Bidirectional Grammars and the Design of Natural Language Generation Systems

Douglas E. Appelt

Artificial Intelligence Center
Center for the Study of Language and Information
SRI International

1 Bidirectional Grammars for Generation

Intuitively considered, a grammar is bidirectional if it can be used by processes of approximately equal computational complexity to parse and generate sentences of a language. Because we, as computational linguists, are concerned with the *meaning* of the sentences we process, a bidirectional grammar must specify a correspondence between sentences and meaning representations, and this correspondence must be represented in a manner that allows one to be computed from the other. Most research in computational linguistics has focused on one or the other of the two sides of the problem, with the result that relatively little attention has been given to the issues raised by the incorporation of a single grammar into a system for tasks of both comprehension and generation.

Clearly, if it were possible to have truly bidirectional grammars in which both parsing and generation processes were efficient, there would be some compelling reasons for adopting them. First, Occam's razor suggests that, if language behavior can be explained by hypothesizing only one linguistic representation, such an explanation is clearly preferable to two that are applicable in complementary circumstances. Also, from the practical standpoint of designing systems that will carry on sophisticated dialogues with their users, a single unified formalism for specifying the syntax and semantics of the language is likely to result in a simpler, more robust implementation. The problems of maintaining consistency between comprehension and generation components when one of them changes have been eliminated. The lexicon is also simpler because its entries need be made but once, and there is no problem of maintaining consistency between different lexical entries for understanding and generation.

It is obvious that not all grammars are bidirectional. The most fundamental requirement of any bidirectional grammar is that it be represented declaratively. If any information is represented procedurally, it must of necessity be represented differently for parsing and generation processes, resulting in an asymmetry between the two. Any change in the grammar would have to be made in two places to maintain the equivalence between the syntactic and semantic analyses given to sentences by each process. A grammar like DIAGRAM [8] is an example of a grammar for which the encoding of linguistic information

is primarily procedural; it is inconceivable how it could be used for generation.

Also, reversibility requires that the grammar define a one-to-one mapping from surface strings to some meaning representation. Presumably this representation would consist of a logical form specifying the predicate argument structure of the sentence, together with a set of functional features that distinguish sentences according to their pragmatic or functional role. For example, active and passive sentences have the same logical form, but different functional features.

The PATR-II formalism [9], which is based on the unification of feature structures, has properties that make a bidirectional grammar possible. This formalism has been demonstrated to be very useful in encoding linguistic information and accommodates a wide variety of linguistic theories [10,12]. The PATR-II formalism has many elegant formal properties, including a denotational semantics [7], but the one most important for bidirectionality is that the unification operation is associative and commutative. This implies that the result of unifying feature structures is independent of the order in which they are unified. This characteristic allows one to write grammar rules that satisfy the two properties cited above, without incorporating into the structure of the rules themselves any assumptions about the process that will employ these rules . Shieber[1] has developed a generation system based on PATR-II grammar rules. The grammar is bidirectional; given a logical form that a parser would have produced had it been given the sentence to parse, the generator will produce the same sentence. All features of the analysis are identical in both cases. If the combination of logical form and functional features is insufficient to determine a unique sentence, the generator can produce a set of sentences whose meanings unify with the specification.

2 Implications of Bidirectionality for System Design

Adopting a bidirectional grammar for a language-understanding and -generation system implies certain constraints on the system's design. Because the grammar for such a system must consist of declarative rules to be interpreted, it must provide exactly the same information to both parsing and generation processes. This implies that at least the lowest level of these processes must be symmetric.

The role the grammar plays in most understanding systems is to define a mapping from surface utterances to a logical form that abstracts predicate-argument structure and quantifier scoping from the sentence. This logical form provides a basis from which inferences are drawn, both to resolve anaphora and to determine the speaker's intentions behind the utterance. The symmetry requirement specifies that the generation process must produce a logical form (together with functional features) that determines the utterance to be produced. Figure 1 illustrates this basic design.

[1] Work in progress.

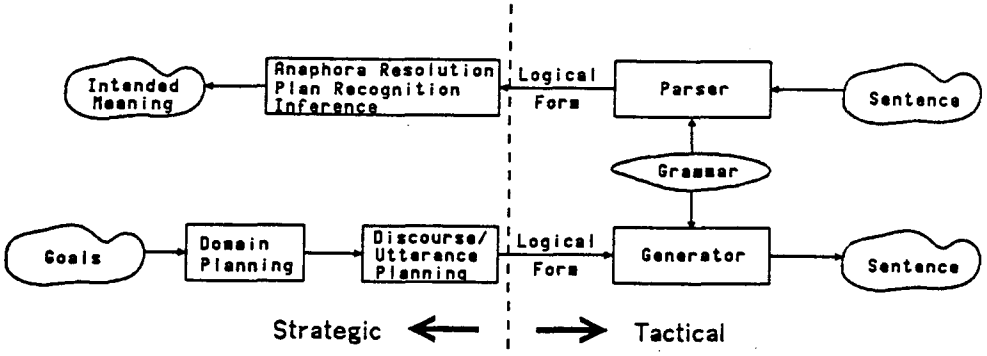

Figure 1: Organization of a Bidirectional System

In most understanding systems there is an easily identifiable boundary between the parsing/morphological component and the part of the system that draws inferences from the resulting logical form. The former is the only component that is concerned directly with the form and content of the grammatical rules, while the latter is the only one that is called upon to do general reasoning. It has been argued that intermediate fragments should undergo semantic and pragmatic analysis as an aid in resolving ambiguities such as prepositional-phrase attachment, as well as for inferring the intentions behind ill-formed input. At present, however, syntactic analysis has been sufficiently cheaper than semantic and pragmatic analysis to nullify any advantage that might be gained from integration of parsing and general inference. In any case, the inference procedures, while perhaps requiring access to certain features of the syntactic and semantic analysis, need not be concerned with the rules themselves. This modularity is clearly beneficial. The grammar, parser, and morphological analyzer, being a more or less self-contained unit, are portable among different applications, domains, and knowledge representation languages.

Because it is so plausible to assume there is a clearly defined "division of labor" among modules in the understanding part of the system, it is natural to wonder whether a similar modularization could exist on the generation side. Such a division of labor has been referred to as a distinction between *strategy* and *tactics*, [6,11] which can be very roughly characterized as a distinction between "deciding what to say" and "deciding how to say it." This distinction has been adopted in some form in nearly every language generation system built to date (Appelt [1] and Danlos [2] are among the few to publish objections) although, as might be expected, different researchers have drawn the boundary in different ways.

In a bidirectional system, the obvious choice for a strategic/tactical modularization is at the point indicated in Figure 1. The strategic component of the system is the part that produces a logical form plus a set of functional features, while the tactical component realizes the strategic specification as an utterance. The implication of drawing the line as suggested is that there are such significant differences on either side of the line between the

respective processes and the information they need to access that it makes sense to modularize the system in this manner. By symmetry with understanding, such a modularization is reasonable if, as in understanding, the strategic component need not be concerned with the specific details of grammar rules and, moreover, the tactical component does not have to perform general reasoning.

3 The Problem Posed by Strategic/Tactical Modularization

Shieber[2] has observed a serious problem that arises as a result of the disparate treatment of logical forms by the strategic and tactical modules, which I shall refer to as the *problem of logical-form equivalence.* As far as the strategic component is concerned, logical forms are constructed because of their *meaning.* This does not mean that the strategic process is as simple as figuring out what propositions the hearer needs to know, then using those propositions as logical forms for utterances. In the KAMP system [1], for example, high-level actions were planned to satisfy requirements about the mental state of the hearer, but those specifications were refined into *surface speech-acts.* The propositional content of the surface speech act serves as the logical form of the utterance finally produced. A good deal of reasoning is involved in the expansion of a plan to perform illocutionary acts into a plan incorporating particular surface speech acts. In fact, there is no one-to-one correspondence between illocutionary acts and the surface speech acts that realize them.

If detailed knowledge of grammar rules is to be avoided by the strategic component, the logical forms of surface speech acts must be planned because of their *meaning.* Any equivalent logical form is as good as the one actually chosen as long as it means the same thing. However, to a tactical generation component (as well as a parser), the logical form is an object that is important primarily because its syntax is related to an utterance in a certain way. Just as the logical form doesn't actually *mean* anything to a parser, it doesn't *mean* anything to the tactical generation component in this typical bidirectional system.

To see why this is a problem, consider a task from a domain of circuit repairs in which the speaker (the system) wants to inform the hearer that a particular resistor has a resistance of 500 ohms. The strategic planner may decide that its goal would be satisfied if it uttered a declarative sentence with the propositional content

$$\text{Resistance-of}(\text{R1}, \text{ohm}(500)). \tag{1}$$

If the grammar is constructed properly, this logical form might result in the production of the sentence "The resistance of R1 is 500 ohms." However, it is unlikely that this statement would be specified by a general grammar of English as the logical form for the utterance,

[2]Personal communication.

because its constituents bear no simple relationship to the constituents of any sentence. It is much more likely that the following statement would be the desired logical form:

$$\iota x \text{ Resistance-of}(\text{R1}, x)(x = \text{ohm}(500)). \tag{2}$$

Logical form (2) is more suitable as a representation of the intended utterance than (1) because there is a more natural mapping from constituents of the logical form to constituents of the sentence. It introduces the equality predicate, corresponding to the verb *be,* and the subject and predicate noun phrases correspond directly to arguments to the equality predicate.

But here is the problem: how can a procedure that cares only about the *meaning* of the logical form decide to produce (2) rather than (1)? Or how is it to avoid producing (3)?

$$\iota x \text{ Resistance-of}(\text{R1}, x)(\text{ohm}(500) = x). \tag{3}$$

Commutativity of equality guarantees the logical equivalence of (2) and (3), but (3) is likely to produce the sentence "Five hundred ohms is the resistance of R1." If the functional features state that the resistance is the topic of the sentence, then that plus (3) constitues an inconsistent specification; consequently no output will be produced at all.

Because the syntax of the logical form is significant. as well as its meaning, knowledge of what constitutes a legitimate logical form must be incorporated into the module that produces logical forms. Because the determination of which of several possible equivalent variations of a logical form actually corresponds to an utterance depends on the details of the grammar, the surface speech act planner must have detailed knowledge of the grammar — thus rendering meaningless the symmetric strategic/tactical modularization suggested above. The only other alternative would be to have the tactical generation component produce logically equivalent variations on the logical form until one is found that succeeds. There are two problems with this approach: (1) there are a great many possibilities for generating equivalent expressions, and (2) it may be possible to propose logical forms that, while logically equivalent to the intended utterance, are quite inappropriate. For example, the sentence "The resistance of R1 is 500 ohms and Bruce Springsteen is the Boss or Bruce Springsteen is not the Boss," is not ruled out in principle.

Obviously, a number of language generation systems have been developed that do not seem to suffer from this problem, so there must be a way out. If you examine a collection of better-known generation systems (e.g. KAMP [1], TEXT [6], MUMBLE [5], NIGEL/PENMAN [3]) you will see that, in spite of vast differences in general approach, coverage, application domain, grammar representation, and system interface, there is one very striking similarity: *none of the grammars employed by these systems has an explicitly represented formal semantics.* In theory, KAMP (to choose the example with which the author is most familiar) plans a surface speech act whose propositional content is intended

as the logical form of the utterance produced. This is really in the nature of a white lie: the actual situation is that the logical form of the utterance is *something logically equivalent to* the propostional content of the surface speech act. There is a procedure that uses the propositional content of the surface speech act as a specification to create an initial feature structure for the unification grammar. Knowledge about the way feature structures relate to the logical form (i.e. the semantics of the grammar) is embedded in this procedure. Although the details differ in each case, an analogous story can be told for each of the generation systems under consideration. There is no problem of logical-form equivalence because the logical form of the utterance plays no direct role in the generation process for any of these systems.

Of course, this procedural embedding of semantics is unsuitable for bidirectional systems. Naturally, none of the authors of the generation systems have made any positive claims about the suitability of their grammars for understanding. In fact, MUMBLE [4], unlike the others, does not even represent its grammar as a set of rules that one can consider in isolation from the rest of the generation system. For those systems with explicit grammars, it may be possible to integrate an explicit formal semantics into the grammar and use it for understanding, but as long as a different procedurally embedded semantics is being used for generation, the grammar cannot be considered bidirectional.

At this time it is not clear what would be the best solution to the problem of logical-form equivalence for bidirectional systems, but there are several approaches that may prove fruitful. One approach is to allow the tactical component to substitute equivalent logical forms whenever it is necessary to produce a sentence, but to restrict the types of inferences that can be drawn. For example, if we assume that a PATR-II grammar is used by the parser and generator, allowing the unification algorithm to assume that equality and logical connectives in logical forms are associative and commutative is one way of making it possible for a limited class of inferences to be drawn during the tactical generation process. Whatever solution is ultimately adopted, it is our belief that the advantages inherent in bidirectional systems are sufficient to warrant a close examination of the problems entailed in a bidirectional design.

Acknowledgements

This research was sponsored by the Nippon Telegraph and Telephone Corporation under a contract with SRI International. The author is grateful for comments by Phil Cohen on an earlier draft of this article.

References

[1] Douglas E. Appelt. *Planning English Sentences.* Cambridge University Press, Cambridge, England, 1985.

[2] Laurence Danlos. Conceptual and linguistic decisions in generation. In *Proceedings of the Tenth International Conference on Computational Linguistics*, pages 501–504, 1984.

[3] William C. Mann. An overview of the PENMAN text generation system. In *Proceedings of the National Conference*, pages 261–265, American Association for Artificial Intelligence, 1983.

[4] David D. McDonald. *Natural Language Generation: Complexities and Techniques.* Counselor Project Tecnical Memo 14, University of Massachusetts, 1986.

[5] David D. McDonald. *Natural Language Generation as a Process of Decision Making under Constraint.* PhD thesis, Massachusetts Institute of Technology, 1980.

[6] Kathleen McKeown. *Text Generation.* Cambridge University Press, Cambridge, England, 1985.

[7] Fernando Pereira and Stuart Shieber. The semantics of grammar formalisms seen as computer languages. In *Proceedings of the Tenth International Conference on Computational Linguistics*, pages 123–129, 1984.

[8] Jane Robinson. DIAGRAM: a grammar for dialogues. *Communications of the ACM*, 25(1):27–47, 1982.

[9] Stuart Shieber. *An Introduction to Unification-Based Approaches to Grammar.* Lecture Note Series Vol. 4, Center For the Study of Language and Information, Stanford University, 1986.

[10] Stuart Shieber. A simple reconstruction of GPSG. In *Proceedings of the Eleventh International Conference on Computational Linguistics*, pages 211–215, 1986.

[11] Henry Thompson. Strategy and tactics: a model for language production. In *Papers from the Thirteenth Regional Meeting*, Chicago Linguistics Society, 1977.

[12] Hans Uszkoreit. Categorial unification grammars. In *Proceedings of the Eleventh International Conference on Computational Linguistics*, pages 187–194, 1986.