure 3. Fortunately, Kandinsky never wrote any papers on this topic, so I am quite safe.

The two operations—substitution and adjoining—both grow trees. Substitution grows them at the frontier, and adjoining grows them in the interior. It is the operation of adjoining that distinguishes LTAG from all

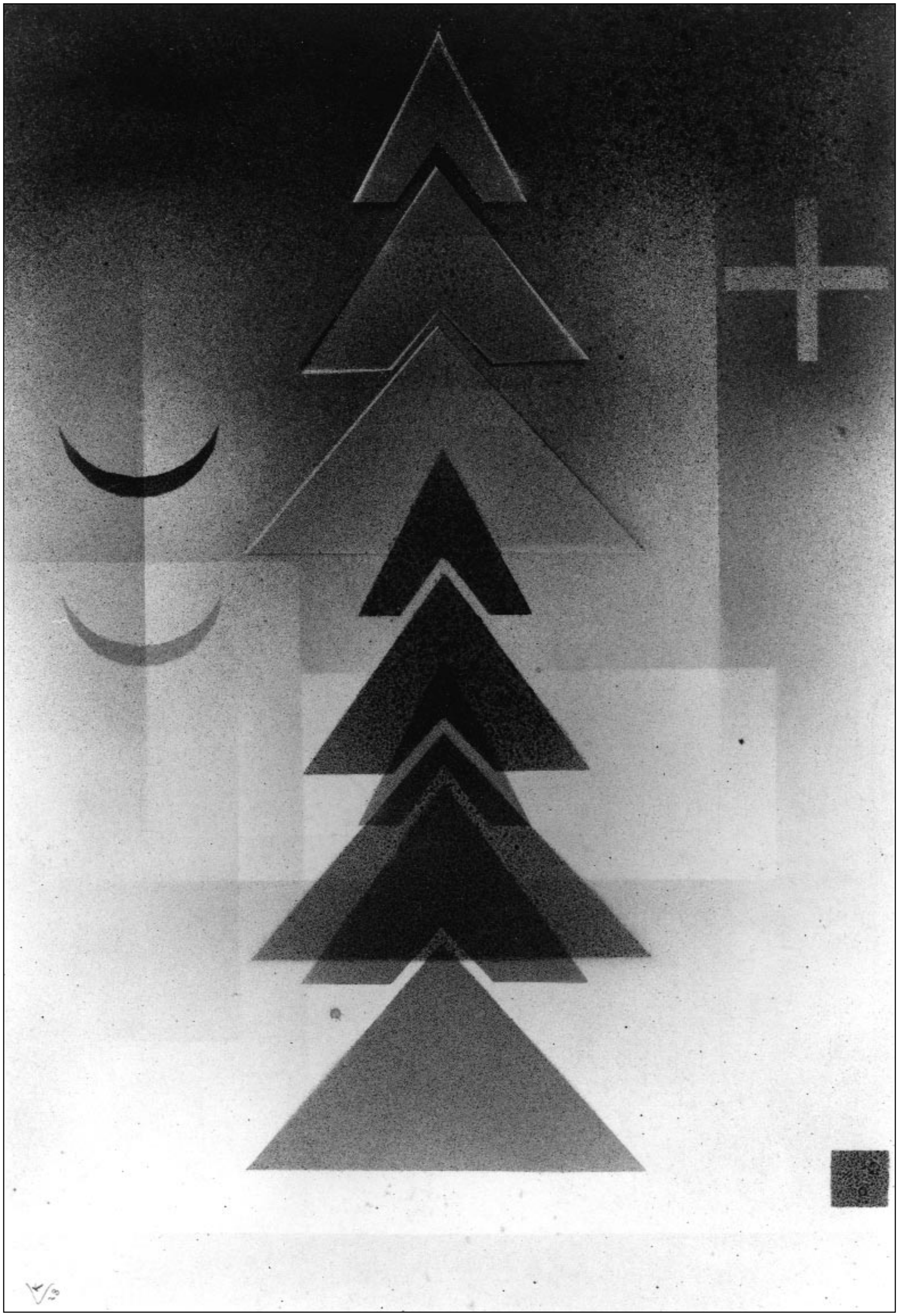


Figure 3. Vasily Kandinsky, Ins Dunkel*, 1928.*

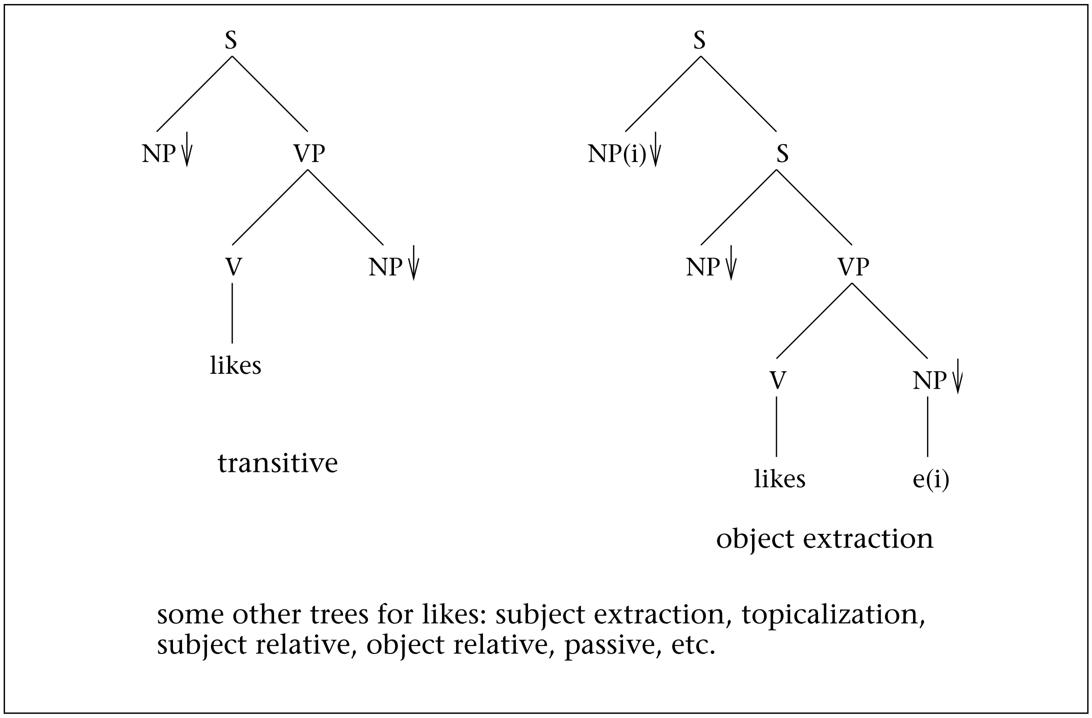

S
NP↓
VP
V
NP↓
likes
transitive
S
NP(i)↓
S
NP↓
VP
V
NP↓
likes
e(i)
object extraction
some other trees for likes: subject extraction, topicalization,
subject relative, object relative, passive, etc.

Figure 4. Lexicalized Tree-Adjoining Grammar Example 1.

other systems. Adjoining allows localization of dependencies including long-distance dependencies, and it allows modification of already built structures, an aspect to which I will return later. Some examples of syntactic dependencies are (1) *agreement,* person, number, gender, for example; (2) *subcategorization,* different verbs taking different complements, for example, *hit* requires noun-phrase (NP) complement and *think* requires NP sentence (S); and (3) filler-gap dependencies, for example, *who(i) did John ask Bill to invite (i),* where there is dependency between who and the complement of invite, which can be at an arbitrary distance. Some examples of semantic dependencies are (1) *function arguments,* the lexical anchor is treated as a functor, and then all its arguments are localized within an elementary tree; (2) *word clusters,* these are flexible idioms, for example, *take a walk;* the noncompositional aspects are localized within the elementary trees; and (3) *word cooccurrences and lexical semantic aspects,* these are, of course, directly related to the statistical dependencies such as the dependencies between a verb and the head nouns of its subject and complements.

Figure 4 shows some highly simplified representations of two lexically anchored structures associated with the word *like.*[6] The tree α1 corresponds to the transitive construction, and the tree α2 corresponds to the object-extraction construction. Of course, there will be many other trees associated with *like*, for example, for subject-extraction, topicalization, subject relative, object relative, and passive. In fact there will be an elementary tree for every minimal syntactic construction in which *likes* can appear. Figure 5 shows more examples of elementary trees. It is easy to see that in each tree the syntactic and the associated semantic dependencies have been localized.

In figures 6 and 7, a very simple example of a derivation is presented. The elementary trees that enter the derivation are shown in figure 6 and the derivation of *who does Bill think Harry likes* is shown in figure 7.

We start with the tree α2 corresponding to *likes* in the object-extraction construction. The trees for *who* and *Harry* are substituted in α2 as shown (by solid lines with arrows), resulting in a tree corresponding to *who Harry likes*. The trees for *Bill* and *think* are substituted in the tree β1 for *think* as shown (again by solid lines), resulting in a tree corresponding to *Bill think S*. Tree β2 for *does* is adjoined to this tree as shown by a dotted line with an arrow, resulting in a tree for *does Bill think S*, which is then adjoined into the tree for *who Harry likes* as shown by a dotted line, resulting in the tree corresponding to *who does Bill think Harry likes?* This derivation is shown in figure 8 where the solid lines represent substitution and the dotted lines represent adjoining. The

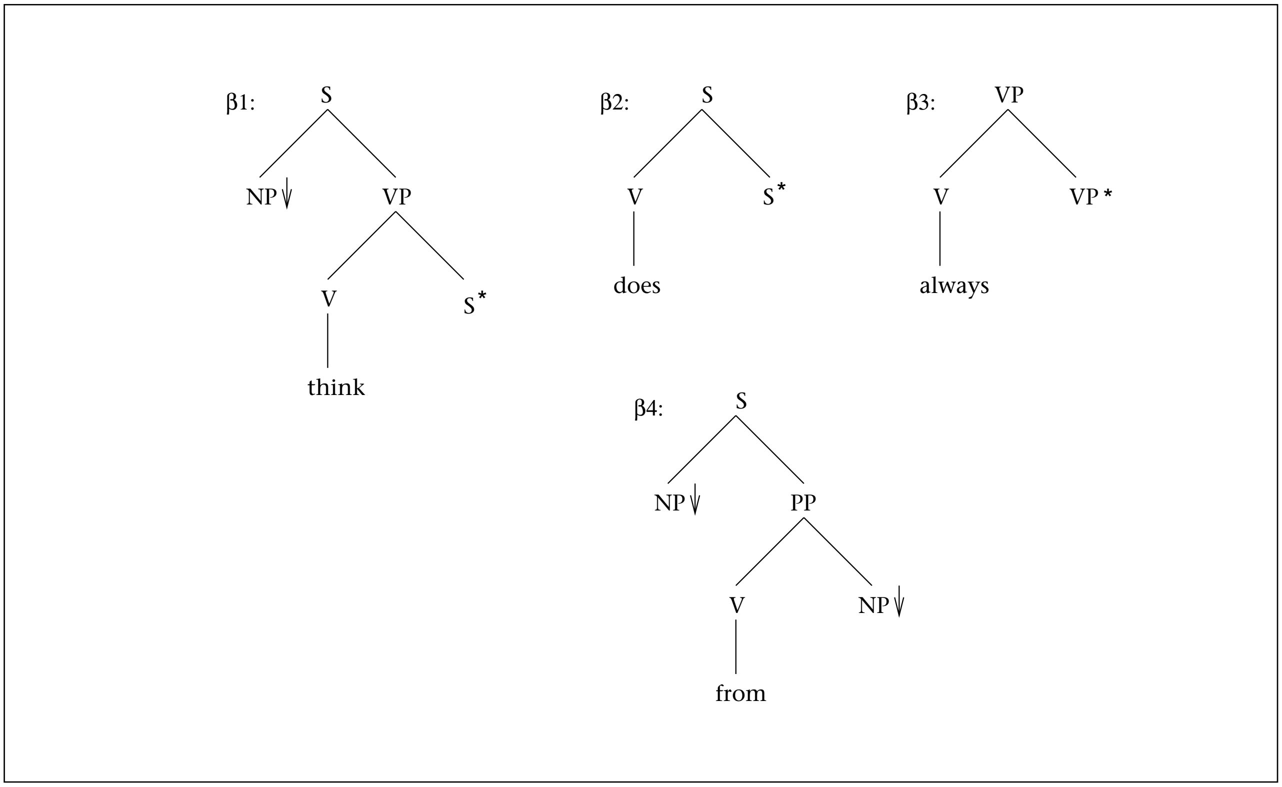


Figure 5. Lexicalized Tree-Adjoining Grammar Example 2.

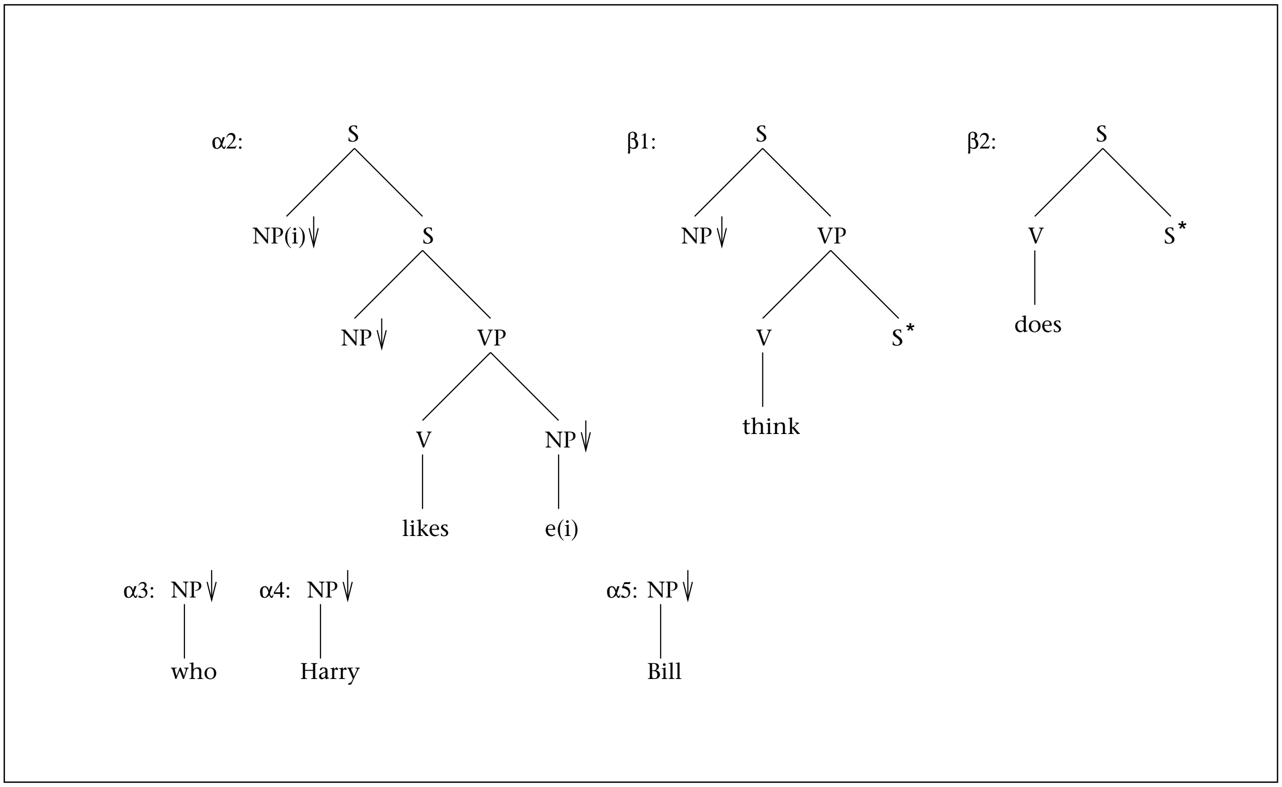


Figure 6. Lexicalized Tree-Adjoining Grammar Example 3.

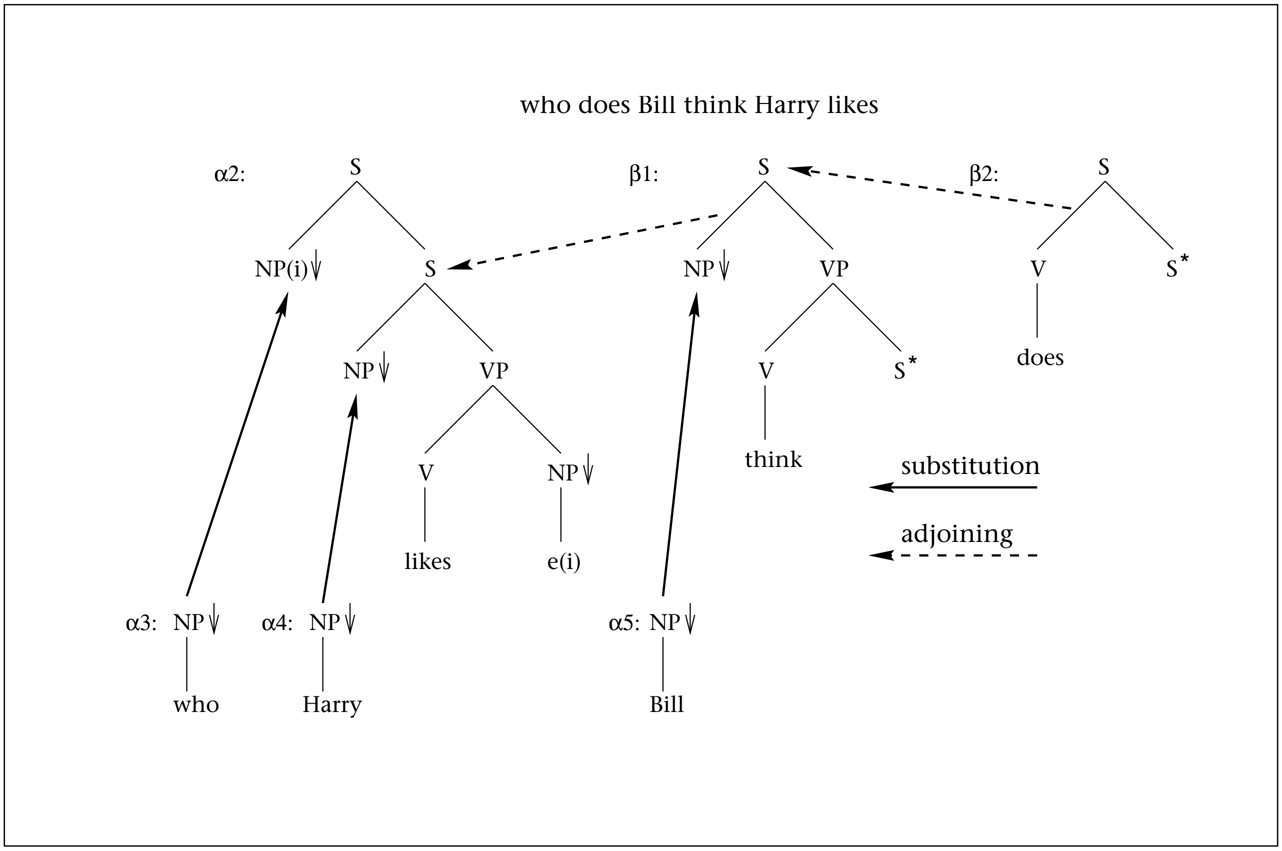


Figure 7. Lexicalized Tree-Adjoining Grammar Example 4.

representation in figure 8 is very close to the so-called dependency diagrams, which directly represent the dependencies among the lexical items.

A very large wide-coverage LTAG grammar for English has been built together with a parser—the XTAG system. It has been used for parsing a wide range of corpora (for example, the *Wall Street Journal* corpus) and also for some applications such as machine translation, information retrieval, and information extraction. I am not going to discuss these aspects because this is not the topic of my article. I am concerned here with constrained computational systems.[7] From this perspective I will focus more on the abstract character of adjoining. Adjoining, unlike substitution, changes (modifies) already built structures; that is, it is a kind of higher-order operation, a higher-order abstraction. This aspect of LTAG, together with the fact that it is a constrained computational system, has led to many applications of LTAG beyond parsing. Some examples are the generation work of Becker, Finkler, and Kliger (1997) in the VERBMOBIL Project and the work of Stone and Doran (1997) and Dras (1997).[8] The constrained nature of LTAG is used for compiling head-driven phrase structure grammar (HPSG) into LTAG, also in the VERBMOBIL Project (Kasper et al. 1995) and in the JPSS project at the University of Tokyo headed by Professor Tsujii (Tsujii et al. 1997),[9] where it has been used to compile LTAG into an HPSG, leading to a very efficient parser.

The abstract character of adjoining has also led to the use of LTAG for modeling incremental aspects of discourse structure as represented, for example, in the works of Gardent (1997) and Cristea and Webber (1997).[10] Further, in a nonlinguistic domain such as modeling complex dependencies in RNA secondary structures, the abstract character of LTAG has been exploited by Umeura et al. (1998) and Abe and Mamitsuka (1994).[11]

Supertagging

I will now take a completely different perspective on LTAG. I will treat the elementary trees associated with a lexical item as if they are a super part of speech (super POS or supertags) in contrast to the standard part of speech such as V (verb) and N (noun). Now it is well known that local statistical techniques can lead to

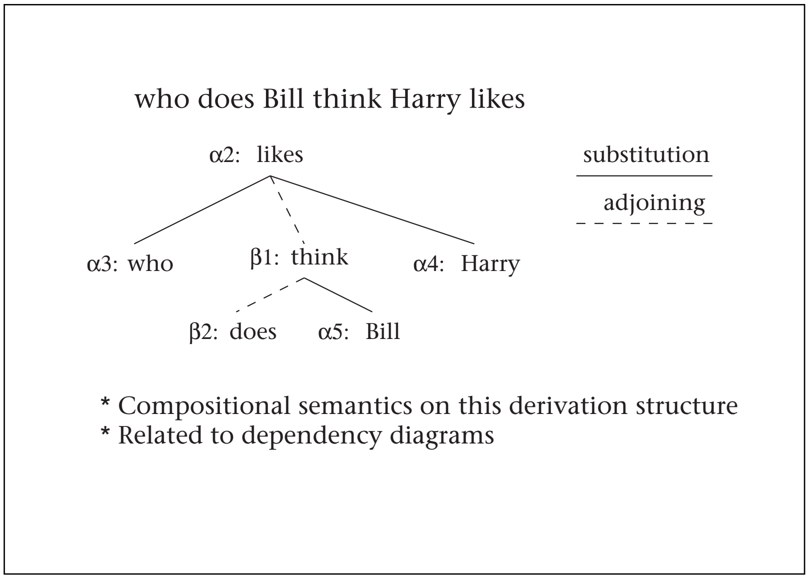

who does Bill think Harry likes
α2: likes
α3: who
β1: think
α4: Harry
β2: does
α5: Bill
substitution
adjoining
* Compositional semantics on this derivation structure
* Related to dependency diagrams

Figure 8. Lexicalized Tree-Adjoining Grammar Derivation Structure.

remarkably successful disambiguation of standard POS. Can we apply these techniques for disambiguating *supertags,* which are very rich descriptions of the lexical items? If we can, then, indeed, this will lead to "almost parsing." The approach is called *supertagging*.[12]

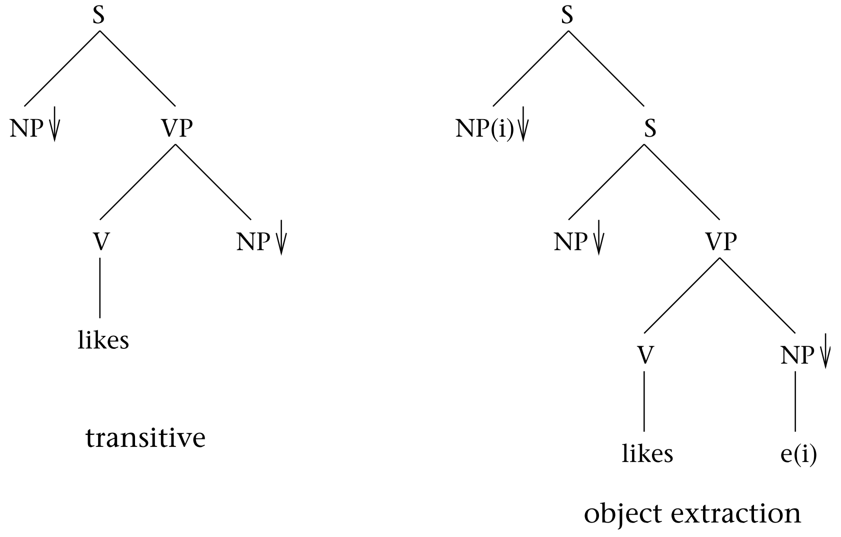
In figure 9 some elementary trees associated with the lexical item *likes* are shown. These are the same trees we have seen before. However, now we are going to regard these trees as supertags associated with *likes*. Given a corpus parsed by LTAG grammar, we can compute the statistics of supertags, statistics such as unigram, bigram, and trigram frequencies. Interestingly, these statistics combine not only lexical statistics but the statistics of constructions (as represented by the elementary trees) in which the items appear, thus combining lexical statistics with the statistics of the environments in which the lexical items appear.

Thus, for example, consider the string *the purchase price includes two ancillary companies* as shown in figure 10. The supertags associated with a word appear on top of each word. Some words have only one supertag associated with them and others have more than one. In the current XTAG system there are about 8 to 10 supertags per word on the average, so there is a very high level of local ambiguity. In figure 11 the same supertags are shown for each word; however, for each word one supertag has been identified (in a box). This is the correct supertag for this word in the sense that this is the supertag associated with this word in the correct parse of this sentence. Suppose we are able to find the correct supertag for each word in this sentence by applying local statistical disambiguation techniques; then for all practical purposes we have parsed the sentence. It is not a complete parse because we have not put the supertags together; hence, we call it *almost parse*.

A supertagging experiment was carried out using trigrams of supertags and techniques similar to the standard POS disambiguation techniques. The corpus used was the *Wall Street Journal* corpus. With a training corpus of 1 million words and a test corpus of 47,000 words, the baseline performance was 75 percent (that is, 75 percent of the words received the correct supertag). The baseline corresponds to the case where the supertag chosen for a word is just the most frequent supertag for this word. We know from the performance of disambiguators for the standard POS that the baseline performance is 90 percent or better. The low-baseline performance for supertagging is because the local ambiguity is very high (about 8 to 10 on the average) in contrast to the local ambiguity of the standard POS, which is about 1.5. The performance of the trigram supertagger, on the other hand, is 92 percent. The improvement from 75 percent to 92 percent is indeed very remarkable. This means that 92 percent of the words received the correct supertag.

In supertagging, we are working with complex (richer) descriptions of primitives (lexical items in our case). This is quite contrary to the standard mathematical wisdom or convention, where we keep the descriptions of the primitives simple and build complex descriptions out of simple descriptions. The descriptions of primitives (lexical items in our case) are complex because we try to associate with each primitive all information relevant to that primitive. Making descriptions more complex has two consequences: (1) local ambiguity is increased, that is, there are many more descriptions for each primitive, and (2) these richer descriptions of primitives locally constrain each other. There is an analogy here to a jigsaw puzzle—the richer the description of each piece the better, in the sense that there are stronger constraints on what other pieces can go with a given piece. Making the descriptions of primitives more complex allows us to compute statistics over these complex descriptions, but more importantly, these statistics are more meaningful because they capture the relevant dependencies directly (for example, word-to-word dependencies and word-to-construction dependencies). Local statistical computations over these complex descriptions lead to robust and efficient processing. Supertag-

Supertags of *likes*



some other trees for likes: subject extraction, topicalization, subject relative, object relative, passive, etc.

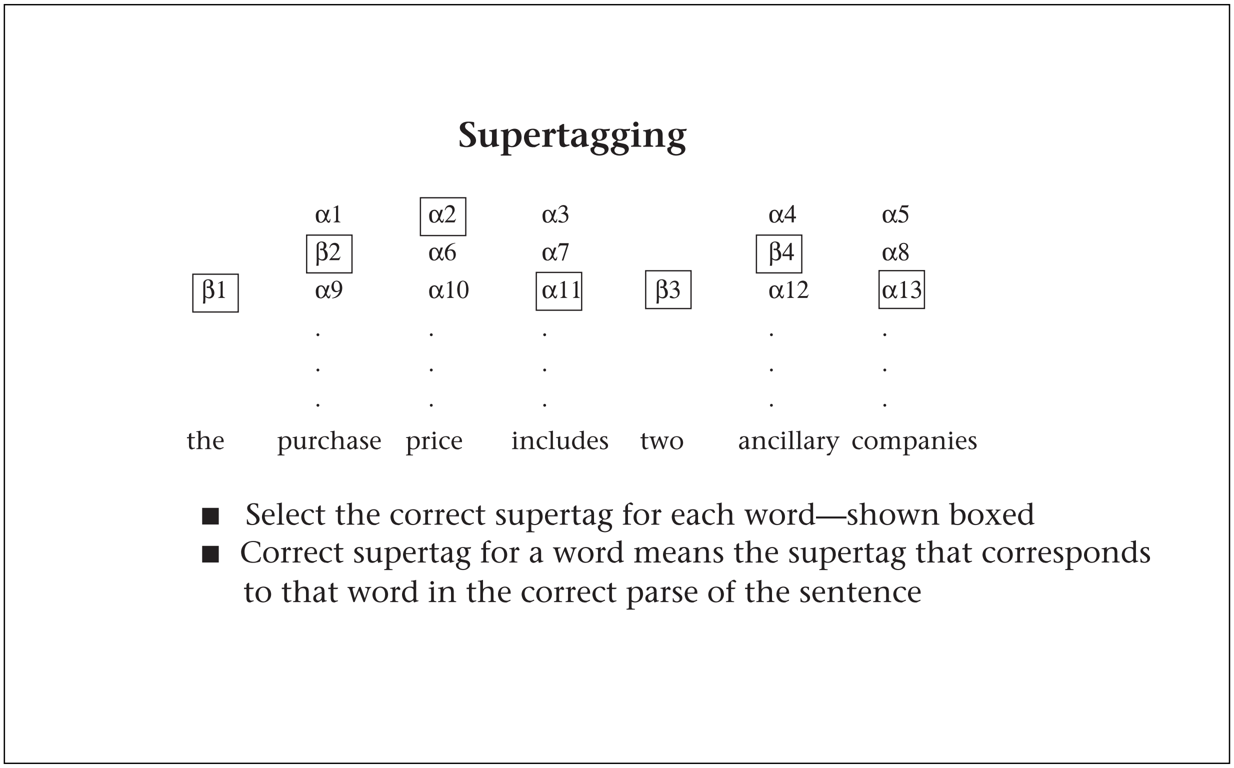
Figure 9. Supertags.

Supertagging

	α1	α2	α3		α4	α5
	β2	α6	α7		β4	α8
β1	α9	α10	α11	β3	α12	α13
	.	.	.		.	.
	.	.	.		.	.
	.	.	.		.	.
the	purchase	price	includes	two	ancillary	companies

On the average a lexical item has about 8 to 10 supertags

Figure 10. Supertagging.



Supertagging

	$\alpha 1$	$\boxed{\alpha 2}$	$\alpha 3$		$\alpha 4$	$\alpha 5$
	$\boxed{\beta 2}$	$\alpha 6$	$\alpha 7$		$\boxed{\beta 4}$	$\alpha 8$
$\boxed{\beta 1}$	$\alpha 9$	$\alpha 10$	$\boxed{\alpha 11}$	$\boxed{\beta 3}$	$\alpha 12$	$\boxed{\alpha 13}$
	.	.	.		.	.
	.	.	.		.	.
	.	.	.		.	.
the	purchase	price	includes	two	ancillary	companies

- Select the correct supertag for each word—shown boxed
- Correct supertag for a word means the supertag that corresponds to that word in the correct parse of the sentence

Figure 11. Supertagging—Almost Parse.

ging is thus an example of a local computation on complex descriptions.

These considerations are directly relevant to AI. I can illustrate this by pointing out interesting relationships to the well-known algorithm of Waltz (1975) for interpreting line drawings.[13] What Waltz did was to make the descriptions of vertices more complex by adding information about the number and types of edge incident on a vertex. Again there is an analogy here to a jigsaw puzzle: the richer the description of a piece the better. By making the descriptions of vertices more complex (richer), the local ambiguity was increased; for example, an L junction (a particular kind of junction in the taxonomy of junctions) has about 92 physically possible labelings. However, local computations on these complex descriptions are adequate to rapidly disambiguate these descriptions; so once again we have here an example of a local computation over complex descriptions, which has been my recurrent theme.[14]

My last example concerns the use of constrained computational systems in the area of discourse, in particular, complexity of inferences in discourse. To integrate an utterance in the previous discourse a variety of inferences need to be made. However, not all these inferences are equal in the sense that some are easier than others. This distinction is related, at least in part, to the structure of an utterance in a discourse. If a uniform machinery is used to subsume all inference mechanisms, then, of course, we will be able to model all these inferences but we will not learn much about the language-specific mechanisms that affect the complexity of inferences. One mechanism of a constrained inferential system is based on the local structure of an utterance in discourse, which is known as *centering.*[15] In my discussion of centering, I will limit myself to certain aspects of this work that relate to my particular position on constrained computational systems, the major theme of this article.

The basic idea is to start with the observation that an utterance in a discourse singles out an individual or entity among all those that are denoted by the arguments of the main predicate. This entity is called the *backward-looking center of the utterance,* which for the purpose of this article, I will call the *center.* The notion of a center is a discourse construct and not a syntactic or semantic construct. Centering an entity is equivalent to ascribing a property to an individual. The property itself may, of course, involve other individuals. As a simple example, consider the utterance

John hit Bill.

In a particular discourse, say D1, *John* may be the center, which we may represent as

(*JOHN x*)(*HIT x BILL*)

In another discourse, say D2, *Bill* may be the center, which we may represent as

(*BILL y*) (*HIT JOHN y)*

The main idea is that the notion of centering allows us to describe the rough logical form of an utterance in a discourse, where an *n*-ary predicate is made to look like a *monadic predicate* (predicate of a single argument) by singling out one argument as the center and temporarily hiding the other *n* – 1 arguments. This leads to a locally monadic structure of an utterance in a discourse, and it is this local monadic aspect of the logical form that has implications for the complexity of inference.

In the predicate calculus representation, all arguments of a predicate have an equal status. This is not true for an utterance in a discourse. It is interesting to note that as early as 1879, Frege was aware of this distinction and its relevance to the ease of certain inferences. Here is an interesting quote from Frege (1967):[16]

> In ordinary languages the subject in the sequence of words has a distinguished place.… This may, for example, have the purpose of pointing out a certain relation of the given judgment to others, and thereby making it easier for the listener to grasp the entire context.…"

Of course, Frege is assigning here the special status to the subject, which is a grammatical construct. In the centering theory the center is a discourse construct and not a syntactic construct, and any entity corresponding to an argument of a predicate can be centered. However, Frege clearly was aware of the (local) focus provided to an entity by the structure of an utterance in a discourse and its influence on the ease of certain inferences. The notion of (local) focus is central to all perceptual strategies for controlling inference. Therefore, it is not surprising that local structuring of an utterance in a discourse plays a role in controlling inferences.

The actual work on centering involves the study of the transition of centers from utterance to utterance because a centered entity does not necessarily remain centered all the time in a discourse (it would be a boring discourse, to be sure). Centers shift, new entities become centered, previously centered entities become centered again, and so on. The transitions of centers have to be described as patterns of continuations of the center and shifting of the centers (in other words, mechanisms for tracking the centers) as the discourse proceeds. Centering theory also involves the study of how pronouns and definite descriptions relate to centering. I will not discuss these details here, but the main observation, important for my topic here, is that the more the transitions and the more the entities centered in a discourse, the more complex the discourse becomes (that is, harder to process).

I will give a simple example to illustrate my main point. Consider the following two discourses, D1 and D2. In each case assume that *John* has been established as the center prior to D1 or D2.

D1: (a) John called Bill.
(b) He wanted his advice.

D2: (a) John called Bill.
(b) He was happy to hear from him.

The *he* in D1b refers to *John* and *his* to *Bill*. On the other hand, *he* in D2b refers to *Bill* and *him* to *John*. Under the centering characterization of an utterance in a discourse, the locally monadic representation of an utterance in a discourse leads to a prediction that the discourse D2 is harder to process than the discourse D1. This is because in centering theory continuation of a center is preferred to shifting (the latter leading to more complexity). In D2b the center has to be shifted from *John* to *Bill*. It is this shifting of the center in D2b that leads to the increased complexity.[17]

The main point here is that a discourse has a locally monadic structure. Monadic calculus is certainly a constrained system compared to the full predicate calculus. Thus by using a constrained system, we are able to capture certain aspects of inferential complexities in discourse. It should be noted that by adopting a constrained system, we have actually complicated the representation of the structure of an utterance in a discourse. Instead of having all the arguments of a predicate with an equal status, we have given one of the arguments a special status, thus complicating the description of primitives (utterances in a discourse, in this case). This situation is similar to the one in my second example (see Supertagging), in the sense that complexity of the description of the primitives is related to the ease of inference.

Conclusion

I gave three examples of my research relevant to the topic of this article, namely, the use of constrained formal-computational systems in modeling various aspects of language. I will now briefly discuss some unifying issues that arise out of these (and other related) examples.

The use of constrained formal-computational systems for modeling various aspects of language allows us to localize complex dependencies. This is one of the crucial results in the study of these systems. The use of such systems often requires us to make the descriptions of primitives more complex. However, this complexity leads to computations that become more local; in other words, the greater the complexity of the primitives the more local the computations over them. Further, statistics computed over primitives with complex descriptions are more meaningful in the sense that they capture the appropriate statistical dependencies (because of the localization of dependencies), and these in turn lead to efficient and robust computations.

Adopting primitives with complex descriptions requires us to have a richly annotated corpora of texts, dialogues, and various interactive situations, either obtained automatically or, most likely, semiautomatically. Statistics then have to be computed over these richly annotated corpora. Statistical techniques tell us how to count, but the computational models of various aspects of language tell us what to count. This is an obvious point but needs to be emphasized because computational modeling of various aspects of language is crucial in deciding the relevant primitive structures we want to deal with and count to collect useful statistical information. A significant result of this line of research is that local structures emerging out of the use of constrained formal-computational systems provide very appropriate units for counting in statistical processing.

To conclude, I have tried to show that the use of constrained formal-computational systems gives us deep insights into various aspects of language. It leads to appropriate notions of locality in syntax, semantics, and discourse. I gave three relevant examples of my research—one from syntax, one from syntax and semantics, and one from discourse. I discussed the relationship between locality and the complexity of descriptions of primitives—the more complex the description the more local the computations over them. I illustrated the relevance of locality for unifying theoretical, computational, and statistical aspects of natural language processing. These are the aspects that, I believe, make constrained formal-computational systems highly relevant to AI. I believe that this approach will be productive in other domains of AI also.

Acknowledgments

I have mentioned my major collaborators in the three topics I have discussed in my talk. I am not giving the entire bibliography that involves these and all other collaborators because of the lack of space. Instead I want to give a public acknowledgment of my debt to my collaborators over all these years by listing them below:

Zellig Harris, Lila Gleitman, Naomi Sager, Bruria Kauffman, Carol Chomsky, and Phil Hopely.

Hisao Yamada, S. Kosaraju, Leon Levy, Masako Takahashi, Richard Upton, Johnson Hart, Takeshi Yokomori, K. Vijayshanker, David Weir, Anthony Kroch, Yves Schabes, Anne Abeille, Tilman Becker, Robert Frank, Owen Rambow, B. Srinivas, Christy Doran, Beth Hockey, Seth Kulick, Anoop Sarkar, Martha Palmer, and Gann Bierner.

Ralph Weishedel, Stan Rosenschein, Jerry Kaplan, Eric Mays, Kathy McKeown, Kathy McCoy, Bonnie Webber, Tim Finin, Steve Kuhn, Scott Weinstein, Barbara Grosz, Ellen Prince, and Marilyn Walker.

This work is partially supported by National Science Foundation grant SBR8920230 and Army Research Office grant DAAH0404-94-GE-0426.

Notes

1. The other participants of this project were Lila Gleitman, Bruria Kauffman, Naomi Sager, and Carol Chomsky. By a remarkable coincidence, this program has recently been reconstructed faithfully from the original documentation, collaboratively with Phil Hopely. Two papers based on this work have also appeared recently: First is Joshi, A. K., and Hopely, P. 1997. A Parser from Antiquity. *Natural Language Engineering* 2(4): 291–294. Second is Kornai, A. 1998. *Extended Finite-State Automata*. Cambridge, U.K.: Cambridge University Press. Forthcoming, in which an extended version of reference 1 appears. This extended version includes an evaluation of this parser on some corpora, for example, the *Wall Street Journal,* IBM computer manuals, and ATIS (several modern parsers have been evaluated on these corpora also).

2. Mohri, M.; Pereira, F.; and Riley, M. 1997. Rational Power Series in Text and Speech Processing, Lecture Notes, AT&T Laboratories, Murray Hill, New Jersey.

Karttunen, L. 1996. Directed Replacement. In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics (ACL96)*, 108–115. San Francisco, Calif.: Morgan Kaufmann.

Hobbs, J. R.; Appelt, D. E.; Bear, J. S.; Israel, D.; and Tyson, W. M. 1992. FAUSTUS: A System for Extracting Information from Natural Language Text, Technical Note 257, SRI International, Menlo Park, California.

Koskenniemi, K.; Tapanainen, P.; and Voutilainen, A. 1992. Compiling and Using Finite-State Syntactic Rules. Paper presented at the Fifteenth International Conference in Computational Linguistics (COLING-92), August, Nantes, France.

3. Another important example is the categorical grammars, in particular, the combinatory categorical grammars of Steedman (Steedman, M. 1997. *Surface Structure and Interpretation.* Cambridge, Mass.: MIT Press).

4. My early collaborators in this work were Leon Levy and Masako Takahashi. My later collaborators were K. Vijayshanker, Anthony Kroch, David Weir, Yves Schabes, and Ann Abeille.

5. Vasily Kandinsky, *Ins Dunkel* (*Into the Dark*), 1928 © The Solomon R. Guggenheim Foundation, New York. © 1998 Artists Rights Society (ARS), New York/ADAGP, Paris.

6. In the actual grammar each node is decorated with feature structures with possible coindexing for the values of features across the nodes of the tree.

7. Mathematical properties of lexicalized tree-adjoining grammars (LTAGs) have been studied extensively. In particular, it is known that LTAG and several of its extensions belong to the class of mildly context-sensitive grammars, and they capture nested, crossed, and other more complex dependencies and they are polynomially parsable.

8. Becker, T.; Finkler, W.; and Kliger, A. 1997. Generation in Dialog Translation: Requirements, Techniques, and Their Realization in VERBMOBIL. Technical Report, 158, University of Saarlandes, Saarbrucken, Germany.

Dras, M. 1997. Representing Paraphrases Using Synchronous TAGs. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics (ACL97),* 88–95. San Francisco, Calif.: Morgan Kaufmann.

Stone, M., and Doran, C. 1997. Sentence Planning as Description Using Tree-Adjoining Grammar. Paper presented at the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics (ACL97), June, Madrid, Spain.

9. Kasper, R.; Kiefer, B.; Netter, K.; and Vijayshanker, K. 1995. Compilation of HPSG to TAG. In *Proceedings of the Thirty-Third Annual Meeting of the Association for Computational Linguistics (ACL95)*, 97–102. San Francisco, Calif.: Morgan Kaufmann.

10. Cristea, D., and Webber, B. 1997. Expectations in Incremental Discourse Processing. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics (ACL97),* 88–95. San Francisco, Calif.: Morgan Kaufmann.

Gardent, C. 1997. Discourse TAG. Technical Report, Computerlinguistik, University of Saarlandes, Saarbrucken, Germany.

11. Umeura, Y.; Hasegawa, A.; Kobayashi, S.; and Yokomori, T. 1998. Tree-Adjoining Grammars for RNA Structure Prediction. *Theoretical Computer Science.* Forthcoming.

Abe, N., and Mamitsuka, H. 1994. A New Method of Predicting Protein Secondary Structures Based on Stochastic Tree Grammars. In Proceedings of the Eleventh International Conference on Machine Learning, June, Tokyo, Japan.

12. This is joint work with B. Srinivas.

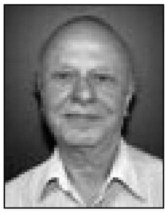
13. Waltz, D. 1975. Understanding Line Drawings of Scenes with Shadows. In *The Psychology of Computer Vision,* ed. P. H. Winston, 19–21. New York: McGraw-Hill.

14. Waltz (1975) did not use statistical information but this is not relevant to my main point.

15. My work on this system is a collaborative effort with Barbara Grosz and Scott Weinstein (and earlier with Steve Kuhn).

16. Frege, G. 1967. Begriffsschrift. In *From Frege to Godel,* ed. J. Van Heijenroot, 1–82. Cambridge, U.K.: Cambridge University Press.

17. A psycholinguistic experiment is suggested here. The prediction is that in interpreting *he* in D2b "attention" would first shift to *John* in D2a and then to *Bill* in D2a. An experiment based on the head-mounted eye-tracker technology is under consideration at present.



Aravind K. Joshi is the Henry Salvatori professor of computer and cognitive science in the Department of Computer and Information Science and the codirector of the Institute for Research in Cognitive Science, a National Science Foundation Science and Technology Center (STC) for Research in Cognitive Science, at the University of Pennsylvania. He is a founding fellow of the American Association for Artificial Intelligence and a fellow of the Association of Computing Machinery and the Institute of Electrical and Electronics Engineers. He was the program chair for the Ninth International Joint Conference on Artificial Intelligence. His e-mail address is joshi@linc.cis.upenn.edu.

Insert IDA 99 and 5th Pacific Rim Conference Ads.
Reduce as necessary to fit