CANONICAL REPRESENTATION IN NLP SYSTEM DESIGN: A CRITICAL EVALUATION

Kent Wittenburg and Jim Barnett
MCC
3500 West Balcones Center Drive
Austin, TX 78759

ABSTRACT

This paper is a critical evaluation of an approach to control in natural language processing systems which makes use of canonical structures as a way of collapsing multiple analyses in individual components. We give an overview here of how the Lucy natural language interface system currently realizes this control model and then evaluate what we take to be the strengths and weaknesses of such an approach. In particular, we conclude that the use of canonical structures can restrain combinatorial explosion in the search, but at the cost of breaking down the barriers between modules and of letting processing concerns infect the declarative representation of information.

1 Introduction

The traditional design for natural language processing systems is one in which processing proceeds a sentence at a time, with syntactic analysis feeding subsequent semantic and discourse analysis in a "conduit" fashion, to borrow a characterization used in a somewhat different setting by Appelt (1982). The basic advantages of this design stem from the fact that it is inherently modular: control is simple, modules can be developed and debugged independently. The main disadvantage from the processing point of view is that the search can explode as each module detects ambiguities that cannot be resolved until later.[1][2] Although well-known alternatives to conduit models exist -- the most obvious being the proposal to interleave syntax, semantics, and discourse processing -- we sense that the simplicity and modularity of some form of the conduit model continue to be the determining design factor in most applied natural language systems to date, at least for those that can be said to have independent modules to start with. In this paper we will discuss the pros and cons of a design paradigm that stays within the basic conduit model. It is a paradigm characterized by the attempt to procrastinate the resolution of ambiguity by means of representing families of analyses with canonical representations.

The discussion will appear as follows. In Section 2 we attempt to define and justify the use of canonical representations, highlighting their appeal from a processing point of view. We then give a brief overview of how this paradigm has been applied in the natural language interface prototype called Lucy (Rich et al. 1987). Our main points appear in Section 4, where we assess the consequences of these design decisions for three of the Lucy system modules. The conclusion attempts to generalize from this experience. Our main purpose here is thus to evaluate this general design paradigm by presenting a case history of that design applied in a particular project.

[1] For example, Martin, Church, and Patil (1981) mention that their phrase structure grammar produced 958 parses for the naturally occurring sentence *In as much as allocating costs is a tough job I would like to have the total costs related to each product.*

[2] Appelt (1982) mentions other problems with the conduit model having to do with its inability to account for interactions, say, between linguistic choices and gestural ones in language generation models.

2 Canonical representations

A frequent response to the problem of an explosion of syntactic parses in natural language systems is to have the parsing module assign *canonical* representations to families of structures. Church (1980), Martin, Church, and Patil (1981), Marcus, Hindle, and Fleck (1983), Pereira (1983), Pulman (1985), and Wittenburg (1987) have all advocated some form of this idea, which has sometimes gone under the name of pseudo-attachment. These canonical structures are unambiguous from the point of view of the parser/grammar, but have several different semantic translations when it comes to intepretation. The advantage of this approach is that the semantics module might be able to choose quickly between the multiple *translations*, even though the syntax could not choose between the *parses*. For example, instead of enumerating all possible prepositional phrase attachments, the grammar could force a consistent attachment (either high or low or perhaps a flat n-ary branching tree) and return only a single parse for strings of multiple PPs. Semantic processing could then expand the canonical structure and consider the alternatives when it had the information necessary to choose among them. Information that could help in this delayed choice would be semantic translations of the nouns and verbs that carry constraints on their possible modifiers. We will take such examples in syntax and semantics as paragon cases of the general design strategy that concerns us.

Figures 1 and 2 present the paradigm in a more schematic way. Figure 1 shows a search space that branches three ways at the first two depths and two ways at depth three. Imagine that that the search in Module 1 represents the parsing of a particular sentence where two structures are three-ways ambiguous, a third is two-ways ambiguous, and a parse exists for each combination of the three. A semantics component that took over in Module 2 would be faced with translating an exponentially growing number of parses, in this case 18. Underscores in Module 2 are intended to represent ill-formedness from the perspective of this component; thus Figure 1 indicates that only one of the 18 inputs to Module 2 passes muster, i.e., only one of the parses is well-formed from a semantic point of view.

If the grammar were changed by finding a single canonical representation for each of the structures that are ambiguous in Figure 1, the search tree for the parsing of this same sentence would be as shown in Figure 2.

Then, as the semantics module takes over, it would begin enumerating the alternatives that each of these canonical syntactic structures actually represents. As we indicate in Figure 2, we assume that there is sufficient (semantic) information available to immediately rule out unproductive branches. In the ideal case, the combinatorics of Figure 1 may be completely circumvented, leaving the basic flow of control intact.

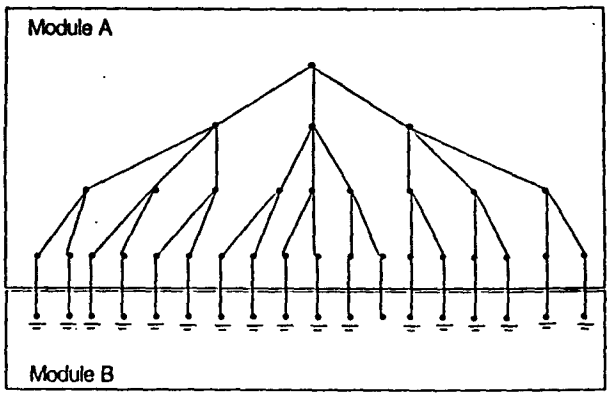


Figure 1: Search in a conduit model

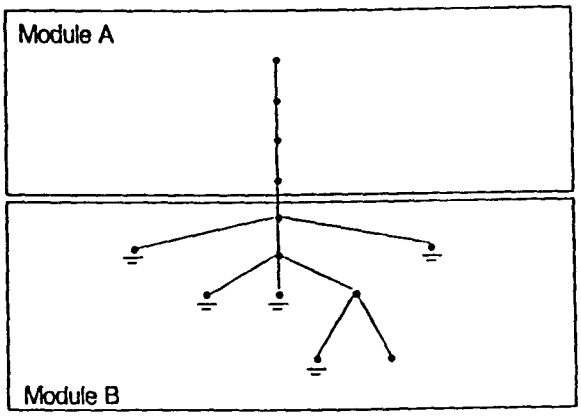


Figure 2: Search in a canonical model

Note, however, that the canonical representation paradigm as we have presented it does not reduce the size of the overall search in the worst case. The canonicalized nodes in Module 1 of Figure 2 still have to be expanded in Module 2 -- the expansion has merely been delayed. But of course semantic information still may not be able to rule out the choices, and if not, the same combinatorics in Module 1 of Figure 1 would appear in Module 2 of Figure 2. Any gain in efficiency will come solely from being able to prune the search tree quickly because of the presence of information that would not have been available at an earlier stage. The success of canonical strategies will thus be determined by the extent to which it is possible to 1) find well-motivated representations that allow painless recovery of the alternatives, 2) choose the correct points to unpack the structures, and 3) do 1 and 2 without undue cost to the rest of the system.

3 Canonical structures in Lucy

Lucy is a natural language interface prototype that has been built by the Lingo group at MCC (Rich et al. 1987). One of the aims has been to design an interface system that is portable across applications, and thus strong modularity has been one of the central design factors. Figure 3 shows the basic system design; it is a classic conduit model where control passes from syntax to semantics and thence to discourse and pragmatics.

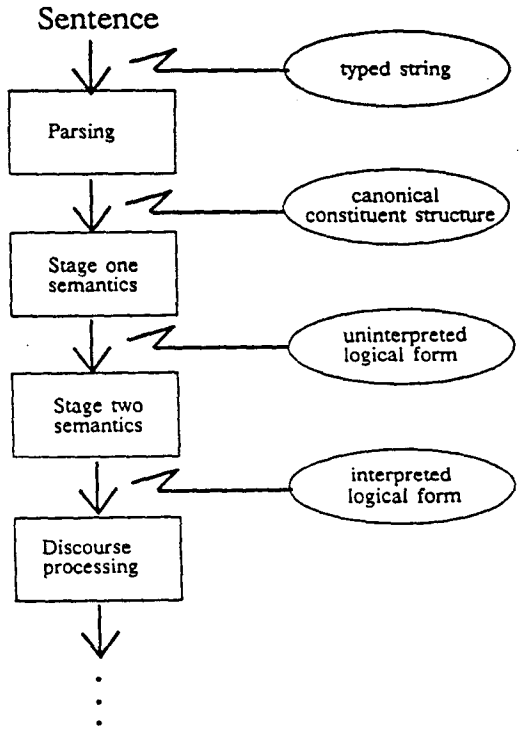


Figure 3: The Lucy system

In many cases Lucy's parser produces a single parse for an input sentence. The resulting structural description is then unpacked and disambiguated in the semantics module. As Figure 3 shows, semantic processing in Lucy procedes in two stages. In Stage 1, the semantic processing module rewrites the parse output as a set of logical assertions. The predicates in these assertions are English words, taken from the parse tree, so that the output of this initial stage can be considered to represent the uninterpreted predicational structure of the sentence, which abstracts away from the meaning of in-

dividual words.[3] In Stage 2 Lucy uses a set of semantic mapping rules to translate the Stage 1 assertions into the vocabulary of a knowledge base, and then considers the various interpretations, letting through only those that are semantically consistent, where consistency is defined in terms of the knowledge base's class hierarchy. (This amounts to checking for semantic subcategorization restrictions.) This interpreted logical form is suitable input for discourse and pragmatic processing, and, ultimately, for the backend program.

The Lucy system uses canonical structures to deal with the following types of ambiguity: semantic sense ambiguity, idiom recognition, noun-noun bracketing, prepositional phrase attachment, and quantifier scope assignment. We summarize a few of these treatments here. See Rich et al. (1987) for more detail.

Lucy assigns the same syntactic analysis to the literal and idiomatic readings of a sentence.[4] Thus the parser produces a canonical representation for idioms that amounts to a full structural description for the literal reading of the sentence. Then the Stage 1 procedure, which rewrites the parser output into logical assertions, uses an idiom dictionary to produce separate sets of assertions for the idiomatic and literal readings. Note that in this case the canonical structure must be expanded quite soon. This is because it is impossible to begin translating assertions into the language of the knowledge base without knowing whether the translation is to be literal and compositional or idiomatic and global. Furthermore, producing a logical form involves making a commitment about how many objects we are talking about, and the idiomatic and literal readings may imply the existence of different numbers of referents.[5] Thus, though idioms can pass through the syntax untouched, they require an early commitment in semantics.

In the case of noun compounds, the parser assigns a canonical right-branching structure which Stage 1 processing rewrites into a flat list of nouns. Stage 2 processing is then free to assign to the compound any bracketing for which it can find an interpretation. The semantic mapping rules contain compounding entries for nouns, allowing separate specifications for the semantics of a noun as a head of a compound and as a modifier. It is also possible for an entry to specify that the "semantic head" of a compound should be flipped (e.g., in the case of "a stone lion", which is a stone and not a lion.) In this case, the canonical structure does not have to be unpacked until a translation for the constituent is required.

In the case of prepositions, the parser attaches them at the highest point in the tree with an indication of their domain, i.e., the subtree within which they can be attached. The high attachment is not altered in Stage 1. In Stage 2, after the nouns and verbs have been translated, Lucy attempts to attach prepositional phrases and other postmodifiers, checking to see which translations are consistent with which attachments. For example, in "I saw the boy on Monday," the parser would attach "on Monday" high, assigning a structure indicating that both "saw" and "boy" were possible attachment sites. The lexical entry for "on" would state that "(on x y)" can mean "(temporally-located-in x y)" if x is an event and y a day. Lucy would then accept "saw-on" as a reading, but not "boy-on" (assuming there is no entry giving a reading for "(on x y)" where x is a person and y a date.) In postponing PP attachment until the end of the semantic translation routine, Lucy assumes that 1) the translations of the nouns and verbs are more likely to constrain the readings and attachments of the prepositions than vice-versa, and 2) that the resulting translation can be built up piecemeal, with the translations of the PPs "added in" to the translations of the nouns and verbs. One result of this strategy is that verb/particle and verb/preposition compounds must be treated as idioms, since in these cases the meaning is not cumulative. (It would be hard to assign independently motivated meanings to "look" and "up" that would combine to give the meaning "look up" in "I looked the word up.")

Finally, Lucy, like most other systems, does not assign quantifier scope either in the parse tree or in the first stages of semantic processing; scope assignment is postponed until the Stage 2 translation into the language of the knowledge base is completed.[6]

4 Consequences for the modules

The Lucy experiment has shown that it is possible to push the technique of canonical representations quite far indeed, thus maintaining the overall simplicity of a conduit control model with a sentence as the basic unit of data. However, the consequences for the knowledge sources involved within each of the modules have been far-reaching. We next review some of those consequences for the grammar, for the syntax-semantics relations, and for those parts of semantics proper having to do with sortal consistency of terms in the knowledge base.

4.1 The grammar

For each of the phenomena discussed in the previous sec-

[3] The design of this level of Lucy is influenced by Hobbs (1985), which advocates a level of "surfacy" logical form with predicates close to actual English words and a structure similar to the syntactic structure of the sentence.

[4] At present, Lucy can treat strings of adjectives and nouns as idioms, as well as verb/particle and verb/preposition compounds. We've done experimental work that indicates that there is no problem in extending this approach to handle full VP idioms, such as "kick the bucket," but this functionality is not yet part of the system.

[5] Lucy's logical form incorporates the notion of a discourse referent (see Kamp (1984), Heim (1982)), and the creation of a discourse referent implies the possibility of anaphoric reference (within the range of accessibility of the referent.) Thus, when a noun phrase "a bucket" or "the bucket" occurs, we normally can refer back to it with "it"; however, if we use the idiom "kick the bucket" to mean "die", no such anaphora is possible. Hence idioms must be detected before discourse referents are created. As noted in footnote above, Lucy does not yet deal with full VP idioms like "kick the bucket," but awareness of the effect such idioms would have on our discourse processing strategy is an additional argument for locating the idiom module relatively early in post-syntactic processing.

[6] At present, Lucy uses no knowledge except that contained in the class hierarchy. Such information is not useful for determining quantifier scope, so Lucy gives a default left-right assignment.

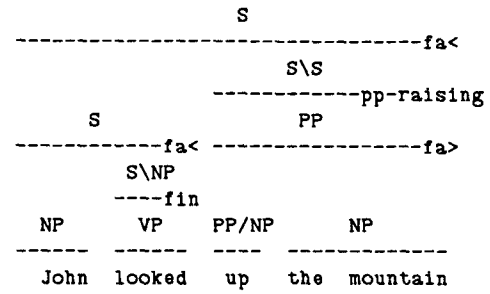
tion, the Lucy grammar (i.e., its syntactic lexicon and rule base) was hand-tooled to pack a number of analyses into a single canonical parse. The goal we had been aiming for, in fact, was to return only a single parse for any given input sentence. In some cases, the effects on the grammar were relatively minor. Forcing high attachment of PPs, for example, involved a slight augmentation of syntactic feature structures in the categories and rules such that low attachments led to feature clashes when the parser tried to incorporate such modified constituents into, say, a higher verb phrase. Forcing right-branching analyses of noun-noun compounds was comparable. However, where there were interactions involving lexical ambiguity, the canonicalization of the grammar had far more radical effects. Interactions among subcategorization of potential phrasal verb heads and ambiguity between prepositions and particles provides one telling example. We give a brief history here of this case in order to illustrate the kind of effects on declarative information that canonicalization can lead to.

We began with the goal of finding a canonical form to conflate structures of the following sort since syntax alone would have insufficient information to force a choice between them:

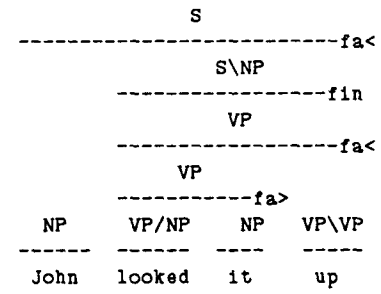
John looked [up the mountain]
John [looked up] the mountain

What should the canonical form be in such a case? One could either analyze such sentences as an intransitive verb followed by a PP or as a transitive verb + particle combination. We chose the former since the PP reading seems to be the one more generally available, whereas the presence of a particle reading depended on there being an entry in our idiom dictionary, which in Lucy is accessed only after the parse is complete. The semantic mapping rules we produced then had to create two logical forms from the one canonical analysis; the first corresponded directly to the PP structure, the other to the phrasal verb structure, even though that latter structure as such was not present in the syntax.

We explored several options for writing a grammar that would produce a PP bracketing, and only this bracketing, in such cases. The one we settled on led us to derivation trees like the following:[7]



In order to be sure that this was the ONLY parse given by our grammar in such cases, we had to be sure that there was no particle analysis for this same sentence. However, we did of course have to allow particle-type bracketings when no prepositional-type bracketing was available as, for instance, in sentences like "John looked it up" or "John caught up". This we did by having prepositions, not verbs, always take the NP as an argument if there was a preposition/particle intervening between the verb and the NP and by having verbs take the NP as an argument if there were no preposition/particle intervening. Particles then took a complete VP as a left argument. An input sentence such as "John looked it up" thus produced the following unique derivation, which was interpreted with the particle reading only.



Now our analysis was complete. Our goals were achieved. But consider what the effects were on the grammar. In order to get an analysis for "John looked it up" we had to assign a transitive verb entry to "look", even though it was really only the two-word entry "look up" that was transitive, not "look" itself. In order to get an analysis for sentences like "John caught up" we had to assign an intransitive entry to "caught", even though "caught up", not "caught", was the actual intransitive form. Also, the analysis of verb particles failed to reflect the fact that English particles do appear between verbs and verb objects--in this grammar particles were specifically excluded from this position in order to avoid particle/preposition ambiguities. So our entire motivation for grammatical analyses was now being driven by the need to stamp out alternative derivations and no longer by principled linguistic concerns. The casualties to the grammar included principled assignments of categories to words in the lexicon, principled definitions of categories themselves, and principled connections between syntactic structures and the interpretations they were capable of producing.

Our example illustrates how far things may go. This is not to say, however, that any form of canonicalization invariably has such devastating consequences for the grammar. The effects of canonicalization of PP modifier attachments seem to be relatively minor, for instance. As an anonymous reviewer stated so clearly, whether canonicalization is likely to work or not depends on the locality of the phenomenon the canonicalization is attempting to account for. The less any other grammatical processes are sensitive to the inter-

[7] Lucy uses a form of categorial grammar in its syntactic component. *Fa<* and *fa>* stand for backward function application and forward function application, respectively. Function application is the basic binary reduction rule in the grammar. It applies a functor category, such as a verb which is looking for some argument, to a category that can satisfy an argument role. *Pp-raising* is a unary rule that makes S-modifiers out of basic PPs and *fin* is the unary rule that lifts VPs to the category for finite verbs, adding the subject argument. See Wittenburg (1986) for details of Lucy style grammars, Uszkoreit (1986), Karttunen (1987), and Zeevat, Klein, and Calder (1987) for related versions of Categorial Unification Grammars.

nals of a canonical representation, the better the prospects for success. However, there are surprisingly few cases where no other grammatical processes are affected. This same reviewer mentioned an interesting example involving noun-noun compounds. Structural ambiguity within noun-noun compounds might seem to be one of the most promising cases for canonicalization in English given that most grammatical processes are not sensitive to the internal structure of NPs. However, when the grammar includes generalized conjunction, problems quickly surface. Consider an ambiguous sequence such as "N1 and N2 N3 V". In order to encompass such examples, the canonicalization of compounds presumably needs to be extended so that only one of the two obvious analyses will be parsed. But subject-verb agreement will be affected by the choice of structure, and it seems difficult to see how any straightforward solution could account for all cases of agreement and still return only a single parse. Thus we see that the internal structure of the NP does matter after all, since conjunction and percolation of agreement features are affected. Attempts to extend canonicalization to cases in which even the most basic constituency is undetermined seems even less likely to succeed. Examples such as "look up the word" along with others such as "I want the chicken to have lunch" share an uncertainty about what the basic constituents in question really are.

4.2 Stage 1 semantics

The main consequence of canonicalization for Stage 1 semantic processing, which corresponds to the semantic translation step, is an increase in complexity. In particular, the domain of locality for translations from syntactic structures to semantic forms is affected. An immediate consequence is that the mapping from parse structures to logical assertions is less transparent than that in approaches that maintain a homomorphism between syntax and semantics such as Montague grammars and related phrase structure frameworks (e.g., Klein and Sag 1985). For canonicalized structures, the syntax-semantics mapping cannot take place in a local, compositional manner. We discuss PPs as an example.

First, consider canonicalization of prepositional phrases in their role as modifiers. In Lucy the syntax attaches PPs high, and Stage 1 processing produces special "Attach" assertions that are interpreted in such a way as to ultimately produce the set of possible attachments. Thus in the example "I saw the man on a hill with a telescope" shown below, the syntax results in a representation indicating modifiers and their attachment dom(ains). Stage 1 semantics processing produces several basic assertions as well as one "Attach" assertion whose arguments consist of a list of (referents of) potential attachment sites followed by a sequence of prepositional phrases that are to be attached.

```
Syntax:
   [mod: [prep: with
          pobj: a telescope]
    dom: [mod: [prep: on
                pobj: a hill]
          dom: [subj: I
                pred: [verb: saw
                       obj: the man]]]]


Stage 1 semantics:
   (I x1)
   (man x2)
   (see e1 x1 x2)
   (hill x3)
   (telescope x4)
   (Attach (e1 x2)(on arg1 x3)(with arg1 x4))
```

Note that the structure of the attachment assertion bears no simple relation to the structure of the syntactic analysis. Producing the semantics assertions entails conducting a search on the attachment domain, pulling out relevant subparts, and reassembling them into a different form. The translation process here is thus no longer a simple function of the translation of the PP and the translation of the constituent that the PP attaches to.

When the canonicalization includes not only the collapsing of attachment sites for PP modifiers but also structures involving two-word verbs as discussed above, the complexity of the translation step goes up again. In the case of "look up the word", there is only one prepositional phrase to attach and only one place to attach it, but processing is complicated by the fact that we must check for the particle reading of "look up." Where there is such a reading, we must generate a separate translation, with a branch in the subsequent search, even though the verb and the preposition are not parts of a single constituent, either in the syntax or in the rest of Stage 1 semantics. The meaning of the whole sentence thus contains readings that are not (simple) functions of the meaning of the constituents in the parse tree. A similar problem would take place wherever prepositional phrases could be taken as arguments to a verb rather than as modifiers of it. (See the discussion below of indirect objects with "to" and "for".)

Additional complications for PPs arise in sentences with "be" and a prepositional phrase. The natural semantics for "John is next to Mary" would have "next to Mary" either as a predicate of "John" or as an argument to "be." In these cases, the Lucy grammar still attaches the PP high to the pseudo-constituent "John is". The Stage 1 routine then has to detach the (first) PP from its position high in the tree and move it down into the VP. The resulting translation can be derived compositionally from the transformed parse tree, but not from the original one. Thus, even in the seemingly straightforward case of prepositional phrases, the relation between syntax and semantics has become opaque, with the readings often differing significantly from the "natural" interpretations of the parse tree. One concrete result of this complexity is that the Stage 1 routine in Lucy is procedurally rather than declaratively stated. It is not a particularly troublesome routine, but the complicated conditionalized transformations it performs would be hard to express declaratively.

4.3 Stage 2 semantics

Stage 2 semantics in Lucy represents the transition from surface linguistic structure to a deeper, knowledge-based

form of representation. In syntax and in Stage 1 semantic representation the lexical items are English words. During Stage 2 processing these are are translated into the predicates of a domain knowledge base. Thus, by the time Stage 2 processing is finished, all information about the surface linguistic form is gone. However, as a result of canonicalization, the Stage 2 semantic module ends up doing (explicitly or implicitly) the syntactic processing that has been put off by earlier components. Since the output of Stage 2 semantics is supposed to represent the meaning of the sentence, modifier attachment must be resolved. Consider the case of PP attachment again. The part of the module that determines attachment must know that crossed branches are not allowed; that is, in a string like "I saw a man on the hill with a telescope", if "on a hill" modifies "saw" then "with a telescope" cannot modify "a man." Thus, the Stage 2 component must keep track of the interactions of the different proposed attachments, and this involves knowledge of the syntactic tree structure. Thus, information that properly belongs in the syntactic module ends up being duplicated in the semantics. Furthermore, if other modules, e.g. discourse, need detailed syntactic information, the semantics component will have to go back and update the syntactic structure to reflect the ultimate attachment of the PPs.

In some cases, lexical information may also have to be passed along fairly far into Stage 2 semantics. Consider the case of the delayed attachment of a PP that might be a semantic indirect object ("I sent a letter to Mary" in the sense equivalent to "I sent Mary a letter.") The problem here is that some verbs ("send", "give", etc.) take "to" as an indirect object marker, while a smaller class of verbs ("buy", "find", etc.) take "for" as a marker. The module will need to know what the surface verb was to make the attachment properly (in order to avoid interpreting "for John" as the recipient in "I sent it for John", etc.) In general, attachment is often sensitive to the lexical items involved, and delaying attachment decisions entails importing surface lexical, as well as syntactic, information into a part of the system that is more naturally thought of as operating on "pure meaning" plus world knowledge. In short, upstream syntactic information is contaminating downstream semantic processing.

Finally, even if we are willing to accept such distortions in the semantics, there are cases involving "of" where late attachment seems to be impossible. Normally, a phrase of the form NP1 Prep NP2 denotes a subset of the denotation of NP1 (e.g., a man in a sweater is a man and not a sweater.) However, "a bottle of beer" is often taken to denote the beer, rather than the bottle. For example, you can pour, drink, or dilute a bottle of beer, though you can do none of these things to a simple glass bottle. Therefore, if semantic processing involves checking for sortal consistency (subcategorization), as Stage 2 semantics in Lucy does, either PPs with "of" will have to be attached *before* verb/argument pairs are checked for consistency, or semantics will reject sentences that in fact have good readings. For example, if "drink" subcategorizes for a liquid as its direct object, and "a bottle" denotes a piece of glass (of the right size and shape, etc.), then "drink a bottle" will fail sortal consistency checking, even though "drink a bottle of beer" would succeed. We could say that "bottle" also denotes a certain quantity of liquid, but by doing so we introduce artificial ambiguity into the unambiguous sentence "I found a bottle on the beach" (since one could certainly find a quantity of liquid on a beach).[8] The best solution would be to treat "of" separately from other prepositions, determining attachment earlier in the processing.[9] However, the added complication that such treatment would entail reinforces the point that, even in cases where canonicalization seems innocuous to the syntax, the side-effects on semantic processing can be significant.

Reflecting on the effect of canonicalization on semantic processing, we see that, as remarked above in the discussion of syntax, the locality of the construction in question is an important factor. In the case of noun-noun compounding, it happens that there are few interactions between the internal structure of the canonicalized construction and the rest of the sentence. Accordingly, canonicalization of these structures provides a painless way of avoiding early branching in the search. Prepositional phrases, however, although they show a high degree of locality in the syntax, are involved in complex, non-local interactions in the semantics, with a corresponding complication of the processing. In such cases, canonicalization can still be made to work, but only at a price.

5 Conclusion

We believe that the Lucy experiment with canonical representations has generally succeeded in lowering the amount of effort Lucy spends on search. The parser usually returns a single analysis, instead of many, and the semantics module usually succeeds in ruling out most of the possibilities when they are finally unpacked. A further benefit is that debugging some individual modules has been made easier. We have found, in particular, that debugging a grammar that typically produces only one or a very small number of parses is much easier than when the grammar returns, say, hundreds of parses for a given sentence.

But what of the hidden costs to the system? The course of our research has caused us to step back and question the whole idea of canonical structures for two primary reasons: first, canonical structures tend to let declarative information be far too influenced by processing concerns; second, modules leak in such designs, essentially doing away with one of the main arguments for such control models in the first place.

There are rather serious practical, as well as theoretical, consequences when canonical forms make their way into the grammar in the way discussed in Section 4.1. First is the problem of lexical acquisition when lexical category assignments become so off-beat. Second, with arcane relations between syntactic output and semantic result as discussed in Section 4.2, it becomes difficult to see how such systems could be easily used for other purposes than the specific ones they have been written for. For instance, it is hard to see how multilingual systems could relate grammars when individual grammars have been so heavily influenced by the accidental vagaries of processing concerns in that language. It is also hard to see how a generation system could easily make use of such grammars, since the mapping rules will tend to be complicated and fundamentally unidirectional.

The moral to be drawn from the remarks in Section 4.3 seems to be that a canonical structure model, at least in its extreme form, does not permit us to maintain the modularity of a traditional conduit model. If we return to Figure 2 above, it is clear that when we finally begin enumerating the branching that has simply been delayed in the canonical out-

[8] Furthermore, almost any physical object can serve as a container: "We had lunch at the dump. I drank a hubcap of beer and ate a distributor cap of pate."

[9] There is some evidence for treating "of" as a member of a distinct syntactic class. For one thing, "of", unlike other prepositions, cannot attach to sentences (though it can mark an argument of the verb: "the time has come, the Walrus said, to talk of many things...")

put of module A, we will still have to use the information that fundamentally belongs in module A, even though we are doing this processing in module B. The effect is that we will require passing along information from box to box. Thus, we end up doing interleaving whether we want to or not.

Although these conclusions seem to be damning for the general design philosophy, we should note that our attempts at evaluation here are open to the criticism that a single case history does not necessarily justify general conclusions about a design philosophy. There is always the possibility that the design wasn't applied "right" in the case at hand. In particular, we should distinguish the proposal for hand-tooling canonical representations into a grammar as we have done in Lucy from the proposal for automatically inferring higher level generalizations from modules that themselves have still been driven by principled linguistic concerns. The proposals of Church and Patil (1982) fall more into this latter camp, and it is a goal of the ongoing redesign efforts in Lucy to incorporate some version of automatic generalization.

Despite the negatives, it is possible that for some NLP applications the balance could still tip in favor of using canonical representations for some limited set of structures such as noun compounding or PP modifier attachment. Applications that have no pretensions of being fully general or easily extensible may be willing to pay the price that canonicalization exacts in order to avoid a more complex design and still achieve acceptable performance results. In fact, we expect that the need for methods that incorporate some form of delayed evaluation will continue to be pressing in natural language analysis, and in view of the short supply of such methods currently available, canonicalization may continue to have its place in the near term. However, our conclusion after two years of pursuing such techniques is that conduit control models using canonical structures ultimately offer no real alternative to more complex designs in which control is interleaved among modules.

6 Acknowledgements

This paper reports on work undertaken by the Lingo project at MCC in 1986 and 1987. Other members of Lingo connected with this work include Elaine Rich, Jon Schlossberg, Kelly Shuldberg, Carl Weir, Greg Whittemore, and Dave Wroblewski. Elaine Rich, in particular, has contributed much to the debates over issues discussed here and has commented on earlier drafts. We'd like to acknowledge Dave Wroblewski's role in these areas also, as well as his essential contributions to implementing Lucy. Finally, the comments of an anonymous reviewer were very useful to us in revising an earlier draft.

REFERENCES

Appelt, D. 1982. Planning Natural-Language Utterances to Satisfy Multiple Goals. Technical report no. 259, A.I. Center, SRI International.

Church, K., and R. Patil. 1982. Coping with Syntactic Ambiguity or How to Put the Block in the Box on the Table. Journal of Computational Linguistics 8:139-149.

Heim, I. 1982. The Semantics of Definite and Indefinite Noun Phrases. Ph.D. dissertation, University of Massachussetts.

Hobbs, J. 1985. Ontological Promiscuity. In Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, pp. 61-69.

Kamp. H. 1984. A Theory of Truth and Semantic Representation. In Groenendijk et al. (eds), Truth, Interpretation, and Information, pp. 1-41. Foris.

Karttunen, L. 1987. Radical Lexicalism. To appear in M. Baltin and A. Kroch (eds), New Conceptions of Phrase Structure, MIT Press.

Klein, E., and I. Sag. 1985. Type-driven Translation. Linguistics and Philosophy 8:163-201.

Marcus, M., D. Hindle, and M. Fleck. 1983. D-Theory: Talking about Talking about Trees. In Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics, pp. 129-136.

Martin, W., K. Church, and R. Patil. 1981. Preliminary Analysis of a Breadth-First Parsing Algorithm: Theoretical and Experimental Results, technical report no. MIT/LCS/TR-261, Massachusetts Institute of Technology.

Pereira, F. 1983. Logic for Natural Language Analysis. Technical report no. 275, A.I. Center, SRI International.

Pulman, S. G. 1983. Generalized Phrase Structure Grammar, Earley's Algorithm, and the minimisation of Recursion. In K. Sparck Jones and Y. Wilks (eds), Automatic Natural Language Parsing, pp. 117-131. Halsted.

Rich, E., J. Barnett, K. Wittenburg, and D. Wroblewski. 1987. Ambiguity Procrastination. In Proceedings of AAAI-87, pp. 571-576.

Uszkoreit, H. 1986. Categorial Unification Grammars. In Proceedings of Coling 1986, pp. 187-194.

Wittenburg, K. 1986. Natural Language Parsing with Combinatory Categorial Grammars in a Graph-Unification-Based Formalism. Ph.D. dissertation, University of Texas at Austin.

Wittenburg, K. 1987. Extraposition from NP as Anaphora. In G. Huck and A. Ojeda (eds), Syntax and Semantics, Volume 20: Discontinuous Constituencies, pp. 427-444. Academic.

Zeevat, H., E. Klein, and J. Calder. 1986. Unification Categorial Grammar. In Edinburgh Working Papers in Cognitive Science, Volume 1, Categorial Grammar, Unification Grammar, and Parsing. Centre for Cognitive Science, pp. 195-222. University of Edinburgh.