Sentence Planning as Description Using Tree Adjoining Grammar[*]

Matthew Stone
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19014
matthew@linc.cis.upenn.edu

Christine Doran
Department of Linguistics
University of Pennsylvania
Philadelphia, PA 19014
cdoran@linc.cis.upenn.edu

Abstract

We present an algorithm for simultaneously constructing both the syntax and semantics of a sentence using a Lexicalized Tree Adjoining Grammar (LTAG). This approach captures naturally and elegantly the interaction between pragmatic and syntactic constraints on descriptions in a sentence, and the inferential interactions between multiple descriptions in a sentence. At the same time, it exploits linguistically motivated, declarative specifications of the discourse functions of syntactic constructions to make contextually appropriate syntactic choices.

1 Introduction

Since (Meteer, 1991), researchers in natural language generation have recognized the need to refine and reorganize content after the rhetorical organization of arguments and before the syntactic realization of phrases. This process has been named *sentence planning* (Rambow and Korelsky, 1992). Broadly speaking, it involves aggregating content into sentence-sized units, and then selecting the lexical and syntactic elements that are used in realizing each sentence. Here, we consider this second process.

The challenge lies in integrating constraints from syntax, semantics and pragmatics. Although most generation systems pipeline decisions (Reiter, 1994), we believe the most efficient and flexible way to integrate constraints in sentence planning is to synchronize the decisions. In this paper, we provide a natural framework for dealing with interactions and ensuring contextually appropriate output in a single pass. As in (Yang et al., 1991), Lexicalized Tree Adjoining Grammar (LTAG) provides an abstraction of the combinatorial properties of words. We combine LTAG syntax with declarative specifications of semantics and pragmatics of words and constructions, so that we can build the syntax and semantics of sentences simultaneously. To drive this process, we take *description* as the paradigm for sentence planning. Our planner, SPUD (Sentence Planner Using Descriptions), takes in a collection of goals to achieve in describing an event or state in the world; SPUD incrementally and recursively applies lexical specifications to determine which entities to describe and what information to include about them. Our system is unique in the streamlined organization of the grammar, and in its evaluation both of contextual appropriateness of pragmatics and of descriptive adequacy of semantics.

The organization of the paper is as follows. In section 2, we review research on generating referring expressions and motivate our treatment of sentences as referring expressions. Then, in section 3, we present the linguistic underpinnings of our work. In section 4, we describe our algorithm and its operation on an example. Finally, in section 5 we compare our system with related approaches.

2 Sentences as referring expressions

Our proposal is to treat the realization of sentences as parallel to the construction of referring expressions, and thereby bring to bear modern discourse-oriented theories of semantics and the idea that language use is INTENTIONAL ACTION.

Semantically, a DESCRIPTION D is just an open formula. D *applies* to a sequence of entities when substituting them for the variables in D yields a true formula. D REFERS to **c** just in case it *distinguishes* **c** from its DISTRACTORS—that is D applies to **c** but to no other salient alternatives. Given a sufficiently rich logical language, the meaning of a natural language sentence can be represented as a description in this sense, by assuming sentences refer to entities in a DISCOURSE MODEL, cf. alternative semantics (Karttunen and Peters, 1979; Rooth, 1985).

Pragmatic analyses of referring expressions model speakers as PLANNING those expressions to achieve several different kinds of intentions (Donellan, 1966; Appelt,

[*]The authors thank Aravind Joshi, Mark Steedman, Martha Palmer, Ellen Prince, Owen Rambow, Mike White, Betty Birner, and the participants of INLG96 for their helpful comments on various incarnations of this work. This work has been supported by NSF and IRCS graduate fellowships, NSF grant NSF-STC SBR 8920230, ARPA grant N00014-94 and ARO grant DAAH04-94-G0426.

1985; Kronfeld, 1986). Given a set of entities to describe and a set of intentions to achieve in describing them, a plan is constructed by applying operators that enrich the content of the description until all intentions are satisfied. Recent work on generating definite referring NPs (Reiter, 1991; Dale and Haddock, 1991; Reiter and Dale, 1992; Horacek, 1995) has emphasized how circumscribed instantiations of this procedure can exploit linguistic context and convention to arrive quickly at short, unambiguous descriptions. For example, (Reiter and Dale, 1992) apply generalizations about the salience of properties of objects and conventions about what words make base-level attributions to incrementally select words for inclusion in a description. (Dale and Haddock, 1991) use a constraint network to represent the distractors described by a complex referring NP, and incrementally select a property or relation that rules out as many alternatives as possible. Our approach is to extend such NP planning procedures to apply to sentences, using TAG syntax and a rich semantics.

Treating sentences as referring expressions allows us to encompass the strengths of many disparate proposals. Incorporating material into descriptions of a variety of entities until the addressee can infer desired conclusions allows the sentence planner to *enrich* input content, so that descriptions refer successfully (Dale and Haddock, 1991) or *reduce* it, to eliminate redundancy (McDonald, 1992). Moreover, selecting alternatives on the basis of their syntactic, semantic, and pragmatic contributions to the sentence using TAG allows the sentence planner to choose words in tandem with appropriate syntax (Yang et al., 1991), in a flexible order (Elhadad and Robin, 1992), and, if necessary, in conventional combinations (Smadja and McKeown, 1991; Wanner, 1994).

3 Linguistic Specifications

Realizing this procedure requires a declarative specification of three kinds of information: first, what operators are available and how they may combine; second, how operators specify the content of a description; and third, how operators achieve pragmatic effects. We represent operators as elementary trees in LTAG, and use TAG operations to combine them; we give the meaning of each tree as a formula in an ontologically promiscuous representation language; and, we model the pragmatics of operators by associating with each tree a set of discourse constraints describing when that operator can and should be used.

Other frameworks have the capability to make comparable specifications; for example, HPSG (Pollard and Sag, 1994) feature structures describe syntax (SUBCAT), semantics (CONTENT) and pragmatics (CONTEXT). We choose TAG because it enables local specification of syntactic dependencies in explicit constructions and flexibility in incorporating modifiers; further, it is a constrained grammar formalism with tractable computational properties.

3.1 Syntactic specification

TAG (Joshi et al., 1975) is a grammar formalism built around two operations that combine pairs of trees, SUBSTITUTION and ADJOINING. A TAG grammar consists of a finite set of ELEMENTARY trees, which can be combined by these substitution and adjoining operations to produce derived trees recognized by the grammar. In substitution, the root of the first tree is identified with a leaf of the second tree, called the substitution site. Adjoining is a more complicated splicing operation, where the first tree replaces the subtree of the second tree rooted at a node called the adjunction site; that subtree is then substituted back into the first tree at a distinguished leaf called the FOOT node. Elementary trees without foot nodes are called INITIAL trees and can only substitute; trees with foot nodes are called AUXILIARY trees, and must adjoin. (The symbol $\downarrow$ marks substitution sites, and the symbol $*$ marks the foot node.) Figure 1(a) shows an initial tree representing *the book*. Figure 1(b) shows an auxiliary tree representing the modifier *syntax*, which could adjoin into the tree for *the book* to give *the syntax book*.

Our grammar incorporates two additional principles. First, the grammar is LEXICALIZED (Schabes, 1990): each elementary structure in the grammar contains at least one lexical item. Second, our trees include FEATURES, following (Vijay-Shanker, 1987).

LTAG elementary trees abstract the combinatorial properties of words in a linguistically appealing way. All predicate-argument structures are localized within a single elementary tree, even in long-distance relationships, so elementary trees give a natural domain of locality over which to state semantic and pragmatic constraints. The LTAG formalism does not dictate particular syntactic analyses; ours follow basic GB conventions.

3.2 Semantics

We specify the semantics of trees by applying two principles to the LTAG formalism. First, we adopt an ONTOLOGICALLY PROMISCUOUS representation (Hobbs, 1985) that includes a wide variety of types of entities. Ontological promiscuity offers a simple syntax-semantics interface. The meaning of a tree is just the CONJUNCTION of the meanings of the elementary trees used to derive it, once appropriate parameters are recovered. Such flat semantics is enjoying a resurgence in NLP; see (Copestake et al., 1997) for an overview and formalism. Second, we constrain these parameters syntactically, by labeling each syntactic node as supplying information about a particular entity or collection of entities, as in Jackendoff's X-bar semantics (Jackendoff, 1990). A node X:x (about x) can only substitute or adjoin into another node with the same label. These semantic parameters are instantiated using a knowledge base (cf. figure 7).

For Jackendoff, noun phrases describe ordinary individuals, while PPs describe PLACES or PATHS and VPs describe ACTIONS and EVENTUALITIES (in terms of a Reichenbachian reference point). Under these assumptions,

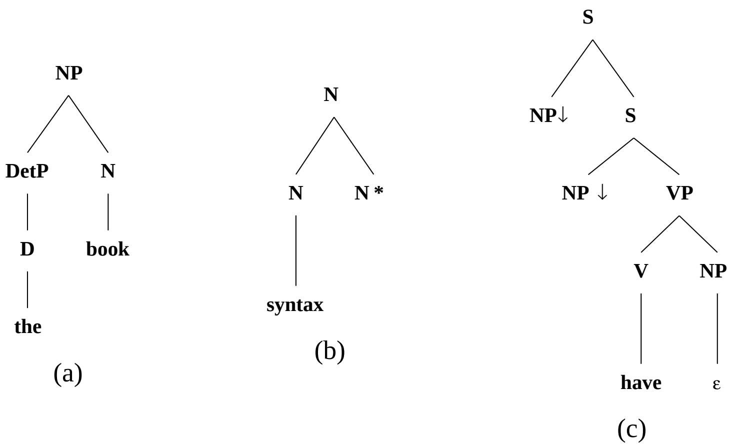


Figure 1: Sample LTAG trees: (a) NP, (b) Noun-Noun Compound, (c) Topicalized Transitive

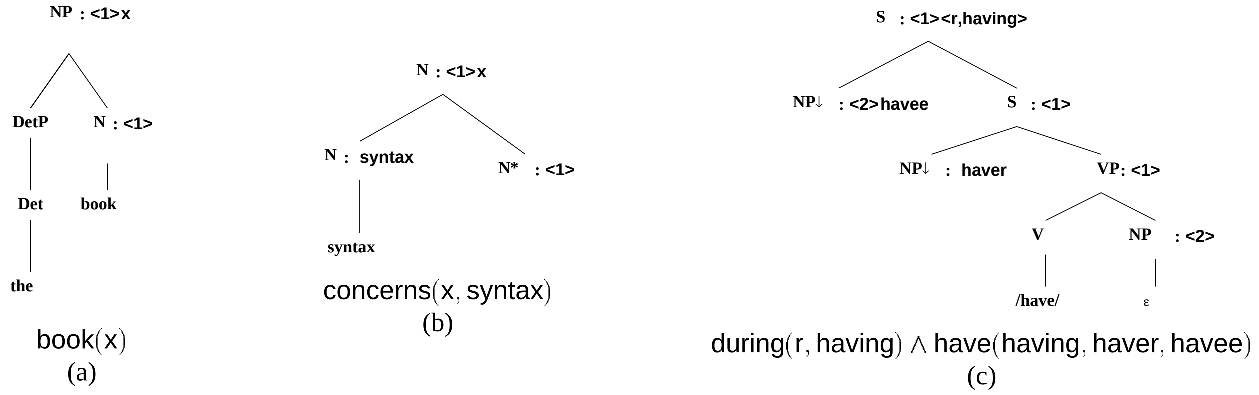


Figure 2: LTAG trees with semantic specifications

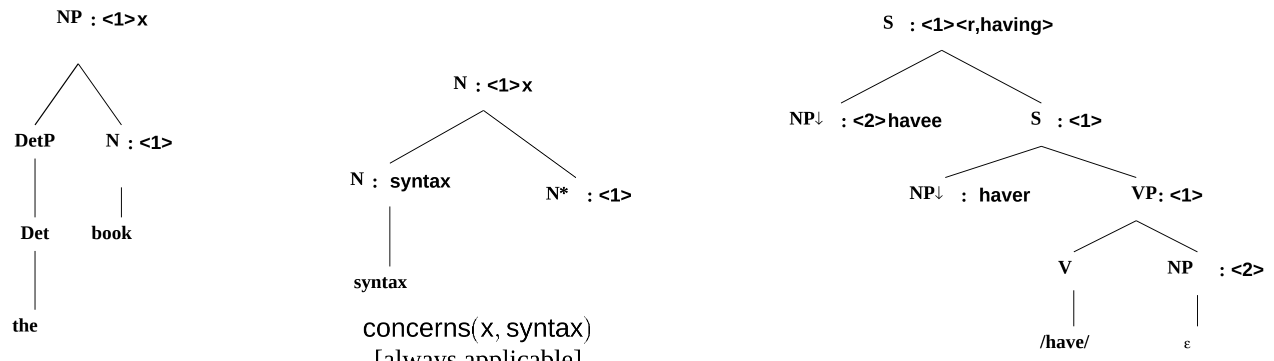
the trees of figure 1 are elaborated for semantics as in figure 2. Ontological promiscuity makes it possible to explore more complicated analyses in this general framework. For example, in (Stone and Doran, 1996), we use reference to properties, actions and belief contexts (Ballim et al., 1991) to describe semantic collocations (Pustejovsky, 1991) and idiomatic composition (Nunberg et al., 1994).

3.3 Pragmatics

Different constructions make different assumptions about the status of entities and propositions in the discourse, which we model by including in each tree a specification of the contextual conditions under which use of the tree is pragmatically licensed. We have selected four representative pragmatic distinctions for our implementation; however, the framework does not commit one to the use of particular theories.

We use the following distinctions. First, entities differ in NEWNESS (Prince, 1981). At any point, an entity is either new or old to the HEARER and either new or old to the DISCOURSE. Second, entities differ in SALIENCE (Grosz and Sidner, 1986; Grosz et al., 1995). Salience assigns each entity a position in a partial order that indicates how accessible it is for reference in the current context. Third, entities are related by salient PARTIALLY-ORDERED SET (POSET) RELATIONS to other entities in the context (Hirschberg, 1985). These relations include part and whole, subset and superset, and membership in a common class. Finally, the discourse may distinguish some OPEN PROPOSITIONS (propositions containing free variables) as being under discussion (Prince, 1986). We assume that information of these four kinds is available in a model of the current discourse state.

The applicability conditions of constructions can freely make reference to this information. In particular, NP trees include the determiner (the determiner does not have a separate tree), the head noun, and pragmatic conditions that match the determiner with the status of the entity in context, as in 3(a). Following (Gundel et al., 1993), the definite article *the* may be used when the entity is UNIQUELY IDENTIFIABLE in the discourse model, i.e. the hearer knows or can infer the existence of this entity and can distinguish it from any other hearer-old entity of equal or greater salience. (Note that this test only determines the status of the entity in context; we ensure separately that the sentence includes enough content to distinguish the entity from all its alternatives.) In contrast, the indef-



book(x)
⟨unique-id(x)⟩
(a)

concerns(x, syntax)
[always applicable]
(b)

during(r, having) ∧ have(having, haver, havee)
⟨in-poset(havee), in-op(have(having, haver, havee))⟩
(c)

Figure 3: LTAG trees with semantic and pragmatic specifications

inite articles, *a*, *an*, and Ø, are used for entities that are NOT uniquely identifiable.

S trees specify the main verb and the number and position of its arguments. Our S trees specify the unmarked SVO order or one of a number of fancy variants: topicalization (TOP), left-dislocation (LD), and locative inversion (INV). We follow the analysis of TOP in (Ward, 1985). For Ward, TOP is not a topic-marking construction at all. Rather, TOP is felicitous as long as (1) the fronted NP is in a salient poset relation to the previous discourse and (2) the utterance conveys a salient open proposition which is formed by replacing the tonically stressed constituent with a variable (3(c)). Likewise, we follow (Prince, 1993) and (Birner, 1992) for LD and INV respectively.

4 SPUD

4.1 The algorithm

Our system takes two types of goals. First, goals of the form *distinguish x as cat* instruct the algorithm to construct a description of entity *x* using the syntactic category *cat*. If *x* is uniquely identifiable in the discourse model, then this goal is only satisfied when the meaning planned so far distinguishes *x* for the hearer. If *x* is hearer new, this goal is satisfied by including any constituent of type *cat*. Second, goals of the form *communicate p* instruct the algorithm to include the proposition *p*. This goal is satisfied as long as the sentence IMPLIES *p* given shared common-sense knowledge.

In each iteration, our algorithm must determine the appropriate elementary tree to incorporate into the current description. It performs this task in two steps to take advantage of the regular associations between words and trees in the lexicon. Sample lexical entries are shown in figure 4. They associate a word with the semantics of the word, special pragmatic restrictions on the use of the word, and a set of trees that describe the combinatory possibilities for realizing the word and may impose additional pragmatic restrictions. Tree types are shared between lexical items (figure 5). This allows us to specify the pragmatic constraints associated with the tree type once, regardless of which verb selects it. Moreover, we can determine which tree to use by looking at each tree ONCE per instantiation of its arguments, even when the same tree is associated with multiple lexical items.

Hence, the first step is to identify applicable lexical entries by meaning: these items must truly and appropriately describe some entity; they must anchor trees that can substitute or adjoin into a node that describes the entity; and they must distinguish entities from their distractors or entail required information. Then, the second step identifies which of the associated trees are applicable, by testing their pragmatic conditions against the current representation of discourse. The algorithm identifies the combinations of words and trees that satisfy the most *communicate* goals and eliminate the most distractors. From these, it selects the entry with the most specific semantic and pragmatic licensing conditions. This means that the algorithm generates the most marked licensed form. In (Stone and Doran, 1996) we explore the use of additional factors, such as attentional state and lexical preferences, in this step.

The new tree is then substituted or adjoined into the existing tree at the appropriate node. The entry may specify additional goals, because it describes one entity in terms of a new one. These new goals are added to the current goals, and then the algorithm repeats.

Note that this algorithm performs greedy search. To avoid backtracking, we choose uninflected forms. Morphological features are set wherever possible as a result of the general unification processes in the grammar; the inflected form is determined from the lemma and its associated features in a post-processing step.

The specification of this algorithm is summarized in the following pseudocode:

STEM	SEMANTICS	SYNTAX	PRAGMATICS
/buy/	S	buyer */buy/* bought */from/* seller, etc.	register(informal)
/sell/	S	seller */sell/* bought */to/* buyer, etc.	register(informal)
/purchase/	S	buyer */purchase/* bought */from/* seller, etc.	register(formal)
/book/	book(x)	*/a/ /book/*, etc.	[always possible]

S = buy(buying,buyer,seller,bought)

Figure 4: Sample entries from the lexicon

SUBCAT FRAME	TREES	PRAGMATICS
Intransitive	Active	[always possible]

Transitive	Active
	Topicalized Object
	Left-Dislocated Object
		[always possible]
		in-poset(obj), in-op(event)
		in-poset(obj)

Ditransitive	Active
	Topicalized Dir Object
	Left-Dislocated Dir Object
	⋮
		[always possible]
		in-poset(dir obj), in-op(event)
		in-poset(dir obj)

PP Predicative	Active Locative Inversion	[always possible] newer-than(subj,loc)
etc.		

Figure 5: Sample entries from the tree database

```
until goals are satisfied:
    determine which uninflected forms apply;
    determine which associated trees apply;
    evaluate progress towards goals;
    incorporate most specific, best ⟨ form, tree ⟩:
        perform adjunction or substitution;
        conjoin new semantics;
        add any additional goals;
```

4.2 The system

SPUD's grammar currently includes a range of syntactic constructions, including adjective and PP modification, relative clauses, idioms and various verbal alternations. Each is associated with a semantic and pragmatic specification as discussed above and illustrated in figures 4 and 5. These linguistic specifications can apply across many domains.

In each domain, an extensive set of inferences, presumed known in common with the user, are required to ensure appropriate behavior. We use logic programming to capture these inferences. In our domain, the system has the role of a librarian answering patrons' queries. Our specifications define: the properties and identities of objects (e.g., attributes of books, parts of the library); the taxonomic relationships among terms (e.g., that a service desk is an area but not a room); and the typical span and course of events in the domain (e.g., rules about how to check out books). This information is complete and available for each lexical entry. Of course, SPUD also represents its private knowledge about the domain. This includes facts like the status of books in the library.

4.3 An example

Suppose our system is given the task of answering the following question:

(1) Do you have the books for Syntax 551 and Pragmatics 590?

Figure 6 shows part of the discourse model after processing the question. The two books, the set they comprise (introducing a poset relation), and the library are mentioned in (1). Hence, these entities must be both hearer-old and discourse-old. As in centering (Grosz et al., 1995), the subject is taken to be the most salient entity. Finally, the meaning of the question becomes a salient open proposition.

On the basis of the knowledge in figure 7, a rhetorical planner might decide to answer by describing state have27 as an *S* and lose5 likewise. To construct its reference to have27, SPUD first determines which lexical and syntactic options are available. Using the lexicon and information about have27 available from figure 7(b), SPUD determines that, of lemmas that truthfully and appropriately describe have27 as an *S*, */have/* has the most specific licensing conditions. The tree set for */have/* includes unmarked, LD and TOP trees. All are licensed, because of the poset relation *R* between book19 and books and the salient open proposition *O*. We choose TOP, the tree with the most specific condition—TOP requires *R* and *O*, while LD requires only *R* and the unmarked form has no requirements.

Thus, a topicalized */have/* tree, appropriately instantiated as shown in figure 8, is added to the description. The tree refers to three new entities, the object book19,

STATUS	ENTITIES
DISCOURSE OLD:	*book19*, *book2*, *books*, *library*, *patron*
HEARER OLD:	*book19*, *book2*, *books*, *library*, *patron*
SALIENCE:	$\{library, patron\} > \{book19, book2, books\}$
POSET RELATIONS:	*book19* MEMBER-OF *books*; *book2* MEMBER-OF *books*
OPEN PROPOSITION:	*library X have Y* : *X*={does/doesn't}; $Y \in$ *books*.

Figure 6: Discourse model for the example

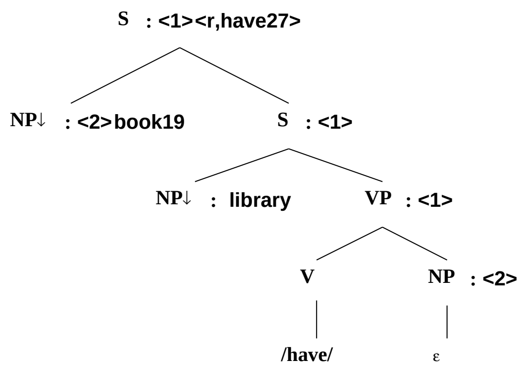
book(book19)
book(book2)
concerns(book19, syntax)
concerns(book2, pragmatics)

(a) Common Knowledge

have(have27, library, book19)
lost(lose5, book2)
during(have27, now)
during(lose5, now)

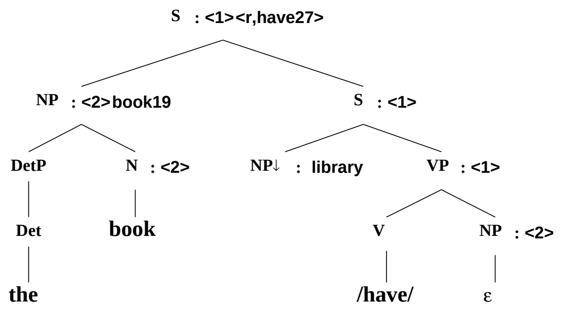
(b) Speaker's Knowledge

Figure 7: Knowledge bases for the example



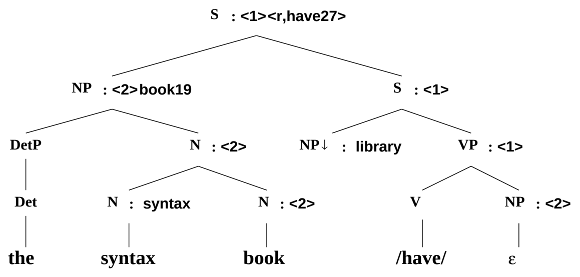
during(r, have27) ∧ have(have27, library, book19)

Figure 8: The first tree incorporated into the description of have27.



during(r, have27) ∧
have(have27, library, book19) ∧ book(book19)

Figure 9: The description of have27 after substituting *the book*.



during(r, have27) ∧ have(have27, library, book19) ∧
book(book19) ∧ concerns(book19, syntax)

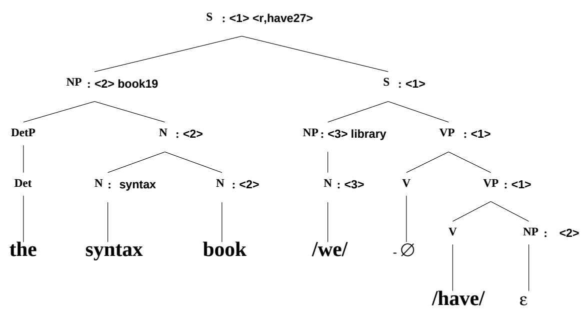
Figure 10: The tree with the complete description of book19.

the subject library and the reference point r of the tense. Subgoals of distinguishing these entities are introduced. Other constructions that describe one entity in terms of another, such as complex NPs, relative clauses and semantic collocations, are also handled this way by SPUD.

The algorithm now selects the goal of describing book19 as an NP. Again, the knowledge base is consulted to select NP lemmas that truly describe book19. The best is */book/*. (Note that SPUD would use *it* if either the verb or the discourse context ruled out all distractors.) The tree set for */book/* includes trees with definite and indefinite determiners; since the hearer can uniquely identify book19, the definite tree is selected and substituted as the leftmost NP, as in figure 9.

The goal of distinguishing book19 is still not satisfied, however; as far as the hearer knows, both book19 and book2 could match the current description. We consult the knowledge base for further information about book19. The modifier entry for the lexical item */syntax/* can apply to the new *N* node; its tree adjoins there, giving the tree in figure 10. Note that because trees are lexicalized and instantiated, and must unify with the existing derivation, SPUD can enforce collocations and idiomatic composition in steps like this one.

Now we describe the library. Since we ARE the library, the lexical item */we/* is chosen. Finally, we describe r. Consulting the knowledge base, we determine that r is now, and that the present tense morpheme applies. For uniformity with auxiliary verbs, we represent it as a separate tree, in this case with a null head, which assigns a morphological feature to the main verb. This gives the



$$\mathsf{during}(\mathsf{r},\mathsf{have27}) \wedge \mathsf{have}(\mathsf{have27},\mathsf{library},\mathsf{book19}) \wedge \mathsf{book}(\mathsf{book19}) \wedge \mathsf{concerns}(\mathsf{book19},\mathsf{syntax}) \wedge \mathsf{we}(\mathsf{library}) \wedge \mathsf{pres}(\mathsf{r},\mathsf{have27})$$

Figure 11: The final tree.

tree in figure 11, representing the sentence:

(2) The syntax book, we have.

All goals are now satisfied. Note that the semantics has been accumulated incrementally and straightforwardly in parallel with the syntax.

To illustrate the role of inclusion goals, let us suppose that the system also knows that book19 is on reserve in the state have27. Given the additional input goal of communicating this fact, the algorithm would proceed as before, deriving *The syntax book we have.* However, the new goal would still be unsatisfied; in the next iteration, the PP *on reserve* would be adjoined into the tree to satisfy it: *The syntax book, we have on reserve.* Because TAG allows adjunction to apply at any time, flexible realization of content is facilitated without need for sophisticated back-tracking (Elhadad and Robin, 1992).

The processing of this example may seem simple, but it illustrates the way in which SPUD integrates syntactic, semantic and pragmatic knowledge in realizing sentences. We tackle additional examples in (Stone and Doran, 1996).

5 Comparison with related work

The strength of the present work is that it captures a number of phenomena discussed elsewhere separately, and does so within a unified framework. With its incremental choices and its emphasis on the consequences of functional choices in the grammar, our algorithm resembles the networks of systemic grammar (Mathiessen, 1983; Yang et al., 1991). However, unlike systemic networks, our system derives its functional choices dynamically using a simple declarative specification of function. Like many sentence planners, we assume that there is a flexible association between the content input to a sentence planner and the meaning that comes out. Other researchers (Nicolov et al., 1995; Rubinoff, 1992) have assumed that this flexibility comes from a mismatch between input content and grammatical options. In our system, such differences arise from the referential requirements and inferential opportunities that are encountered.

Previous authors (McDonald and Pustejovsky, 1985; Joshi, 1987) have noted that TAG has many advantages for generation as a syntactic formalism, because of its localization of argument structure. (Joshi, 1987) states that adjunction is a powerful tool for elaborating descriptions. These aspects of TAGs are crucial to SPUD, as they are to (McDonald and Pustejovsky, 1985; Joshi, 1987; Yang et al., 1991; Nicolov et al., 1995; Wahlster et al., 1991; Danlos, 1996). What sets SPUD apart is its simultaneous construction of syntax and semantics, and the tripartite, lexicalized, declarative grammatical specifications for constructions it uses. Two contrasts should be emphasized in this regard. (Shieber et al., 1990; Shieber and Schabes, 1991) construct a simultaneous *derivation* of syntax and semantics but they do not *construct the semantics*—it is an input to their system. (Prevost and Steedman, 1993; Hoffman, 1994) represent syntax, semantics and pragmatics in a lexicalized framework, but concentrate on information structure rather than the pragmatics of particular constructions.

6 Conclusion

Most generation systems pipeline pragmatic, semantic, lexical and syntactic decisions (Reiter, 1994). With the right formalism, constructing pragmatics, semantics and syntax simultaneously is easier and better. The approach elegantly captures the interaction between pragmatic and syntactic constraints on descriptions in a sentence, and the inferential interactions between multiple descriptions in a sentence. At the same time, it exploits linguistically motivated, declarative specifications of the discourse functions of syntactic constructions to make contextually appropriate syntactic choices.

References

D. Appelt. 1985. *Planning English Sentences*. Cambridge University Press.

A. Ballim, Y. Wilks, and J. Barnden. 1991. Belief ascription, metaphor, and intensional identification. *Cognitive Science*, 15:133–171.

B. Birner. 1992. *The Discourse Function of Inversion in English*. Ph.D. thesis, Northwestern University.

A. Copestake, D. Flickinger, and I. A. Sag. 1997. Minimal Recursion Semantics An Introduction. MS, CSLI, http://hpsg.stanford.edu/hpsg/sag.html.

R. Dale and N. Haddock. 1991. Content determination in the generation of referring expressions. *Computational Intelligence*, 7(4):252–265.

L. Danlos. 1996. G-TAG: A formalism for Text Generation inspired from Tree Adjoining Grammar: TAG issues. Unpublished manuscript, TALANA, Université Paris 7.

K. Donellan. 1966. Reference and definite description. *Philosophical Review*, 75:281–304.

M. Elhadad and J. Robin. 1992. Controlling content realization with functional unification grammars. In Dale, Hovy, Rösner, and Stock, editors, *Aspects of Automated Natural Language Generation: 6th International Workshop on Natural Language Generation*, pages 89–104. Springer Verlag.

B. Grosz and C. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12:175–204.

B. J. Grosz, A. K. Joshi, and S. Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

J. K. Gundel, N. Hedberg, and R. Zacharski. 1993. Cognitive status and the form of referring expressions in discourse. *Language*, 69(2):274–307.

J. Hirschberg. 1985. *A Theory of Scalar Implicature*. Ph.D. thesis, University of Pennsylvania.

J. R. Hobbs. 1985. Ontological promiscuity. In *ACL*, pages 61–69.

B. Hoffman. 1994. Generating context-appropriate word orders in Turkish. In *Seventh International Generation Workshop*.

H. Horacek. 1995. More on generating referring expressions. In *Fifth European Workshop on Natural Language Generation*, pages 43–58, Leiden.

R. S. Jackendoff. 1990. *Semantic Structures*. MIT Press.

A. K. Joshi, L. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of the Computer and System Sciences*, 10:136–163.

A. K. Joshi. 1987. The relevance of tree adjoining grammar to generation. In Kempen, editor, *Natural Language Generation*, pages 233–252. Martinus Nijhoff Press, Dordrect, The Netherlands.

L. Karttunen and S. Peters. 1979. Conventional implicature. In Oh and Dineen, editors, *Syntax and Semantics 11: Presupposition*. Academic Press.

A. Kronfeld. 1986. Donellan's distinction and a computational model of reference. In *ACL*, pages 186–191.

C. M. I. M. Mathiessen. 1983. Systemic grammar in computation: the Nigel case. In *EACL*, pages 155–164.

D. D. McDonald and J.. Pustejovsky. 1985. TAG's as a grammatical formalism for generation. In *ACL*, pages 94–103.

D. McDonald. 1992. Type-driven suppression of redundancy in the generation of inference-rich reports. In Dale, Hovy, Rösner, and Stock, editors, *Aspects of Automated Natural Language Generation: 6th International Workshop on Natural Language Generation*, pages 73–88. Springer Verlag.

M. W. Meteer. 1991. Bridging the generation gap between text planning and linguistic realization. *Computational Intelligence*, 7(4):296–304.

N. Nicolov, C. Mellish, and G. Ritchie. 1995. Sentence generation from conceptual graphs. In W. Rich G. Ellis, R. Levinson and F. Sowa, editors, *Conceptual Structures: Applications, Implementation and Theory*, pages 74–88. Springer.

G. Nunberg, I. A. Sag, and T. Wasow. 1994. Idioms. *Language*, 70(3):491–538.

C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. CSLI.

S. Prevost and M. Steedman. 1993. Generating contextually appropriate intonation. In *EACL*.

E. Prince. 1981. Toward a taxonomy of given-new information. In P. Cole, editor, *Radical Pragmatics*. Academic Press.

E. Prince. 1986. On the syntactic marking of presupposed open propositions. In *CLS*, pages 208–222, Chicago. CLS.

E. Prince. 1993. On the functions of left dislocation. Manuscript, University of Pennsylvania.

J. Pustejovsky. 1991. The generative lexicon. *Computational Linguistics*, 17(3):409–441.

O. Rambow and T. Korelsky. 1992. Applied text generation. In *ANLP*, pages 40–47.

E. Reiter and R. Dale. 1992. A fast algorithm for the generation of referring expressions. In *Proceedings of COLING*, pages 232–238.

E. Reiter. 1991. A new model of lexical choice for nouns. *Computational Intelligence*, 7(4):240–251.

E. Reiter. 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 163–170.

M. Rooth. 1985. *Association with focus*. Ph.D. thesis, University of Massachusetts.

R. Rubinoff. 1992. Integrating text planning and linguistic choice by annotating linguistic structures. In Dale, Hovy, Rösner, and Stock, editors, *Aspects of Automated Natural Language Generation: 6th International Workshop on Natural Language Generation*, pages 45–56. Springer Verlag.

Y. Schabes. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, University of Pennsylvania.

S. Shieber and Y. Schabes. 1991. Generation and synchronous tree adjoining grammars. *Computational Intelligence*, 4(7):220–228.

S. Shieber, G. van Noord, F. Pereira, and R. Moore. 1990. Semantic-head-driven generation. *Computational Linguistics*, 16:30–42.

F. Smadja and K. McKeown. 1991. Using collocations for language generation. *Computational Intelligence*, 7(4):229–239.

K. Vijay-Shanker. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, University of Pennsylvania.

M. Stone and C. Doran. 1996. Paying heed to collocations. In *Eighth International Workshop on Natural Language Generation 96*, pages 91–100.

W. Wahlster, E. André, S. Bandyopadhyay, W. Graf, and T. Rist. 1991. WIP: The coordinated generation of multimodal presentations from a common representation. In Stock, Slack, and Ortony, editors, *Computational Theories of Communication and their Applications*. Springer Verlag.

L. Wanner. 1994. Building another bridge over the generation gap. In *Seventh International Workshop on Natural Language Generation*, pages 137–144, June.

G. Ward. 1985. *The Semantics and Pragmatics of Preposing*. Ph.D. thesis, University of Pennsylvania.

G. Yang, K. F. McCoy, and K. Vijay-Shanker. 1991. From functional specification to syntactic structures: systemic grammar and tree-adjoining grammar. *Computational Intelligence*, 7(4):207–219.