ELSEVIER

Artificial Intelligence 103 (1998) 117–132

Artificial Intelligence

Role of constrained computational systems in natural language processing [☆]

Aravind K. Joshi [1]

Department of Computer and Information Science and Institute for Research in Cognitive Science, Room 555, Moore School, University of Pennsylvania, Philadelphia, PA 19104, USA

Abstract

The use of constrained formal/computational systems just adequate for modeling various aspects of language—syntax, semantics, pragmatics and discourse, among others, has proved to be an effective research strategy leading to deep understanding of these aspects, with implications to 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 languages processing in AI. It is expected that this approach will be productive in other domains of AI. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Abstract character of adjoining; Adjoining; Almost parse; Centering; Center of an utterance; Complexity of inference; Control of inference; Constrained formal systems; Finite state transducers; Lexicalized grammars; Lexicalized tree-adjoining grammars; Local computations on complex structures; Local statistical computations; Locality of structures; Locally monadic structure; Monadic predicate; Substitution; Supertags; Supertagging; Universal and stipulative constraints

This paper is a somewhat expanded version of my IJCAI-97 Research Excellence Award lecture. It is not a survey of the field of natural language processing. It is not even a survey of all of my own work. I will focus on only a few topics which illustrate an approach that has influenced my own work in a significant way. The

[☆] This work is partially supported by NSF Grant SBR8920230 and ARO Grant DAAH0404-94-GE-0426.
[1] Email: joshi@linc.cis.upenn.edu.

particular set of ideas that characterize theses efforts could best be described as the use of constrained formal/computational systems for describing various aspects of language—syntax, semantics, pragmatics, 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) universal across languages and not language particular. All other properties which may be languages 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.

Here are 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 (LTAG).
(3) Some aspects of bilingual processing.
(4) Computation of certain classes of inferences, for example, presupposition and entailments.
(5) Local structure of discourse—centering.

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

1. Cascaded finite state transducers

My first topic or example is the use of finite state transducers (fst) for parsing. It also happens to be the very first work I did in natural languages processing. As far as I know this is the first use of fst's 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.[2]

[2] The other participants of this project were Lila Gleitman, Bruria Kauffman, Naomi Sager, and Carol Chomsky. By a remarkable coincidence this program has been recently faithfully reconstructed from the original documentation, collaboratively with Phil Hopely. Two papers based on this work have also appeared recently—A.K. Joshi, P. Hopely, A parser from antiquity, Natural Language Engineering 2 (4) (1997). An extended version of this paper which includes an evaluation of this parser on some corpora, for example, Wall Street Journal (WSJ), IBM Computer Manuals, and ATIS (several modern parsers have been evaluated on these corpora also) will appear in: A. Kornai (Ed.), Extended Finite State Automata, Cambridge University Press, 1998.

The fst parser consists of a cascade of finite state transducers corresponding to the following computations:

- dictionary look-up and computation of the so-called *grammatical idioms*, i.e., word clusters which behave as a single part-of-speech;
- part-of-speech disambiguation;
- computation of simple noun phrases, prepositional phrases, and verb clusters;
- 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 look-up, grammatical idioms 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 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 fst's are an example of a constrained computational system but, more importantly, fst's are once again playing a very significant role in natural languages processing and many of the techniques used in this early work have close connections to some very recent work on fst's. This resurgence of fst technology in natural language processing is due to our substantial knowledge of finite state calculi, the new techniques for handling enormous sizes of fst's and for their determinization and minimization, and, of course, techniques for handling stochastic and weighted fst's. Some of the key efforts in this area are Koskenniemi et al. (1992), Karttunen (1996), Hobbs et al. (1992) and Mohri et al. (1997).[3]

Finite state transducers are an example of a constrained system but they also serve as an example of a computational system which 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 which I call *local computation on complex structures*. I will return to this theme repeatedly.

2. Lexicalized tree-adjoining grammars

Now I will turn to my second example—lexicalized grammars, which are also examples of constrained computational systems. A lexicalized grammar consists of a finite set

[3] K. Koskenniemi, P. Tapanainen, A. Voutilainen, Compiling and using finite-state syntactic rules, in: Proc. 15th International Conference in Computational Linguistics, COLING-92, Vol. I, 1992, Nantes, France, pp. 156–162. L. Karttunen, Directed replacement, in: Proc. 34th Annual Meeting of the Association for Computational Linguistics, ACL-96, Santa Cruz, 1996. J.R. Hobbs, D.E. Appelt, J.S. Bear, D. Israel, W.M. Tyson, FAUSTUS: a system for extracting information from a natural language text, Technical Note, SRI International, Menlo Park, 1992. M. Mohri, F. Pereira, M. Riley, Rational power series in text and speech processing, Lecture Notes, AT&T Laboratories, Murray Hill, 1997.

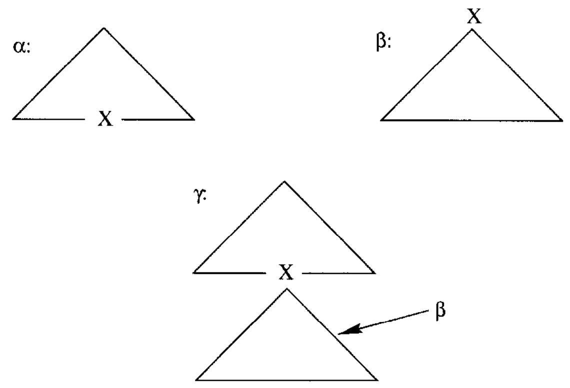


Fig. 1. LTAG substitution.

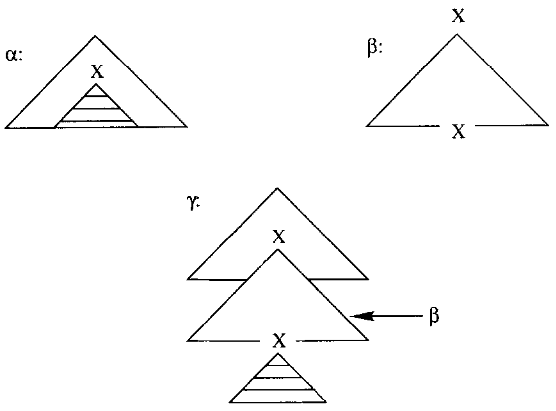
of elementary structures (for example, strings, trees, or directed acyclic graphs), each structure associated with a lexical anchor. Each anchor may have more than one associated structure. Further there is a finite set of composition operations, which are universal, i.e., languages independent. Thus 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[4] is the Lexicalized Tree-Adjoining Grammar (LTAG), where each 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.[5] 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 frontier node of another tree. Adjoining is a more complex operation—splicing a tree into the interior of another tree. In Fig. 1 the tree β is is substituted at the node X on the frontier of the tree α resulting in the tree γ. In Fig. 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.

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 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:

[4] Another important example is the Categorial Grammars, in particular: M. Steedman, Combinatory Categorial Grammars of Steedman, Surface Structure and Interpretation, Linguistic Inquiry Monograph, MIT Press, Cambridge, 1997.

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



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

Fig. 2. LTAG adjoining.

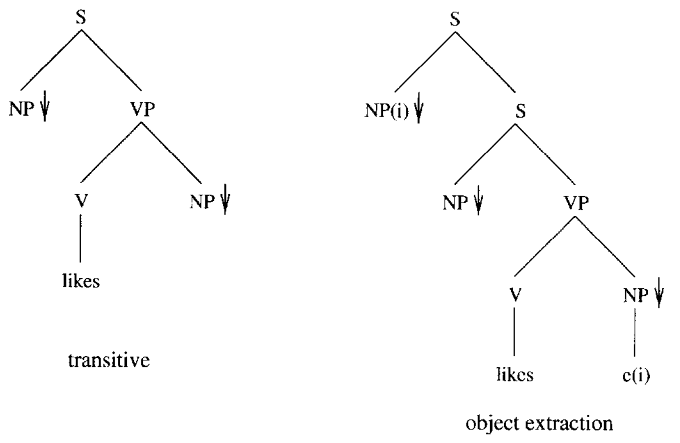
different verbs take different complements, *hit* requires NP (noun phrase) complement, *give* requires NP NP, *think* requires NP S (sentence), for example, (3) filler-gap dependencies: *who*(*i*) *did John ask Bill to invite* (*i*) where there is dependency between *who* and the complement of *invite*, which can be at an arbitrary distance, *who* is the filler of the gap that appears after *invite*, (4) word-order variation within and across clauses. Some examples of semantic dependencies are: (1) function-argument: the lexical anchor is treated as a functor and then all its are arguments are localized within an elementary tree, (5) word-clusters: flexible idioms, for example, *take a walk, give a cold shoulder to*; the noncompositional aspects are localized within the elementary trees, and (6) word co-occurrences 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.

Fig. 3 shows some highly simplified representations[6] of two lexically anchored structures associated with the word *like*. The tree $\alpha 1$ corresponds to the transitive construction and the tree $\alpha 2$ 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, passive, etc. In fact there will be an elementary tree for every "minimal" syntactic construction in which *likes* can appear. Fig. 4 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 Figs. 5 and 6 a very simple example of a derivation is presented. The elementary trees that enter the derivation are shown in Fig. 5 and the derivation of *who does Bill think Harry likes* is shown in Fig. 6.

We start with the tree $\alpha 2$ corresponding to *likes* in the object-extraction construction. The trees for *who* and *Harry* are substituted in $\alpha 2$ as shown (by solid lines with arrows), resulting in a tree corresponding to *who Harry likes*. The trees for *Bill* and *think* are

[6] In the actual grammar each node is decorated with feature structures with possible co-indexing for the values of features across the nodes of the tree.



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

Fig. 3. LTAG examples 1.

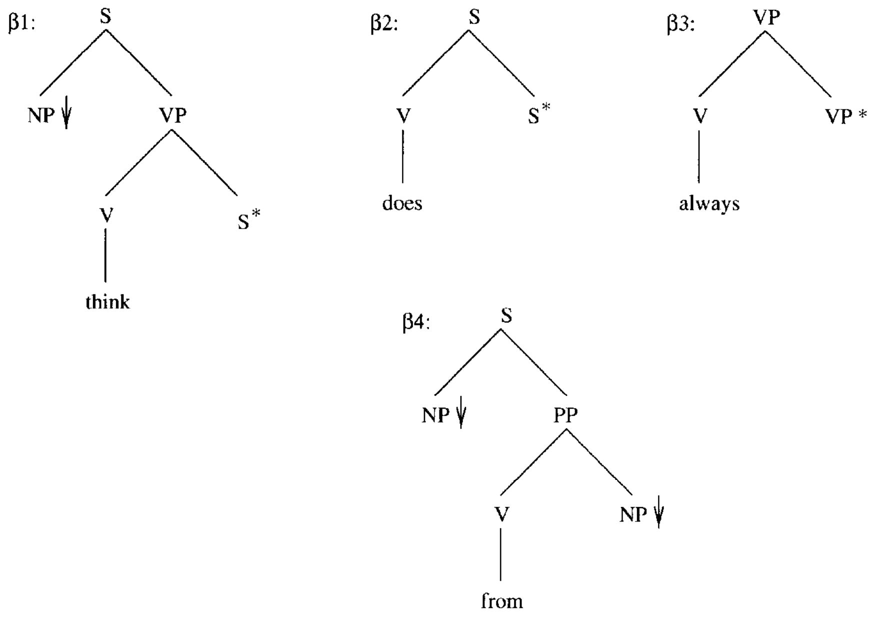


Fig. 4. LTAG examples 2.

substituted in the tree $\beta 1$ for *think* as shown (again by solid lines), resulting in a tree corresponding to *Bill think S*. Tree $\beta 2$ 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

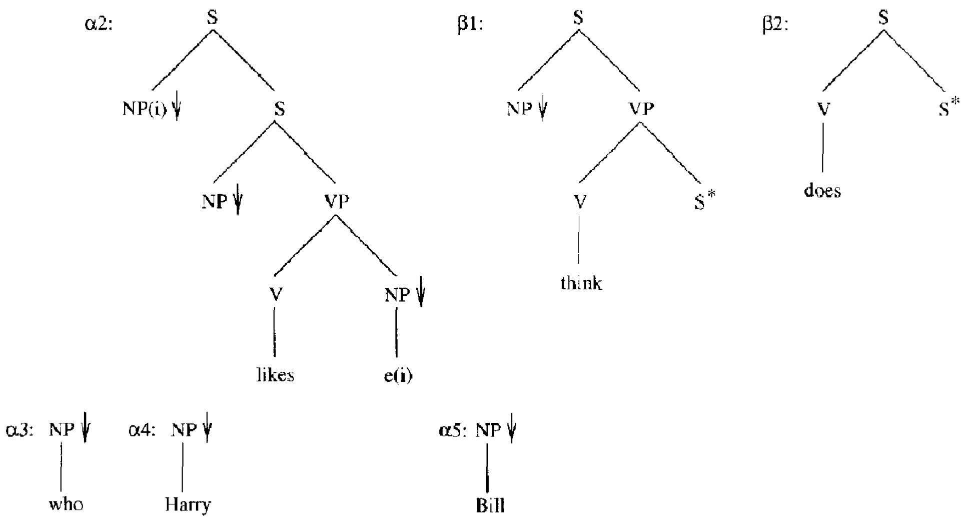


Fig. 5. LTAG examples 3.

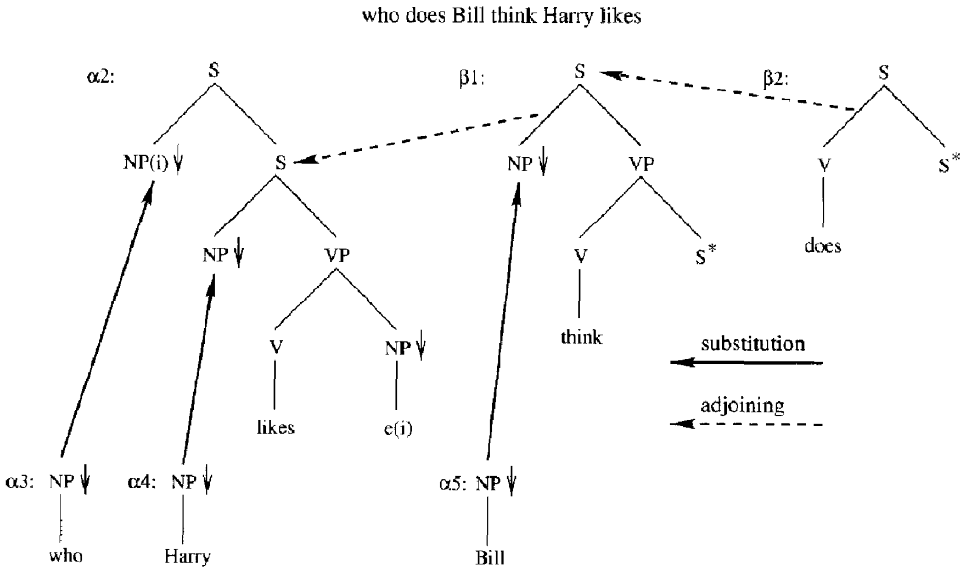
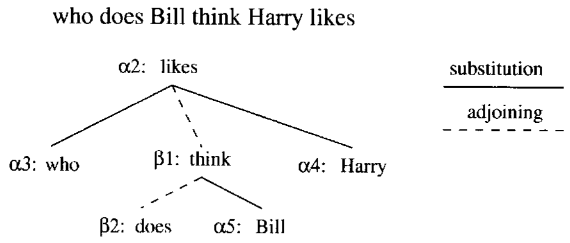


Fig. 6. LTAG examples 4.

to *who does Bill think Harry likes?* This derivation is shown in Fig. 7 where the solid lines represent substitution and the dotted lines represent adjoining. The representation in Fig. 7 is very close to the so-called dependency diagrams, which directly represent the dependencies among the lexical items.



* Compositional semantics on this derivation structure
* Related to dependency diagrams

Fig. 7. LTAG derivation structure.

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 extraction. I am not going to discuss these aspects as this is not the topic of my paper. 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, i.e., 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 Kilger in the Verbmobil Project (1997) and the work of Stone and Doran (1997) and Dras (1997). [8] The work on generation by Stone and Doran uses the LTAG framework for the design of a sentence planner using descriptions (SPUD), which takes in a collection of goals to achieve in describing an event or a state in the world. It incrementally and recursively applies lexical specifications to determine which entities to describe and what information to include about them. The operations of substitution and adjoining allows the possibility of construction 'descriptions' in a flexible manner. SPUD uses a declarative specification of three kinds of information: first what operators are available and how they combine; second, how operators specify the content of a description; and third, how operators achieve pragmatic effects. The operators are represented as the elementary trees of LTAG and the LTAG operations to combine them. Meaning for each tree is expressed as a formula in the so-called "flat" semantics

[7] Mathematical properties of LTAG have been extensively studied. 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] T. Becker, W. Finkler, A. Kilger, Generation in dialog translation: requirements, techniques, and their realization in Verbmobil, Technical Report, DFKI, University of Saarlandes, Saarbrücken, 1997. M. Stone, C. Doran, Sentence planning as description using tree-adjoining grammar, in: Proc. 35th Annual Meeting of the Association for Computational Linguistics, ACL-97, Madrid, 1997. M. Dras, Representing paraphrases using synchronous TAGs, in: Proc. 35th Annual Meeting of the Association for Computational Linguistics, ACL-97, Madrid, 1997.

representation and the pragmatics of the operators is modeled by associating with each tree a set of discourse constraints describing when the operator can and should be used.

- The constrained nature of LTAG is used for compiling HPSG (head-driven phrase structure grammar) into LTAG, also in the Verbmobil project (Kasper et al. (1995))[9] and in the JSPS project at the University of Tokyo headed by Professor Tsujii, where it has been used to compile LTAG into an HPSG, leading to a very efficient parser (Tsujii, Torisawa, Tateisi, Makino and Nishida, 1997 (personal communication).
- 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] The key aspects of TAG used in these works are the syntactic and semantic encapsulation in the domain of the elementary trees, localization of dependencies, and the operation of adjoining allowing the possibility of "modifying" already built structures. These abstract properties TAG what makes TAG attractive for modeling certain aspects of discourse. However, in the approaches taken so far the full potential of these abstract properties have not been utilized. Recently, Webber and Joshi (1998)[11] have explored a "fully" lexicalized TAG for discourse, allowing the possibility to examine how the basic insights of LTAG carry over to discourse. Just as lexicalized grammars have shown the value of taking the basic elements of a clause to be not simple words, but "structures" that reflect an item's role and local syntactic/semantic scope, there is value in taking the basic elements of discourse to be not simple clauses, but structures that reflect the syntactic/semantic scope of coherence relations.
- In a non-linguistic domain such as modeling complex dependencies in RNA secondary structures, the abstract character of LTAG has been exploited by Umeura et al. (1998) and by Abe and Mamitsuka (1994).[12]
- Localization of dependencies and the operations of substitution and adjoining allows us to capture complex dependency patterns in a "local" manner. For example, nested dependencies as in a German subordinate clause *Hans(i) Peter(j) Marie(k) schwimmen(k) lassen(j) sah(i)* (*Hans saw Peter make Marie swim*), where the nouns and verbs are in a nested order, as the subscripts indicate; however, in a Dutch subordinate clause, these dependencies are crossed, as in *Jan(i) Piet(j) Marie(k) zag(i) laten(j) zwemmen(k)* (*Jan saw Piet make Marie swim*). There are, of course, more complex dependencies such as combinations of nested and crossed

[9] R. Kasper, B. Kiefer, K. Netter, K. Vijayshanker, Compilation of HPSG to TAG, in: Proc. 33rd Annual Meeting of the Association for Computational Linguistics, ACL-95, Cambridge, 1995.

[10] C. Gardent, Discourse TAG, Technical Report, Computerlinguistik, University of Saarlandes, Saarbrücken, 1996. D. Cristea, B. Webber, Expectations in incremental discourse processing, in: Proc. 35th Annual Meeting of the Association for Computational Linguistics, ACL-97, Santa Cruz, 1997.

[11] B. Webber, A.K. Joshi, Anchoring a lexicalized tree-adjoining grammar for discourse, in: Proc. Workshop on Discourse Relations, COLING/ACL-1998 Workshop, Montreal, 1998.

[12] Y. Umeura, A. Hasegawa, S. Kobayashi, T. Yokomori, Tree-adjoining grammars for RNA structure prediction, Theoretical Computer Science, to appear in 1998. N. Abe, H. Mamitsuka, A new method of predicting protein secondary structures based on stochastic tree grammars, in: Proc. 11th International Conference on Machine Learning, 1994.

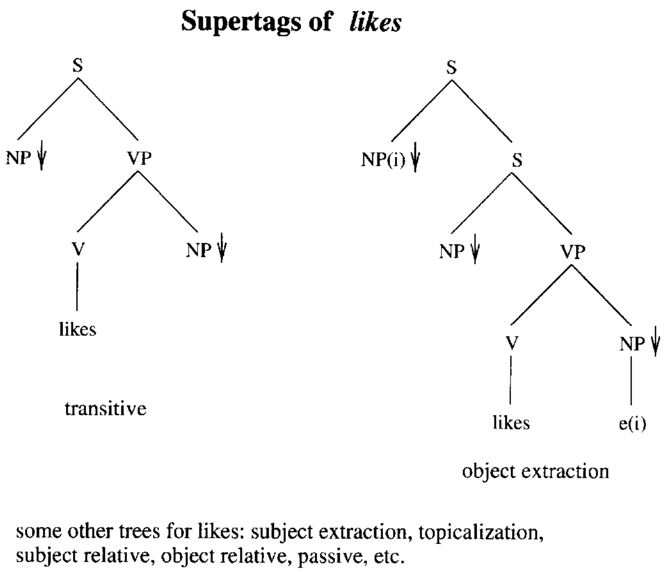

Supertags of *likes*
S
NP↓
VP
V
NP↓
likes
transitive
S
NP(i)↓
S
NP↓
VP
V
NP↓
likes
e(i)
object extraction
some other trees for likes: subject extraction, topicalization, subject relative, object relative, passive, etc.

Fig. 8. Supertags.

dependencies. There is a fascinating psycholinguistic result of Bach et al. (1986),[13] which shows that the crossing dependencies are "easier" to process than the nested dependencies. Now it turns out that the automaton that exactly corresponds to LTAG predicts this psycholinguistic result quite precisely. The relevance of this correspondence for the present paper is that such a result would not have been possible without modeling grammars by a constrained formal system such as LTAG.

2.1. 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 super part-of-speech (super POS or supertags) in contrast to the standard part-of-speech such as V (verb), N (noun) etc. Now it is well known that local statistical techniques can lead to 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.[14]

In Fig. 8 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 super part-of-speech (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

[13] E. Bach, C. Brown, W. Marslen-Wilson, Crossed and nested dependencies German and Dutch: a psycholinguistic study, Language and Cognitive Processes 1 (1986) 249–262.

[14] This is joint work with B. Srinivas.

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

Fig. 9. Supertagging.

Supertagging

	α1	$\boxed{\alpha 2}$	α3		α4	α5
	$\boxed{\beta 2}$	α6	α7		$\boxed{\beta 4}$	α8
$\boxed{\beta 1}$	α9	α10	$\boxed{\alpha 11}$	$\boxed{\beta 3}$	α12	$\boxed{\alpha 13}$
	.	.	.		.	.
	.	.	.		.	.
	.	.	.		.	.
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

Fig. 10. Supertagging.

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 Fig. 9. The supertags associated with that word appear on top of each word. Some words have only only one supertag associated with them and others have more than one. In the current XTAG system there are about 8–10 supertags per word on the average, so there is a very high level of local ambiguity. In Fig. 10 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 (WSJ). With a training corpus of 1 million words and a test corpus of 47 000 words, the baseline performance was 75% (i.e., 75% of the words received the "correct" supertag). The baseline corresponds to the case when 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% or better. The low baseline performance for supertagging is due to the fact that the local ambiguity is very high (about 8–10 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%. The improvement from 75% to 92% is indeed very remarkable. This means that 92% of the words received the "correct" supertag.

Of course, more can be said about this supertagging approach. There are techniques to improve the performance and to make the output look more like a complete parse. I will not discuss these aspects; rather, I will talk about the abstract nature of supertagging and its relevance to the use of constrained computational systems. 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, i.e., there are many more descriptions for each primitive, however, (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. Supertagging is thus an example of a local computation on complex descriptions.

The supertag perspective (i.e., regarding the elementary trees in LTAG associated with lexical items as complex part-of-speech) suggests a view of parsing consisting of only the resolution of *attachments* (substitution and adjoining in LTAG can both be viewed as attachments). Traditionally the so-called PP attachment problem (e.g., the attachment ambiguity in *I saw the man in the park with a telescope*) as a special problem. However, from the supertag perspective the PP attachment problem is just a special case of parsing where the entire process of parsing consists of resolution of attachments. In this perspective the parser does not build constituents (in the traditional sense) but only resolves attachments. This perspective also suggests some novel approaches to corpus based parsing using unsupervised techniques (currently being pursued by Anoop Sarkar).

These considerations are directly relevant to AI. I can illustrate this by pointing out interesting relationships to the well-known algorithm of Waltz (1975) [15] for interpreting line drawings. What Waltz did was to make the descriptions of vertices more complex by adding information about the number and types of edges 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. [16]

3. Local structure of discourse—centering

My last example concerns the use of constrained computational systems in the area of discourse, in particular, complexity of inferences in discourse. In order 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*. [17] 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 paper.

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 paper, I will call as 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

[15] D. Waltz, Understanding line drawings of scenes with shadows, in: P.H. Winston (Ed.), The Psychology of Computer Vision, McGraw-Hill, New York, 1975.

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

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

(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 (1879): [18]

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 judgement 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 (i.e., 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.

[18] G. Frege 1879, "Begriffsschrift" reproduced in Jean van Heijenroot (Ed.), From Frege to Gödel, Harvard University Press, Cambridge, 1967.

D1:
(a) John called Bill,
(b) He wanted his advice.
D2:
(a) John called Bill,
(b) He was happy to hear from him.

In D1 *he* in D1(b) refers to *John* and *his* to *Bill.* In D2, on the other hand, *he* in D2(b) 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 D2(b) the center has to be shifted from *John* to *Bill.* It is this shifting of the center in D2(b) that leads to the increased complexity. [19]

The main point here is that a discourse has a locally monadic structure. Monadic calculus is certainly a constrained system as 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 having 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 (Section 2.1), in the sense that complexity of the description of the primitives is related to the ease of inference.

4. Conclusion

So far I have given three examples of my research, which are relevant to the topic of this paper, namely, the use of constrained formal/computational systems in modeling various aspects of language. I will now briefly discuss some unifying issues which 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 which become more local, in other words, the greater the complexity of the primitives the more local the computations over them.

Statistics computed over primitives with complex descriptions are more meaningful in the sense that they capture the appropriate statistical dependencies (due to the localization of dependencies) and these in turn lead to efficient and robust computations.

[19] A psycholinguistic experiment is suggested here. The prediction is that in interpreting *he* D2(b) "attention" would first shift to *John* in D2(a) and then to *Bill* in D2(a). An experiment based on the head-mounted eyetracker technology is under consideration at present.

Adopting primitives with complex descriptions requires us to have richly annotated corpora of texts, dialogues, and various interactive situations, either obtained automatically or most likely, semi-automatically. 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* in order 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.

In summary, 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 languages 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 above. I am not giving the entire bibliography which involves these and all other collaborators due to the lack of space. Instead I want to give a public acknowledgement 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, Natasha Kurtonina, Anoop Sarkar, Martha Palmer, 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, Marilyn Walker.