Understanding Dialogue Ellipsis

Robert E. Frederking

CMU-CMT-93-142

7 November 1993

Abstract

The interpretation of dialogue ellipsis has been the subject of much work in computational linguistics. Approaches to this problem can be divided into ''deep'' systems, which attempt to use discourse structure or reasoning about plans to resolve elliptical utterances, and ''shallow'' systems, which attempt to use the structure of linguistic antecedents to resolve these utterances. Based on insights gained from a taxonomy of dialogue ellipsis, it can be seen that these systems actually attack different classes of ellipsis. With this background, a system, Psli3, is presented that resolves elliptical utterances in database access dialogues. It uses a case frame based technique to resolve all varieties of elliptical utterances that are resolvable with ''shallow'' techniques. This system also demonstrates the capabilities of *semantic chart parsing*. This is an adaptation of chart parsing to a production system architecture, with the chart extended to handle ambiguity in semantic representations and reference relations. The semantic chart technique has been further extended to handle a form of dialogue level ambiguity as well.

1. Introduction

There has been a substantial amount of research on the interpretation of ellipsis in dialogue [Hendrix 77, Burton 75, Waltz 77, Carbonell 83, Cellio 84, Grosz 77, Carberry 89, Litman 87, Allen 80]. Some of the techniques proposed [Grosz 77, Carberry 89, Litman 87, Allen 80] involve large amounts of inference, or the application of much world knowledge. In other words, they are ''deep'' ellipsis resolution techniques. While these techniques promise to *resolve* (derive a full interpretation from) very difficult forms of ellipsis, they are also computationally expensive, even when applied to relatively simple forms. There is another class of techniques [Hendrix 77, Burton 75, Waltz 77, Carbonell 83, Cellio 84] that employ matching between an elliptical utterance and its surface linguistic antecedent. These ''shallow'' techniques clearly cannot resolve the hardest forms of ellipsis, but they are very efficient in resolving the more frequent simple forms. A system is presented here that resolves intersentential ellipsis in dialogue using a case frame based technique. This technique can resolve any elliptical utterance resolvable with ''shallow'' techniques.

This distinction between different forms of elliptical utterance can be made much more precise. Through the study of naturally occurring elliptical utterances, a taxonomy of intersentential ellipsis has been developed. The categories in this taxonomy are based on the presence or absence of a surface linguistic antecedent, and coreference, and on the type of case frame match when there is an antecedent and coreference. This taxonomy is very useful in understanding how different ellipsis resolution techniques are related, and which utterances they apply to.

The system presented here, Psli3 (pronounced ''sly 3''), interprets and, to a limited extent, generates natural language utterances, using the *semantic chart* technique within an OPS5 [Forgy 81, Brownston 85] production system. The working memory of a production system provides exactly the sort of continuous context needed to integrate the dialogue across sentence and process boundaries. The semantic chart technique provides an elegant, general means for handling ambiguity within the production system framework. The flexibility and highly-integrated processing of this system has been demonstrated primarily through its handling of dialogue intersentential ellipsis, although it also addresses a number of other problems, including definite reference interpretation, error correction, and natural language generation [Frederking 88].

Figures 1-1, 1-2, and 1-3 present a sample dialogue handled by Psli3. This example, as well as the other examples in this paper, was produced by a version of Psli3 consisting of 361 OPS5 rules, running in

Common Lisp on a Symbolics 3600 Lisp Machine. The domain of the system is information retrieval from a (sometimes sparse) medical database.

- The first two interactions demonstrate *reformulation* ellipsis, where a new query is formed by modifying an earlier one[1]. The latter of these requires the system to handle the semantics (reference properties, actually) of ''other''.
- The third interaction shows the system failing to find a referent for a definite reference, generating an elliptical question to fix this, and understanding the elliptical answer.
- The fourth contains an elliptical input with a missing constituent instead of the more typical missing superstructure. The sentence could be ambiguous syntactically, but not when semantics are taken into account.
- The last two interactions show the system interpreting utterances that combine a sentence and an elliptical utterance in two different ways: as a clarification of a single request, or as a reformulation producing two requests. The first results in an interpretation with a single underlying action, while in the second there are two underlying actions, because the case frame representations of the utterances cannot be merged. These final two utterances produce semantic charts that span more than one top level input, referred to here as *dialogue charts*.

We begin with a general discussion in section 2 of previous research on ellipsis in natural language interfaces. Section 3 then describes the taxonomy of dialogue ellipsis developed in this work. Section 4 introduces semantic chart parsing, shows in detail how this mechanism can be used to resolve one type of ellipsis from the taxonomy, and describes an initial application of this technique to dialogue level ambiguity. Section 5 concludes the paper with some evaluation and possible directions for future development.

---

[1] Throughout this paper, indentation will be used to show change of speakers. User inputs are shown in bold italics, while system responses are in all capitals.

Show me some ribs.

RIB_NUMBER_7
(IS-A (VALUE TRUE_RIB))

RIB_NUMBER_6
(IS-A (VALUE TRUE_RIB))

RIB_NUMBER_5
(IS-A (VALUE TRUE_RIB))

Some other ribs.

RIB_NUMBER_4
(IS-A (VALUE TRUE_RIB))

RIB_NUMBER_3
(IS-A (VALUE TRUE_RIB))

RIB_NUMBER_2
(IS-A (VALUE TRUE_RIB))

Display the irregular bones.
WHICH IRREGULAR BONES?
Any irregular bones.

ETHMOID
(IS-A (VALUE IRREGULAR_BONE))
(PART-OF (VALUE CRANIUM))

HYOID
(IS-A (VALUE IRREGULAR_BONE))
(BILATERAL (VALUE F))
(LOCATED-IN (VALUE NECK))

INCUS
(IS-A (VALUE IRREGULAR_BONE))
(LOCATED-IN (VALUE HEAD))

Figure 1-1: Example Psli3 dialogue (beginning)

Show Mark.

```
ETHMOID
(IS-A (VALUE IRREGULAR_BONE))
(PART-OF (VALUE CRANIUM))

HYOID
(IS-A (VALUE IRREGULAR_BONE))
(BILATERAL (VALUE F))
(LOCATED-IN (VALUE NECK))

INCUS
(IS-A (VALUE IRREGULAR_BONE))
(LOCATED-IN (VALUE HEAD))
```

Display some sternal ribs. To Phil.

```
RIB_NUMBER_7
(IS-A (VALUE TRUE_RIB))

RIB_NUMBER_6
(IS-A (VALUE TRUE_RIB))

RIB_NUMBER_5
(IS-A (VALUE TRUE_RIB))
```

Figure 1-2: Example Psli3 dialogue (middle)

Display some asternal ribs. Some flat bones.

```
RIB_NUMBER_9
(IS-A (VALUE FALSE_RIB))

RIB_NUMBER_8
(IS-A (VALUE FALSE_RIB))

RIB_NUMBER_10
(IS-A (VALUE FALSE_RIB))


VOMER
(IS-A (VALUE FLAT_BONE))
(PART-OF (VALUE FACE))
(BILATERAL (VALUE T))

STERNUM
(IS-A (VALUE FLAT_BONE))
(BILATERAL (VALUE F))
(LOCATED-IN (VALUE TRUNK))

SCAPULA
(JOINED-BY (VALUE SHOULDER_JOINT))
(IS-A (VALUE FLAT_BONE))
(LOCATED-IN (VALUE SHOULDER TRUNK))
(BILATERAL (VALUE T))
```

Figure 1-3: Example Psli3 dialogue (conclusion)

2. Ellipsis in natural language interfaces

An *elliptical utterance* occurs when a speaker produces an utterance that, in isolation, would be syntactically or semantically incomplete. The hearer is expected to complete the utterance during interpretation, usually using the current conversational context along with general knowledge about language and about the world. For example, a user might type the following to a system for ordering computer components:

> ***Give me a disk with two ports.***
> <System indicates it has done so.>
> ***With one port.***

We will be concerned here only with such *intersentential* ellipsis, where the *antecedent*, or linguistic source for the missing material, is in a preceding sentence, and not with *intrasentential* ellipsis, such as:

> ***Give me a dual port and a single port disk.***

where the antecedent is in the same sentence. Intersentential ellipsis is more frequent in dialogue [Carbonell 83], and does not have such complex syntactic constraints as intrasentential ellipsis. Ellipsis is one of several linguistic devices, including definite reference and deixis, that exploit the coherence of natural language dialogue to allow shorter, simpler utterances. Natural language interfaces need to be able to interpret elliptical utterances, both because they tend to be quite frequent in dialogue and because they allow briefer[2] and more natural interactions.

Previous work on intersentential ellipsis in natural language interfaces has been carried out within a number of paradigms. As stated above, these can be grouped into those which use ''shallow'', utterance-oriented techniques, such as semantic grammars [Hendrix 77, Burton 75, Waltz 77] and case frame based parsing [Carbonell 83, Cellio 84], and those which use ''deeper'' techniques [Grosz 77, Carberry 89, Litman 87, Allen 80], attempting to resolve ellipsis based on discourse structure or deeper reasoning about users' plans. Since this system follows the ''shallow'' approach, we will look at the first two techniques in detail, and then discuss the plan-based approaches.

2.1. Semantic grammars

The semantic grammar paradigm uses rules and categories that represent the semantic distinctions of the target domain together with syntactic distinctions. The parse trees produced, therefore, also represent semantic descriptions of their inputs. The semantic grammar ellipsis resolution technique, in its original form, attempted to parse a sentence fragment bottom-up as a single semantic grammar non-terminal from the previous parse. This non-terminal would then be replaced in the previous parse to produce the resolved utterance. In addition to requiring an entirely different parsing strategy for fragments, since semantic grammars usually parse full sentences top-down, this technique is very sensitive to the way the grammar is constructed. So, given the following two semantic grammar fragments for ***Disk with two ports***:

```
Grammar1:       <disk>  -->  <disk>  <prep>  <disk-property>
```

[2]While some methods of ellipsis resolution require more time to parse an elliptical sentence than a full sentence, I feel that, handled correctly, elliptical utterances should be faster to interpret than full ones, as is the case in this system. This is in keeping with a general belief that a good match between the methods used in natural language interfaces and those used in human cognitive processing will be necessary if dialogue systems are to eventually achieve human-level capabilities.

```
Grammar2:       <disk>  -->  <disk>  <disk-pp>
                <disk-pp>  -->  <prep>  <disk-property>
```

the original semantic grammar algorithm could parse the elliptical utterance ***With one port*** with the second grammar fragment, but not the first. This is because the first has no single non-terminal for a prepositional phrase modifying a disk description. This technique was later improved by allowing the elliptical phrase to match a sequence of non-terminals that occur together in a grammar rule. This later version still required a different parsing strategy for fragments, and still could not handle an elliptical utterance that did not have the same syntactic form as its antecedent.

2.2. Case frame based parsing

Case frame based parsing[3] matches the input recursively to a set of case frames, which specify at a semantic level what sorts of objects and relations can exist in the given domain. Each case frame consists of a *head*, indicating the domain-specific semantic type of the frame, and an unordered set of domain-specific *cases*, each with a unique *case name* and a *filler*, which can be another case frame or a primitive element. The matching of heads, case markers, and primitives is done with a pattern matcher in a style similar to semantic grammars. However, the more flexible structure of the case frame and the greater separation of syntax and semantics allows more detailed and less syntactic ellipsis resolution. For example, given the two inputs:

Show me all single port disks.

Disks with two ports.

the first might be parsed, as shown schematically in figure 2-1, as a ‘‘Display’’ case frame with a ‘‘recipient’’ case indicating the speaker and an ‘‘object’’ case containing a ‘‘Disk’’ case frame, this in turn having case fillers indicating universal quantification and a property of having one port. The second input would be parsed as a ‘‘Disk’’ with the property of having two ports, as depicted in figure 2-2. Note that this is an entirely parallel internal representation, despite the difference in surface syntax between an adjective phrase and a prepositional phrase modifying the noun phrase. This is the sort of utterance that causes problems for the semantic grammar technique, due to the intermingling of syntax and semantics in its representation.

```
[Action
    head: Display
    recip:
        [Object
            head: *me*]
    object:
        [Object
            quant: *univ*
            head: Disk
            port-count: 1]]
```

Figure 2-1: Case frame diagram for antecedent

The remaining step of replacing the appropriate parts of the antecedent with the elliptical utterance is a bit

[3]The term ‘‘case frame’’ here simply means a type of knowledge representation for natural language, and does not imply any particular position regarding the stronger claims of either Fillmore [Fillmore 68] or Schank [Schank 75].

```
[Object
    head: Disk
    port-count: 2]
```

Figure 2-2: Case frame diagram for elliptical utterance

more complicated with this approach, due to the finer grain of the ellipsis resolution[4]. The new case frame is merged recursively into the old structure, taking any case frame selectional restrictions into account. Cases present in the new structure replace corresponding cases in the old structure; cases present in only one of the two are included in the result. Using the current example, this merging produces a representation equivalent to:

Show me all disks with two ports.

as in figure 2-3. The universal quantifier has been retained from the antecedent, as has the top level case frame. As seen in this example, when the top level case frames cannot be merged, a match with the elliptical utterance is attempted at lower levels, until a place is found where it is acceptable. One difficult, open question is what to do when the head of a case frame is replaced: are all the old cases removed, or only those incompatible with the new head? Different systems within this paradigm have taken different courses, depending primarily on what seemed best in the domain at hand. More research is clearly needed here.

```
[Action
    head: Display
    recip:
        [Object
            head: *me*]
    object:
        [Object
            quant: *univ*
            head: Disk
            port-count: 2]]
```

Figure 2-3: Case frame diagram for resolved utterance

2.3. Plan-based techniques

We will now review the plan-oriented approaches. Grosz [Grosz 77] has studied focus in task-oriented dialogues. She defines *focus spaces* using semantic network contexts that represent the subtask structure of the user's plan. These focus spaces are ordered hierarchically, and the system tracks its conversational partner as the dialogue progresses from one subtask to another. Objects in the current focus space or its parent can be referred to using definite references or elliptical utterances, and references by the participants to objects in other focus spaces are used to indicate when the dialogue moves on to a new subtask.

Grosz points out that these techniques are most applicable to dialogues where there is some task structure available. In the database access dialogues she examined, there did not appear to be a useful task structure on which to base a division into focus spaces. If the Psli3 system were to be applied to a domain with significant task structure, this type of focusing could be incorporated using mechanisms

[4] I am tempted to say ''higher resolution resolution'', but I won't.

already present in the system [Frederking 88].

In their work on planning and speech acts [Allen 80], Allen and Perrault attempt to recognize a speaker's underlying plans, and use that information to produce helpful responses. They attempt to infer the speaker's plan from his actions, reasoning about his beliefs and desires. One of their desired results is the ability to recognize the intentions behind elliptical utterances. In the example they give, a person goes up to the information booth at a train station, and asks

The train to Windsor?

to which the person in the booth replies

It leaves at 3:15 from gate 7

Their system goes through a plan inference process to deduce that there are two possible things the first speaker might need to know, and then generates the answer to both. This approach to ellipsis has been further pursued in the work of Carberry [Carberry 89] and Litman [Litman 87]. Carberry's work tries to identify the speaker's goal that an elliptical utterance fulfills, while Litman's work adds the interesting idea of modelling discourse-level plans in addition to the domain-level plans used in the earlier systems.

As will be further clarified below, these methods attempt to handle types of ellipsis that the ellipsis resolution technique described here cannot, even in principle, handle. However, even in the limited domains these researchers have investigated, plan inference is a hard problem, and their reasoning systems generally use a variety of heuristics to reduce search.

2.4. Psli3's technique

The ellipsis resolution in Psli3 is based on the case frame approach, adapted to work within a production system, and extended to handle other dialogue ellipsis types in addition to reformulation, including those with antecedents uttered by the other speaker (in other words, uttered by the system itself). The semantic chart parsing system eases the handling of the additional types of ellipsis, by its straightforward handling of ambiguity, smooth integration of different computational processes, flexible control structure, and lack of hidden state. It also provides an implicit dialogue model, as will be described in section 4.3.

As we describe in the next section, the types of ellipsis handled by plan-based techniques require a different type of processing than the linguistically-oriented ellipses with which we are concerned. If a computationally feasible method of inferring user plans were found, it certainly would be useful in handling this type of ellipsis. But where our methods are applicable, they are much more efficient.

3. Taxonomy of ellipsis

In order to understand the nature of intersentential ellipsis, a taxonomy of dialogue ellipsis was developed. It indicates the different kinds of processing necessary to understand the different classes of elliptical utterances. It also explains how the utterances handled by the ‘‘shallow’’ and ‘‘deep’’ approaches are related, and therefore provides an indication of how the two types of techniques could be combined. A diagram presenting this taxonomy of dialogue ellipsis with examples is shown in figure 3-1.

This processing-based taxonomy of dialogue ellipsis is complete. This claim is based on a study of naturally occurring dialogue ellipsis, described below. All the examples in this fairly large corpus can be clearly assigned to one of these categories, and the categories presented exhaust the dimensions I have found. This is also the basis for the claim that the technique presented here has broad coverage.

Non-antecedent ellipsis

Context-free/Task-dependent—While sitting at a dinner table: ***Salt?***

Context-dependent—In response to the above request: ***Empty!***

Antecedent ellipsis

Non-coreferential

Reformulation—
Parent speaking to a child: ***Please wash your hands.***
Parent when child returns: ***Show me your hands. The other side.***

Coreferential

Elaboration—
Parent again: ***Please clean up your room.***
Parent, as child continues to walk towards the door:
Before you go out to play.

Correction—
Customer in a restaurant: ***I'll have the salmon steak.***
Waitress: ***I wouldn't.***
Customer again: ***The tuna sandwich.***

Echo—
Another customer: ***I'll have a cup of chowder.***
Waitress: ***Chowder?***
Customer: ***Chowder.***

Figure 3-1: Processing based taxonomy of ellipsis

Examples of all the subcategories of antecedent ellipsis, except echo, have been implemented. Non-antecedent ellipses are so difficult that no current technique can in practice handle them, although the plan-based work described above demonstrates how such utterances could be resolved, once problems in large-scale inferential efficiency and knowledge organization are solved. Thus it appears that no other single technique handles as wide a range of ellipsis dialogue phenomena.

3.1. First level

The first division in this classification scheme is between elliptical utterances that have antecedents, and those that do not. It is important to understand that, in this work, the term *antecedent* is used only to refer to a surface linguistic antecedent, that is, a linguistic object (such as a case frame structure) that can be merged with the elliptical utterance. In particular, preceding utterances that require inference to link them to the current utterance (at a semantic or pragmatic level) are not antecedents.

A substantial fraction of elliptical utterances have no antecedent, but rely either on general knowledge or global information about the task currently being carried out to determine their missing parts. For example, in using a natural language interface to an operating system, a user might type:

Directory size?

and expect the system to know from its general purpose that the user wants to be told the size of the directory for the current account, rather than, say, the definition of the term ‘‘directory size’’. Such utterances are referred to here as *non-antecedent* ellipsis[5], a category which is divided into two subcategories: *context-free*, if the resolution is based on general knowledge, possibly including the expected task domain of the system, or *context-dependent*, if the resolution depends on general information about the current state of the dialogue, but not a literal surface antecedent[6]. An example of this second subclass would be a follow-up question to the above:

My other directory?

Here the user clearly wants the size of the other directory, although in another context the user might be asking for a listing of the other directory. This utterance cannot use the preceding one as a direct, linguistic antecedent, because there is no syntactic way to link the two.

The research presented here has not been concerned with interpreting such non-antecedent elliptical utterances, concentrating instead on the other major category: *antecedent* ellipsis[7]. This focus is due both to the fact that non-antecedent utterances are much less frequent in dialogue, and to the level of difficulty of the subclasses of non-antecedent ellipsis: one of its subclasses is too easy, while the other is too hard. Context-free elliptical utterances should be relatively simple to handle in a given domain, using some kind of default mechanism, while context-dependent elliptical utterances can require a substantial amount of inference to link them to the current context.

Interestingly, recent work in using plan recognition in task-oriented systems to resolve elliptical utterances [Carberry 89, Litman 87] seems to apply to precisely the context-dependent category of elliptical utterances that ‘‘shallow’’ approaches cannot handle, while the type of utterance that Carberry says her system cannot handle [Carberry 85, page 197]:

IS: ***Who is the teacher of CS200?***
IP: Dr. Herd is the teacher of CS200.
IS: ***CS263?***

is exactly the antecedent category handled well by ‘‘shallow’’ approaches such as mine (a possibility she suggests herself). So it seems that these two methods could interleave quite well in some future system for handling all elliptical utterances.

This comparison illustrates the usefulness of this taxonomy in understanding how to process elliptical

---

[5]Some non-antecedent ellipses are referred to as *sentence fragments* in plan-based work.

[6]*Telegraphic speech*, such as ***Been looking for you all day. Really tired*** was not considered in this study, since it seems to be an entirely separate phenomenon from ellipsis.

[7]Of course, non-antecedent ellipsis is not really ‘‘intersentential’’ anyway, although it is ‘‘non-intrasentential’’.

utterances. It enables one to apply appropriate techniques to different problems. The case frame based ellipsis resolution approach alone cannot handle inferential ellipsis problems or telegraphic speech, but on the other hand one should not apply knowledge and computation intensive techniques such as plan inference if computationally simpler methods, such as this one, will solve the problem at hand[8].

3.2. Second level

The major sub-division within the antecedent class is between elliptical utterances that are *coreferential* with their antecedents and those that are not. By ‘‘coreference’’ is meant that the utterance describes exactly the same instance of an event or state, rather than another instance, possibly with the same description.

Many natural language interfaces assume that each new utterance describes a new event. In fact, however, the opposite assumption should be made. That is, a new utterance refers to the same event as the last one unless there is some reason it cannot. This applies to full sentences as well as elliptical utterances. Words such as ***another*** and ***again*** seem to be used precisely for preventing coreference, although disjoint reference can also be forced by semantic clashes or pragmatic considerations.

An example of pragmatic non-coreference is that an action already taken cannot be coreferent with a new command. Even if the two action descriptions are identical, they cannot be the same instance of the action. So if a user says in a single utterance:

> ***Show me a large disk. A disk with two ports.***

the request may be interpreted as asking to be shown a large disk with two ports just once, rather than two separate displays of a large disk and a dual port disk. However, if the two top level utterances are separated by a system response, as in:

> ***Show me a large disk.***
> <System displays information about a large disk.>
> ***A disk with two ports.***

it is clear that the user is requesting a second display action, this time of a disk with two ports. It is not clear whether the disk should be large, due to the head replacement problem mentioned above.

3.3. Third level

On a dimension orthogonal to coreference, there are three ways in which the case frame structure of an elliptical utterance can combine with that of its antecedent in the merging process described in section 2 above:

> Extension—The elliptical utterance adds new structure to the old representation.

> Replacement—The elliptical utterance replaces part of the old.

> Repetition—The elliptical utterance only duplicates material already present.

A combination of these is of course also possible. This split simply exhausts the ‘‘syntactic’’ possibilities

---

[8]As Litman points out, there are some cases that can be handled by either technique. Where this occurs, I have considered the utterance to be in the easiest processing class.

(in terms of case frame syntax).

This structural match combines with the presence or absence of coreference to determine the function of the elliptical utterance in the discourse; that is, the effect the speaker is achieving with the elliptical utterance. Four possible discourse functions result, one for each leaf in the antecedent class taxonomy:

- **Reformulation**—If the elliptical utterance is not coreferent with its antecedent, any kind of match will result in a reformulation; that is, a new utterance that simply reuses old structure for the sake of efficiency.
- **Elaboration**—If the elliptical utterance is coreferent with its antecedent and is an extension of its structure, then the elliptical utterance is an elaboration, or more specific description of the previously described action.
- **Correction**—A coreferent elliptical utterance that replaces old structure is a correction. The old description was incorrect, and is replaced by the new. Either speaker may correct an utterance. A special case of correction is the **Question answer**, where the questioned element is replaced by the answer[9].
- **Echo**—If the new utterance is coreferential with its antecedent and only repeats part of it, it is an echo. The repetition is used to ask for confirmation or confirm that the echoed portion is correct.

These categories fully determine the function of the elliptical utterance in the discourse. They appear to also apply to full sentences, although this has not been investigated in detail. These categories were identified at least as early as [Carbonell 83], but without the taxonomic explanation presented here.

It may seem superfluous to rename the leaves of the antecedent branch of the taxonomy, but these terms describe *discourse function*, while terms such as ‘‘coreferent repetition’’ describe the *structure* that determines a discourse function such as ‘‘echo’’.

3.4. Derivation of taxonomy

In order to derive the taxonomy presented above, I conducted an empirical study of dialogue ellipsis. This involved reading through all the short stories in the collection ‘‘Welcome to the Monkey House’’, by Kurt Vonnegut, Jr. [Vonnegut 70]. Every instance of intersentential dialogue ellipsis that I found was recorded, assigned to a category of the evolving taxonomy, and re-assigned as the taxonomy changed. I also looked at a verbal protocol I had collected in an earlier project[10], to see if the types or frequencies of intersentential dialogue ellipsis there were markedly different.

The results of this investigation are presented in tables 3-1 and 3-2[11]. In these tables, ‘‘AC’’ stands for Antecedent Coreferential, ‘‘AN’’ for Antecedent Non-coreferential, ‘‘CD’’ for Context Dependent, and ‘‘CF’’ for Context Free or Task Dependent.

From these results, it is apparent that antecedent coreferential ellipsis accounts for the majority of dialogue ellipsis, both in the short stories and in the actual dialogue. In the short stories, there are two

---

[9]Since the question was not actually being asserted, the answer is not really a correction. This is a special case also due to the strong expectation that the questioned element will be replaced by the next utterance.

[10]A natural language interface to a turbine blade factory shop scheduling system.

[11]Three of the twenty-five short stories contained essentially no dialogue, and therefore contained no dialogue ellipsis.

Overall in 630 literary examples:

AC	358	56.8 %
AN	30	4.8 %
CD	110	17.5 %
CF	132	21.0 %

Overall in 51 verbal protocol examples:

AC	37	73 %
AN	3	6 %
CD	11	21 %
CF	0	0 %

Table 3-1: Statistics compiled on naturally occurring dialogue ellipsis

Story	AC #	AN #	CD #	CF #	Total #	AC %	AN %	CD %	CF %
1	8	0	1	2	11	73	0	9.1	18
2	18	3	3	2	26	69	11.5	11.5	7.7
3	10	2	4	5	21	48	9.5	19	24
4	18	3	16	5	42	43	7.1	38	12
5	24	5	12	4	45	53	11	27	8.9
6	20	2	3	1	26	77	7.7	12	3.8
7	11	0	6	19	36	31	0	17	53 *
8	13	0	6	3	22	59	0	27	14
9	4	1	6	22	33	12	3	18	67 **
10	24	0	4	3	31	77	0	13	9.7
11	16	0	1	0	17	94	0	5.9	0
12	24	0	2	11	37	65	0	5.4	30
13	4	0	2	0	6	67	0	33	0
14	23	0	4	5	32	72	0	12.5	16
15	12	0	5	3	20	60	0	25	15
16	27	0	10	14	51	53	0	20	27
17	31	3	3	5	42	74	7.1	7.1	12
18	4	0	0	1	5	80	0	0	20
19	32	4	6	4	46	70	8.7	13	8.7
20	6	0	6	1	13	46	0	46	7.7
21	18	2	10	13	43	42	4.7	23	30
22	11	5	0	9	25	44	20	0	36

* This contains a task-oriented dialogue: a chess game with spoken moves.

** Another task-oriented dialogue: a radio DJ reading dedications.

Table 3-2: Literary ellipsis statistics broken down by short story

instances where the majority of elliptical utterances were task-dependent (stories 7 and 9), but both of these contained task-oriented dialogues, which are clearly special cases. Thus it is important for any system that understands elliptical utterances to handle the case of antecedent coreferential ellipsis.

While antecedent non-coreferential ellipsis accounts for only a small percentage of the ellipsis in these dialogues, command reformulation is widely believed to be an important use of ellipsis, and it is based on antecedent non-coreferential ellipsis. I believe this discrepancy is due to the master/slave nature of the typical man/machine interaction, versus the generally free-form interactions in these dialogues.

Context dependent ellipsis accounts for a significant percentage of ellipsis in both kinds of dialogues, but, as already pointed out, it cannot be handled by direct linguistic processing, and can be quite difficult due to the deep level of inferencing necessary. The plan-based techniques discussed above attempt to deal with this class. Similarly, context-free or task-dependent ellipsis seems to involve issues somewhat orthogonal to our concerns, although important for task-dependent man/machine interaction.

3.5. A non-processing classification

There is a third dimension along which elliptical utterances can be classified: the relationship of the structure present in the ellipsis to the structure ellided. Although it does not affect the type of processing the ellipsis requires, it is very useful for describing a case frame based ellipsis resolution implementation. In a typical case frame based system, there are two types of case frames: *clausal* frames that represent verb phrases, and *nominal* frames that represent noun phrases. Each of these can be involved in two basic types of ellision: *functional* ellipsis, in which the head and possibly some other cases are missing, and *constituent* ellipsis, in which some subordinant constituent is missing [Shopen 72]. The cross product of these two distinctions gives four structural types of elliptical utterance. An example of each is presented below, preceded by a context sentence, with its ellided part parenthesized and not bold:

Clausal Functional:
Show me a cheap disk.
(Show me) ***A high-density tape drive.***

Clausal Constituent:
Locate a high-density tape drive.
Show me *(a high-density tape drive).*

Nominal Functional:
Show me a tape drive.
(Show me a) ***High-density*** *(tape drive).*

Nominal Constituent:
Show me a high-density disk.
(Show me a high-density) ***Tape drive.***

In these examples, both of the nominal ellipses are also clausal ellipses.

3.6. Implementation of taxonomy

While the design of the ellipsis resolution in Psli3 is based on this taxonomy, the taxonomy as such is not implemented directly anywhere in the system. It is present instead in the program structure of the system, in terms of having different rules handle different discourse-functional and structural subcategories. Although an example run will be presented for only one type (clausal functional reformulation) of ellipsis resolution here, examples of each discourse-functional type of antecedent ellipsis except echoes[12], and all four structural types of reformulation ellipsis, have been implemented [Frederking 88].

4. Semantic chart parsing

In most natural language processing systems, sentences are treated as individual entities. Different kinds of intersentential interactions are handled as distinct problems, often in one or more later stages of processing. Natural language dialogue, however, is an integrated, continuous phenomenon, with sentences interpreted and generated ''on the fly'', within the context of an on-going conversation. Computational systems that are intended to carry out truly natural dialogues must be designed to interpret and generate utterances in an integrated fashion, using a uniform architecture for the processes involved. Such an architecture must provide a continuous context that crosses both sentential and process boundaries. Psli3 is a system designed as a step toward achieving this goal.

Before ellipsis resolution within the semantic chart parsing technique can be presented, the technique itself must be explained. Semantic chart parsing solves the problem of parsing natural language within a production system architecture without losing the benefits of the production system architecture. To begin, chart parsing and production systems will be introduced in turn. After the concept of semantic chart parsing is explained, an example of the interpretation of a full sentence will be given, followed by an example of ellipsis resolution. Then a speculative approach to ambiguous dialogue will be presented.

4.1. Chart parsing

Chart parsing was initially developed as a syntactic parsing technique by Martin Kay [Kay 67, Kay 73]. It is similar to *dynamic programming*, a programming technique where the results of recursive calls are stored in a table indexed by their calling parameter values. When a subsequent call is made with the same parameters, the result is looked up in the table rather than being recalculated. Thus no sub-calculation is ever done more than once. In *chart parsing*, the result of applying a grammar rule to a segment of the input is stored as an *edge* in a table called the *chart*, indexed by its beginning and ending locations in the input. Whenever a rule needs a constituent at a particular position, it looks in the chart first. In this way, no constituent is ever parsed on the same interval more than once. Another important property of chart parsing is that, unlike many other parsing methods, none of the state of the parse is hidden in the control structure of the program. The entire state is present in a declarative form. In addition to greater clarity, this gives the designer complete freedom in choosing parsing strategies. This enabled Earley [Earley 70] to specify a control structure that is theoretically optimal for context-free grammars, resulting in *active chart parsing*. Several examples of Psli3's version of charts will be presented below. A very clear explanation of chart parsing with examples has been provided by Winograd [Winograd 83].

---

[12] Echo ellipsis was not implemented due to its lack of usefulness in a database retrieval system with noise-free, typewritten input. Echoes might prove useful in a domain where the program could recognize semantically questionable but not necessarily incorrect phrases.

4.2. Production systems

OPS5 [Forgy 81] is a widely used *production system language*, a programming language in which programs consist of a large number of *production rules*, rules each consisting of a set of tests, called the *left hand side*, and a set of actions, the *right hand side*. To give the reader some idea of what these rules are like, a sample rule from Psli3 is presented in figure 4-1, with an English translation in figure 4-2. When all the tests on the left hand side of a rule are satisfied, the actions on the right hand side can be carried out. These tests and actions are carried out in *working memory*, an unordered set of records called *working memory elements*. The actions include creating, modifying, and deleting working memory elements, as well as such things as input/output operations. The matching and execution of rules is carried out by an interpreter, which in OPS5 is quite sophisticated. When more than one OPS5 rule has its left hand side satisfied, a small set of general *conflict resolution* principles are used to decide which rule to execute, or *fire*. In order of priority, these principles are:

- Refraction—A rule that has already fired on the same (unmodified) set of matching elements cannot fire again.
- Recency—The rule that matches the most recently modified or created elements is preferred.
- Specificity—The rule that makes the most specific match to its elements is preferred.

After the chosen rule is ececuted, all of the rules are matched again. This matching and execution repeats continuously as long as the system is running. In addition, working memory is actually represented as a caching discrimination network, called the *RETE network*, so that only those working memory elements that change are retested, and tests shared among a number of rules are only performed once. This makes large OPS5 production systems much faster than they would be otherwise.

One motivation for using a production system is that they are widely used in cognitive modelling. I feel that, in the long run, systems that roughly follow human information processing strategies will prove to be the best at holding dialogues with human beings. This is primarily due to the need for a match or coupling between the information processing activities of the partners in a conversation; a system with an architecture that is highly different from the human one would need to simulate some human cognitive activities at some level anyway. It is simpler to do it more directly. A fine-grained simulation is not necessary, however; it does not matter exactly how many milliseconds each operation takes.

Production systems are also attractive from a software engineering perspective for problems requiring complex, flexible control structures. The individual rules fire when applicable, interacting in a complex, highly integrated fashion. High-level control is implemented using declarative *goals* in working memory. A rule creates a goal when it recognizes that some complex action is necessary, usually including parameters in the goal indicating, for example, the object to be acted on. Other rules match this goal and carry out the necessary component actions, and then either delete the goal or mark it as satisfied, if some other rules might need to know of its completion. This system of control still allows other rules to fire at any time, whenever appropriate, while providing large-scale control of the processing.

For both compatibility with cognitive modelling and software engineering reasons, it seems that accessing information by matching patterns against a working memory is more important than the use of a rule-like programming construct. It creates the system's continuous dialogue context, and encourages a style of programming that seems to fit well with human cognitive processing. This is why languages such as OPS5 are preferrable to languages such as Prolog [Clocksin 81]. It is not so much the difference between forward-chaining and backward-chaining, or between a flexible control structure and a more fixed one (although these are both important), as it is the difference between storing information in

```
(p ellipsis-initial-1-unmarked
    (input ^utt nil ^utt nil ^position <npf> ^end left)
    (word-seq ^utt nil ^first <npf> ^last <npl> ^prev <npp>
              ^next <npn> ^state <older3>)
    (state ^state <older3> ^status posted ^utt <n>)
    (POS ^utt nil ^type noun-phrase ^status complete
         ^con <npname> ^state <older3>)
    (reference ^token <npname> ^deref nil ^state <older3>)

    (expect ^con <cname> ^token <etok> ^state <older1>
            ^slot <slot>)
    ( << request question >> ^utt { <n0> <=> 0 } ^object <acttok>
                             ^state <rstate>)
    { (act-reference ^token <cname> ^deref <acttok>
                     ^utt { <n0> <=> 0 })           <actref> }
    (constraint ^element <cname> ^slot <slot> ^type case)
    - (constraint ^slot <> <slot> ^type case ^state <older3>)
    - (goal ^name cp-understand-error-answer ^status subgoaled)
    - (bindings ^rule ellipsis-initial-1-unmarked ^first <npf>
                ^last <npl> ^bindings <older3> <rstate> <slot>)
    - (bindings ^rule ellipsis-initial-1-unmarked ^first <npf>
                ^last <npl> ^bindings <unbound1> <unbound2>
                                        <unbound3> { <> <n0> } )
    -->
;;; Build a new state.
    (bind <new-state> (gint))
;;; The state is posted, since it doesn't need semantic checking:
    (make state ^utt <n> ^state <new-state> ^status posted)
;;; This is built for display purposes
    (make POS ^utt nil ^type ellipsis ^con <cname>
              ^state <new-state>)
    (make POS ^type verb-phrase ^con <cname> ^status complete
              ^state <new-state> ^utt nil)
    (make ancestor ^state <new-state> ^ancestor <rstate>)
    (make ancestor ^state <new-state> ^ancestor <older3>)
;;; Inhibit further rule firings using same expectation.
    (make bindings ^rule ellipsis-initial-1-unmarked ^first <npf>
             ^last <npl> ^bindings <older3> <rstate> <slot> <n0>)
;;; Propose the connections.  Inhibit expectations.
;;; Also makes this part of a new concept, and a new utterance.
    (make act-reference ^1 (substr <actref> 1 inf) ^utt nil
                ^deref nil ^<slot> <npname> ^state <new-state>)
    (make constraint ^element <cname> ^slot <slot> ^type case
                     ^filler <npname> ^state <new-state>)
    (make word-seq ^first <npf> ^last <npl> ^prev <npp>
                   ^next <npn> ^state <new-state>))
```

Figure 4-1: Production rule for unmarked, initial, verb phrase functional ellipsis

pattern-matched data structures and storing information in a simpler, more sequential database.

This property, together with the flexible control structure provided by production rules, also makes them seem to be a good choice as a high-level language for connectionist architectures, and some very preliminary work has already been done in this direction [Touretzky 88]. More information on programming in OPS5, including extensive descriptions and examples, can be found in [Brownston 85].

```
RULE ellipsis-initial-1-unmarked:
IF     the left end of the input sentence is <npf>
       and state <older3> goes from <npf> to <npl>, preceded by <npp>
             and followed by <npf>
       and this state is part of utterance <n>
       and this state is a complete noun phrase named <npname>
       and this state does not have a referent assigned
       and verb phrase <cname> has a case named <slot>
       and previous utterance <n0> was a request for or question about
             act <acttok> in state <rstate>
       and verb phrase <cname> was used to refer to <acttok>
       and the case named <slot> was filled in the previous utterance
       and this noun phrase is not assigned to some other slot
       and we are not in a question answering situation
       and this rule has not fired on this location for this noun phrase,
             sentence, and slot yet
       and this rule has not fired on this location for an earlier utterance
THEN   obtain a unique name for a new state
       build a posted state for utterance <n>
       indicate that this state is an ellipsis
       indicate that this state is a verb phrase
       indicate that this state has ancestor <rstate>
       indicate that this state has ancestor <older3>
       indicate that this rule has now fired on this location, noun phrase,
             sentence, slot, and utterance
       build a copy of verb phrase <cname> with <slot> now filled by <npname>
             as part of the new state and current utterance
       indicate a constraint that case <slot> is filled by <npname>
             in the new verb phrase
       indicate that this state covers the same interval as <npname>
```

Figure 4-2: English version of rule in previous figure

4.3. Combining the two

In semantic chart parsing, chart parsing has been adapted for use within a production system. The edges consist of sets of working memory elements, which are called *states*, and include semantic descriptions and referent identifications in addition to syntactic descriptions and starting and ending positions. A trace of one of these charts is shown in figure 4-3 (explained in detail in the next section); a single, rather syntactic, state is illustrated in figure 4-4, along with its English description. The rules for combining states to produce new states require that they be semantically and referentially compatible, as well as adjacent and syntactically correct[13]. This extension to include semantics and reference is quite natural, since in a production system everything is implemented via one type of rule, including semantic interpretation, referent identification, etc. This scheme corresponds to a *general* chart parser, and not an *active* chart parser, for two good reasons.

From a theoretical standpoint, active chart parsers are best suited for context-free languages, and no one believes that natural language semantics is ‘‘context-free’’. That is to say, the manipulation of semantic entities within a natural language system can be thought of, on another level, as a *syntactic* process as well, one that is clearly not representable by a context-free language, even if the syntax of natural language perhaps is. So any mechanism that works with semantics must have more than context-free power.

---

[13]Inputs containing errors, including semantic and referential problems, can in fact be handled within this framework. They simply must be explicitly recognized and corrected, as will be shown in the example.

From a practical standpoint, active chart parsing requires the storage of partially matched rules in the chart's edges. Since the matching here is being handled by OPS5's RETE network, partially matched rules are never available to the Psli3 system. They are buried within the OPS5 machinery, an unaccessible part of the system's environment. Only when a rule is fully matched does anything happen at the Psli3 level of the system.

Since all of the action takes place within the chart, all types of ambiguity can be handled in a direct manner, by creating competing states within the chart. In earlier incremental interpretation, production system parsers [Frederking 85], I had tried to handle ambiguity using backtracking or a search-tree approach. But in the first approach, backtracking with semantic structures proved to be very difficult, especially in a production system[14], while in the second approach, the search trees generated grew much too large and needlessly cumbersome. The semantic chart parser is in comparison remarkably straightforward, and much more efficient. A similar system has been developed independently by Jardine and Shebs [Jardine 83], based on a parallelization of the PHRAN parser [Wilensky 80]; however, it was not a part of an overall production system, which is what provides the high degree of integration and flexible dialogue control exhibited by Psli3.

Within the semantic chart parsing paradigm, a number of design choices remain open. In particular, the complete freedom in choice of control strategies of chart parsing is retained. Psli3 has been designed to parse in a left-to-right, primarily bottom-up, primarily breadth-first fashion. Left-to-right parsing seems the obvious choice with typewritten input; in a system with spoken input, an island-driven approach might be more appropriate. The system builds its interpretation mostly bottom-up because both elliptical utterances and error correction require bottom-up interpretation. All valid interpretations are produced breadth-first because the system should notice true ambiguities. Since ''valid'' here includes semantic and referential validity, the use of incremental semantics to prune out invalid interpretations helps to prevent these choices from leading to a combinatorial explosion, as does the chart technique[15].

4.3.1. Dialogue modelling

The working memory of the production system also serves as an implicit dialogue model, since the charts built for previous utterances remain available to the rules interpreting the current utterance. The case frame structures residing in working memory represent both natural language input and output. Together with other data structures in working memory, some of which represent the utterances as actions by the user or the system, and some of which represent accessed frames from the semantic network, including reference sets, they provide a model of the dialogue processing that has occurred during a run of the system up to the current time.

This approach could be adapted to provide the functionality of explicit discourse models that have been proposed [Grosz 77, Carberry 89, Litman 87, Allen 80, Grosz 86], as explained elsewhere [Frederking 88]. However, dialogue modelling would remain a (somewhat abstract) subset of cognitive modelling. In

---

[14]In addition to the problems associated with keeping track of inferences and withdrawing them, backtracking in OPS5 is in general difficult. This is due both to deletion from the RETE network being relatively slow, and to the danger of partially deleted structures inadvertantly causing inappropriate rules to fire. It *is* possible, but not especially pleasant.

[15]Additionally, a system with more extensive syntactic coverage could be designed to parse as deterministically as possible [Marcus 80]. While truly deterministic parsing is probably not possible, it is a good idea to restrict the number of ambiguous interpretations as much as possible. There is enough real local ambiguity in natural language to make the semantic chart necessary.

addition to the obvious connections between natural language processing and cognitive psychology, there is a sort of parsimony involved. For example, if cognitive processing in general has a recency property, it is unparsimonious to specifically implement recency separately for ellipsis, references, etc., by modelling each with its own stack. Another problem with stacks is that they often do not correspond well to the linguistic phenomenon beiong modelled. Quite often in the literature, an author will describe some linguistic behavior as involving a stack, except that they violate the stack discipline by sometimes looking down the stack for something without ‘‘popping’’. Recency allows one to violate the stack without apologies, since working memory is *not* a stack. The fact that these stacks do have to be violated is in fact an argument for the ‘‘working memory with recency’’ point of view; there is no particular reason to believe that true stacks occur inside human language users.

In addition to naturally providing properties such as recency, the fact that all the state in a production system is stored in the working memory means that none of it is hidden away in the control structure of code. Complex sub-dialogues and shifts in conversational control present no major programming problems, beyond the unavoidable need to prescribe what behavior is appropriate when. Psli3 takes advantage of this lack of hidden state in recursively parsing input during clarification dialogues [Frederking 88].

4.4. Interpreting a sentence

The semantic chart parsing technique will be demonstrated using the interpretation of the sentence:

> ***Show me xn*** [sic] ***irregular bone***

This sentence will also serve as the antecedent for the example of ellipsis resolution. Relative to the system’s knowledge, it contains three different types of local ambiguity:

- The case of ***me*** is locally ambiguous in a left-to-right parse, between a person being shown something and a person being shown to another person, as in ***Show me to your mother.***
- There is a locally ambiguous spelling mistake. The user has typed ***xn***, which is not in the lexicon and can be spelling corrected to either ***in*** or ***an*** using the implemented lexicon.
- The word ***bone*** can either be a noun itself, or part of the fixed phrase ***irregular bone***. Both of these are names of frames in the application database.

A trace of the chart built by Psli3 while interpreting this sentence is shown in figure 4-3. This and the other charts in this paper were created by a set of special tracing rules during actual runs of the system. Each state created in the chart is represented by one or more line segments, with time progressing from top to bottom, and with the horizontal position indicating which segment of the input the state represents. States represented by a line of hyphens, such as the first instance of T6669, are ‘‘proposed’’ states that must be verified and then ‘‘posted’’ before becoming constitutents in larger structures. States represented by more than one posted entry, such as T6655, simply contain more than one element that the tracing rules print out. The arcs between states represent backlinks, which indicate the ancestry of a state. These have only been drawn in for states that are part of the final interpretation. The states consist of sets of working memory elements, as previously mentioned, and generally represent either surface case frame structures or deeper semantic or reference structures such as deep case frames representing actions or sets of semantic network frames. Figure 4-4 depicts a single state from this chart, along with its English description. The ‘‘word-seq’’ element indicates the state’s position in the input, while the ‘‘state’’ field of each element contains the unique name of this state.

```
  1    2    3    4    5    6

 SHOW ME   XN   IRREGULAR
                    BONE *CR

!===! VERB-PHRASE 2     T6648
!===! VERB     T6648
!===! VERB-PHRASE 1     T6655
!===! VERB     T6655
!===! VP-REF 1     T6661
     !===! NOUN     T6663
     !===! PRONOUN     T6663
     !===! NOUN-PHRASE COMPLETE     T6665
     !===! NP-REF     T6667
!========! VERB-PHRASE 1     T6668
!--------! VP-REF 1     T6669
!========! VP-REF 1     T6669
!========! VERB-PHRASE COMPLETE     T6671
!--------! VP-REF COMPLETE     T6672
Attempting (ambiguous) spelling correction: XN
Words tried: IN AN
          !===! DETERMINER     T6676
          !===! NOUN-PHRASE     T6678
          !===! PREPOSITION     T6680
               !===! NP-NULL     T6681
                    !===! NOUN     T6682
               !========! NOUN     T6690
               !========! NOUN-PHRASE COMPLETE     T6706
          !=============! NOUN-PHRASE COMPLETE     T6710
          !=============! NP-REF     T6711
!======================! VERB-PHRASE COMPLETE     T6716
!----------------------! VP-REF COMPLETE     T6717
!======================! VP-REF COMPLETE     T6717
                        !===! PUNCTUATION     T6719
!===========================! IMPERATIVE     T6720

 SHOW ME   XN   IRREGULAR
                    BONE *CR

  1    2    3    4    5    6
```

Figure 4-3: Chart built for ***Show me xn*** [sic] ***irregular bone***

The parse begins when a production rule fires that gets input from the user and creates a linked list of input states representing the input words[16]. These input states are marked to indicate each end of the utterance and each end of each sentence in the utterance[17]. Discourse information is also recorded, indicating that this utterance was an action by the current speaker. Then the left-most word is activated. Left-to-right parsing is enforced by marking an input state ‘‘active’’ only after all the rules that can operate on the input to its left are finished.

Two lexicon rules fire on the first word, one creating a state representing the declarative verb ***show*** (state T6648 in figure 4-3), the other a state representing the imperative of ***show*** (T6655, shown also in figure

[16]Reading and tokenizing the user input and creating input states is handled by a Common Lisp function called from the right hand side of an OPS5 rule, as are several other utility functions, such as spelling correction. Since Psli3 treats words as atomic, these Lisp functions can be thought of as new primitives. All other processing in the system is carried out within OPS5. The interested reader can find copious details elsewhere [Frederking 88].

[17]‘‘Sentence’’ is used here to refer to syntactically separate parts of the utterance, whether full sentences or elliptical.

```
(STATE ^STATUS POSTED ^UTT 1 ^STATE T6655)
(ANCESTOR ^ANCESTOR I6645 ^STATE T6655)
(POS ^CON T6656 ^TYPE VERB ^STATE T6655)
(POS ^CON T6656 ^STATUS 1 ^TYPE VERB-PHRASE ^STATE T6655)
(WORD-SEQ ^FIRST 1 ^LAST 1 ^NEXT 2 ^PREV 0 ^STATE T6655)
(ACT-REFERENCE ^TOKEN T6656 ^ACTION DISPLAY ^MODE IMP ^E-SUBJ T
               ^STATE T6655)
(EXPECT ^TOKEN T6660 ^CON T6656 ^TYPE E-UNMARKED ^MARKER 1ST
               ^SLOT RECIP ^STATE T6655)
(EXPECT ^TOKEN T6659 ^CON T6656 ^TYPE E-MARKED ^MARKER TO
               ^SLOT RECIP ^STATE T6655)
(EXPECT ^TOKEN T6658 ^CON T6656 ^TYPE E-UNMARKED ^MARKER 2ND
               ^SLOT OBJECT ^REQUIRED T ^STATE T6655)
(EXPECT ^TOKEN T6657 ^CON T6656 ^TYPE E-MARKED ^MARKER FOR
               ^SLOT BENE ^STATE T6655)
```

State t6655 has been posted (it is active), as part of utterance 1.
This state is backlinked to state i6645 (an input word).
This state represents a verb part-of-speech.
This state also represents a verb phrase, with 1 required case.
This state represents the first word of the input.
This state contains action t6656, which is a "display" action, in imperative
    mood, with the subject already taken care of.
This state expects an indirect object for action t6656, which will be the
    recipient of the action,
or a recipient contained in a prepositional phrase marked by "to".
This state expects a direct object for action t6656, which will be the
    object acted on by the action, and is a required case.
This state expects a prepositional phrase marked by "for", which will be the
    benefactor (who the action is done for).

Figure 4-4: Actual representation of state T6655 in working memory

4-4 in detail). These states each appear twice in figure 4-3 because they each contain markers for both the ‘‘verb’’ and the ‘‘verb-phrase’’ category. Each of these states also indicates that it spans the interval [1:1] in the input, indicates that it describes a ‘‘display’’ action, and lists the cases it can take, how they are marked, and whether they are required. The declarative verb state indicates that it requires a syntactic subject, while the imperative indicates that it does not. The initial referent for the imperative verb is then created, with none of its deep cases filled (T6661). Since imperative verbs are taken to refer to the (as yet hypothetical) action that will be taken to satisfy them, this new action is simply fabricated. If this were a declarative verb, a search for a referent would have to be carried out[18].

No more rules can fire on the current content of the chart, so the next input state is activated. The lexicon rule for ***me*** then creates a noun (and pronoun) referring to the special token ‘‘*me*’’ (T6663). This starts a new noun phrase (T6665) because there is a noun with no preceding determiner or quantifier. This noun phrase is marked as (possibly) ‘‘complete’’ because its head has been found. Then a referent for ‘‘*me*’’ is created (T6667) that represents the current speaker.

Now the first of two verb phrases on the interval [1:2] is created (T6668), representing the verb and its indirect object, with the case frame for ‘‘*me*’’ filling the deep recipient case of the display case frame. A state representing this verb phrase’s referent (T6669) is created, based on the referents of the ancestor

---

[18]This same hypothetical action is in fact made real if the system carries out a response to the request. The system as implemented does not include rules for finding declarative referents. This does not affect this example, since the declarative started here has no subject, and so is never completed.

verb (T6661) and noun (T6667) phrases and the case relationship between them, found in state T6668. This state is ''proposed'' rather than ''posted'', meaning that it is only a hypothesis that must be confirmed by some other (usually semantic) rule before other parser rules can act on it. A rule associated with the implementation of displaying then checks the proposed display action, and indicates that the speaker is an acceptable recipient by changing its status to ''posted'', which causes state T6669 to be redisplayed in figure 4-3 with a double line[19].

The other verb phrase on the interval [1:2] (T6671) is constructed in a similar fashion, with two main differences. In this verb phrase, ''*me*'' fills the direct object case, which is the only required case for ***show***, so the verb phrase is marked as (possibly) syntactically ''complete''. The other difference is that, for this verb, the direct object is also the deep object, and the speaker is not an acceptable filler of the object of the display action. This means that the verb phrase referent state (T6672) is never marked by a semantics rule as ''posted'', but is simply left in the chart as ''proposed''. It cannot be used as a constituent by other parser rules.

No more rules can fire, so once again the next input state, this time the one for ***xn*** on [3:3], is activated. No rules at all can match this state. An error-detection rule that watches for input states with no descendants now fires, creating a goal of correcting this evidently malformed word. A spelling correction rule calls a Common Lisp spelling corrector, and puts the resulting answers from the (small) current lexicon, ***in*** and ***an***, into two input states, each on the interval [3:3]. These incompatible states coexist, along with ***xn***, on the same interval without interfering with each other, just like any other ambiguity.

The lexicon rule for ***an*** now indicates (T6676) that there is an indefinite determiner. Another rule, which spots determiners without preceding quantifiers, starts a new noun phrase on [3:3]. This noun phrase (T6678) is not marked ''complete'' because it has no head yet, and thus cannot be a syntactically complete noun phrase. Then the lexicon rule for ***in*** recognizes (T6680) that the preposition ''in'' is present on [3:3].

The input advances again, to ***irregular***. Its lexicon rule[20] indicates (T6681) that it is the beginning of a fixed phrase. The input advances once again, and ***bone*** is recognized (T6682) as a singular noun on [5:5], referring to the ''bone'' frame in the semantic network. This causes the ''bone'' frame to be loaded into working memory from the semantic network, if it is not already present.

A singular noun is then created (T6690) on [4:5] referring to the frame ''irregular_bone''. This frame is also loaded into working memory if not present. A new noun phrase is built (T6706) for this noun[21]. In addition, the old noun phrase started by ***an*** is extended with the noun on [4:5], to produce a ''complete'' indefinite noun phrase (T6710) on [3:5]. A referent-building rule then creates a set descriptor on [3:5] for

---

[19]This is a small demonstration of how syntactic and semantic rules can interact in a closely integrated fashion, even though they are cleanly separated in the design. The confirmation of a state can be carried out by a set of rules acting on the description of a proposed action, with no special connection to the parser, but simply verifying the acceptability of a parameter to a hypothetical action.

[20]This lexicon rule and the one for the corresponding full phrase ***irregular bones*** are generated automatically from lexicon frames in the semantic network, as are all other open word class lexicon rules. The addition of these lexicon frames was the only change made to the database, which had been built previously for another project.

[21]The existance of a state representing the preposition ''in'' on [3:3] allows this new noun phrase to be built. If the only state on [3:3] were the one for ***an***, this rule would not start a new noun phrase at 4, since it would have to be spurious.

a singleton set of type irregular_bone (T6711).

Now the complete noun phrase on [3:5] is combined with one of the verb phrases on [1:2] as its direct object, to produce (T6716) a verb phrase on [1:5]. Since the other verb phrase on [1:2] already has a direct object, it cannot be used[22]. As before, the new verb phrase with a direct object is marked as complete, and a new action with this singleton set as its display object is proposed (T6717). Since the display rules do know how to display sets of objects from the database, a rule associated with objects of display actions marks this state as posted.

The input is again advanced, and state T6719 is created for an end-of-sentence punctuation mark on [6:6]. We now have an imperative verb phrase with a posted referent, starting at the beginning of the sentence, followed by an end-of-sentence punctuation. This allows a rule to decide (T6720) that this is an imperative sentence on [1:6] and that it is a request for the described action.

The final interpretation of the input and its referent are shown schematically in figure 4-5. Terms in all capitals refer to specific objects the system knows, such as users and database frames. Terms such as ‘‘t6656’’ are unique identifiers that are used to establish references between structures: the filler of the ‘‘deref’’ case of the ‘‘Act-reference’’ frame is object ‘‘t6718’’, which is also (identically the same object as) the filler of the ‘‘object’’ slot of the ‘‘Request’’ frame. The nesting of case frame structures is done using such identifiers; they are drawn here as two separate, nested data structures simply for clarity. There is no difference between ‘‘Act t6718’’ filling the ‘‘deref’’ slot and filling the other frame’s ‘‘object’’ slot. References to identical objects are indicated in the diagrams by arrows connecting them.

```
[Act-reference t6656
    action: Display
    mode: Imperative
    recip:
        [Reference t6668
            head: *ME*
            deref: BDF]
    object:
        [Reference t6686
            det: Indefinite
            head: IRREGULAR_BONE
            number: Singular
            deref: t6712]
    deref: t6718]

[Request t6721
    object:
        [Act t6718
            action: Display
            mode: Imperative
            recip: BDF
            object:
                [Schema-spec t6712
                    slot: is-a*
                    filler: IRREGULAR_BONE
                    number: singular]]]
```

Figure 4-5: Case frame diagram for ***Show me an irregular bone***

[22]This is an example of an incorrect local interpretation not hindering a competing correct one.

Since this is also the end of the input, the existance of a posted state spanning the whole utterance means that a consistent interpretation for the whole input has been found. So a rule indicates parsing success, and the interpretation phase is completed. Next follows an ''installation'' phase, where backlinks are used to find and mark all the states in the chart that contributed to the final interpretation. After the ''response'' phase generates the requested display, a ''garbage collection'' phase deletes all other states from the chart.

4.5. Interpreting elliptical utterances

In keeping with the comments in section 3, the overall process of interpreting an input utterance could be viewed as follows:

- Interpret the input as a full sentence
- If this fails (not a complete sentence), interpret the input as coreferent antecedent ellipsis, with its discourse function determined by the structural match
- If this fails (no coreference possible), interpret the input as noncoreferent antecedent ellipsis (reformulation)
- If this fails (no viable antecedent), call the plan-based techniques

A portion of this process was implicitly implemented in Psli3, since the rules that carry out coreference have precedence over those that do not, due to the greater specificity of their working memory matches. This organization is only meant to be suggestive. It is clearly oversimplified; there is often coreference between full sentences, and a system capable of plan-based reasoning would try to generate justifications in all situations.

4.5.1. Interpreting clausal functional reformulation ellipsis

We will now look at how Psli3 interprets one specific form of intersentential ellipsis: a clausal functional ellipsis that is used as a reformulation. As discussed in section 3, a reformulation ellipsis is one that does not corefer with its antecedent. The speaker is simply reusing the structure of the previous utterance to simplify the production of a new utterance. This seems to be the most frequent type of ellipsis in database retrieval applications, presumably due to the repetitious nature of the task. This example uses the correct version of the sentence just discussed as its antecedent:

Show me an irregular bone
THORACIC_VERTEBRA
Some flat bones

Structurally, this is an example of clausal functional ellipsis, since the surrounding clause has been ellided to leave a bare noun phrase, with no clear sentential function out of context. It is a reformulation because the antecedent's action has already been carried out, so a new hypothetical action must be created for the new utterance.

The chart built in interpreting this utterance is shown in figure 4-6. The utterance is interpreted in the same way as the noun phrase in the previous example until state T1764 has been produced. Then an elliptical utterance interpretation rule (the rule shown in figures 4-1 and 4-2) that watches for noun phrases at the left end of a sentence fires twice. This rule resolves initial clausal functional ellipses with *unmarked* case, that is, those not preceded by a preposition (case marker). It copies the top-level working memory element of the most recent verb phrase into a state in the current chart, replacing an already filled case with the new noun phrase. The most recent verb phrase is chosen automatically, due

```
  1    2    3    4

 SOME FLAT BONES
               *CR

!===! DETERMINER     T1744
!===! NOUN-PHRASE     T1746
     !===! NP-NULL     T1748
          !===! NOUN     T1749
     !========! NOUN     T1751
!=============! NOUN-PHRASE COMPLETE     T1763
!=============! NP-REF     T1764
!=============! VERB-PHRASE COMPLETE     T1766
!=============! ELLIPSIS     T1766
!-------------! VP-REF COMPLETE     T1767
!=============! VP-REF COMPLETE     T1767
!=============! VERB-PHRASE COMPLETE     T1769
!=============! ELLIPSIS     T1769
!-------------! VP-REF COMPLETE     T1770
               !===! PUNCTUATION     T1772
!==================! IMPERATIVE     T1773

 SOME FLAT BONES
               *CR

  1    2    3    4
```

Figure 4-6: Chart built for ***Some flat bones***

to the recency criterion in OPS5's conflict resolution. Only the top-level element needs to be copied because the old chart is still in working memory, and the unique identifiers used as pointers to its subsidiary structures still work. This is what makes the resolution process faster than interpreting a full sentence. At the clause level, an unmarked elliptical utterance can only replace a case that was present in the antecedent; if the case was not present, the speaker must use a marked elliptical utterance. This seems to be simply a requirement of English ellipsis syntax.

The first time this rule fires, it replaces the object case, to produce a verb phrase (T1766)[23] corresponding to the sentence:

Show me some flat bones

Then another rule produces a new verb phrase reference (T1767) for the ellided verb phrase by extending the reference of the parent verb phrase of its antecedent. This state only differs from other verb phrase references in that its ancestor is in a preceding utterance. It is then posted by the same semantic rule that approved the display object in the previous example. A diagram of the resulting case frame structure is presented in figure 4-7; the ''Act-reference'' and its ''recip'' have the same unique identifiers as in the antecedent chart, showing that they are the same objects[24].

The second time, this rule replaces the recipient, to produce (T1769) the equivalent of:

---

[23]This state appears twice in the diagram, indicating that it is a syntactically complete verb phrase due to its elliptical case frame relationship to the previous utterance's verb phrase.

[24]The ''actor: *PSLI*'' case in the verb phrase reference comes from the antecedent, and was added to it during the ''response'' phase. The discontinuity in the numbering of the identifiers is due to the fact that these diagrams come from a different run, and have been ''unified'' with the other diagram by hand.

```
[Act-reference t6656
    action: Display
    mode: Imperative
    recip: t6668
    object:
        [Reference t1747
            det: Indefinite
            head: FLAT_BONE
            number: Plural
            deref: t1765]
    deref: t1768]

[Request t1774
    object:
        [Act t1768
            action: Display
            mode: Imperative
            actor: *PSLI*
            recip: BDF
            object:
                [Schema-spec t1765
                    slot: is-a*
                    filler: FLAT_BONE
                    number: plural]]]
```

Figure 4-7: Case frame diagram for resolved ellipsis: *(Show me)* ***Some flat bones***

****Show some flat bones an irregular bone***

This time the proposed verb phrase reference (T1770) does not get approved by the semantics, since the display rules do not know how to display something *to* a database object. The proposed case frame structure is shown in figure 4-8. There is no ‘‘Request’’ case frame in this diagram because the ‘‘Act’’ failed to get posted.

No additional resolutions are attempted, even if there are earlier utterances in working memory, due to some bookkeeping working memory elements that indicate that resolutions have been attempted with this utterance. The input is advanced, and the end punctuation is recognized (T1772). Then the same request rule fires as for a full sentence (T1773), since there is now a completely normal verb phrase and verb phrase referent spanning the input.

There is a separate rule for initial, marked clausal functional ellipsis:

To Bill

which does not require the case to have been filled in the antecedent (unlike the unmarked elliptical utterance rule above).

There is also a rule for extending an elliptical utterance with a marked case:

Some flat bones to Bill

and a rule for adding a second unmarked case to an elliptical utterance:

Bill some flat bones

In this last case, the rule implements a syntactic constraint that there must be only two noun phrases, and their surface cases must be the indirect and direct object, respectively. Presumably this has to do with the way unmarked cases are parsed by English speakers[25]. In a real cognitive model of English parsing, one would expect this relationship to arise naturally somehow from the structure of the system.

```
[Act-reference t6656
    action: Display
    mode: Imperative
    recip:
        [Reference t1747
            det: Indefinite
            head: FLAT_BONE
            number: Plural
            deref: t1765]
    object: t6686
    deref: t1771]

[Act t1771
    action: Display
    mode: Imperative
    actor: *PSLI*
    recip:
        [Schema-spec t1765
            slot: is-a*
            filler: FLAT_BONE
            number: plural]
    object: t6712]
```

Figure 4-8: Case frame diagram for incorrectly resolved ellipsis:
***** *(Show)* ***Some flat bones*** *(an irregular bone)*

One possible problem with this treatment of clausal functional ellipsis is that the initial noun phrase of a declarative sentence would trigger these rules. This could cause an efficiency problem in a system that frequently used declarative sentences, although the chart mechanism would prevent the spurious ellipsis resolutions from being activated, since they would not cover the whole input. This could be prevented by slightly rewriting these rules so that they could only fire after the whole utterance had been parsed, and no top-level utterance had been produced.

Several separate rules are necessary to handle this class of elliptical utterance, given the structure of OPS5, because of the need to match several types of structure and capture several different linguistic constraints within this class. This is not an *ad hoc* collection of rules; no more rules for this class of elliptical utterance should be needed to extend the system to full English.

This single technique handles a wide range of ellliptical utterances. The ellipsis resolution process is easy and natural due to the case frame representation and bottom-up parsing, but can be quite difficult in systems that either do not understand case-level structure or require top-down parsing. Case structure appears to be the natural unit of ellipsis. In Psli3, the modularity of production rules also simplifies the process, as only a few new rules are needed, to describe the allowable syntactic forms and the antecedent connections. The bulk of the rules used in interpreting elliptical utterances are the same ones used in full sentence processing.

---

[25] In fact, in German, which has morphologically marked surface cases, three noun phrases can occur together in such an elliptical utterance, without causing any difficulty. This supports the conjecture that this is an English-specific constraint, perhaps caused by the lack of constructs in full sentences containing three adjacent unmarked noun phrases.

4.6. Dialogue charts

One interesting possibility that has so far only been touched on in this work is the ability to handle dialogue level ambiguities within the chart paradigm. Since all of the processing occurs within a rule-based system, the chart in working memory can be extended to handle ambiguities in the interpretation of a complete utterance, including the analysis of its purpose. The presence of a preferred dialogue relationship between two sentence-level parts of an utterance can be used to prefer one interpretation over another. This use of the semantic chart is referred to as a *dialogue chart*, and is demonstrated here with an example of an elliptical error correction.

If after the interaction:

> ***Show Phil a rib for me***
> RIB_NUMBER_7

a user were to type:

> ***Rich***

the system would find two valid interpretations, since ***Rich*** can replace either ***Phil*** or ***me***[26]. These two interpretations are shown in figures 4-9 and 4-10[27].

If the user had instead typed:

> ***Show Phil a rib for me***
> RIB_NUMBER_7
> ***For Rich***

the interpretation would be clear, since the user explicitly indicates the case marker.

The user could attempt to correct the awkwardness in the first version of this dialogue using a follow-on elliptical utterance:

> ***Show Phil a rib for me***
> RIB_NUMBER_7
> ***Rich. For Rich***

where the second utterance is meant as a replacement for the first, and both refer to the same (as yet hypothetical) action. This would therefore be a correction ellipsis according to the taxonomy. The dialogue chart comes into play because each of these elliptical sentences is actually in isolation ambiguous, but the pair together has a preferred interpretation, due to this error correction relationship.

The chart created during the interpretation of this utterance is shown in figure 4-11[28]. In the first (top) half, the first part of the utterance produces two resolved verb phrases, by the process described in the previous section. One represents the equivalent of the phrase ***Show Rich a rib for me***; the other represents the equivalent of ***Show Phil a rib for Rich***. In the second half, the second part of the utterance also produces two interpretations, corresponding to ***Show Phil a rib for Rich*** and ***Show Rich a rib for Rich***. This last interpretation uses the first part of the utterance as its antecedent, which is in general valid, although here it produces an interpretation that violates common sense.

---

[26]There is probably a preference for replacing ***Phil***, as this utterance makes more sense in context, but the preference is weak enough that the utterance is definitely unclear.

[27]A third interpretation, ***Show Phil Rich for me***, is attempted, but is ruled out by domain semantics.

[28]State T0774 represents the failed attempt to interpret the ellipsis as ***Show Phil Rich for me***.

```
[Act-reference t0687
    action: Display
    mode: Imperative
    recip:
        [Reference t0765
            head: RHT
            deref: RHT]
    object: t0708
    bene: t0725
    deref: t0772]

[Request t0778
    object:
        [Act t0772
            action: Display
            mode: Imperative
            recip: RHT
            object: t0717
            bene: BDF]]
```

Figure 4-9: Case frame diagram for *(Show)* ***Rich*** *(a rib for me)*

```
[Act-reference t0687
    action: Display
    mode: Imperative
    recip: t0697
    object: t0708
    bene:
        [Reference t0765
            head: RHT
            deref: RHT]
    deref: t0769]

[Request t0780
    object:
        [Act t0769
            action: Display
            mode: Imperative
            recip: PJH
            object: t0717
            bene: RHT]]
```

Figure 4-10: Case frame diagram for *(Show Phil a rib for)* ***Rich***

The only really new thing in this chart is state T0802, which was created by a rule that says: if an antecedent immediately precedes its elliptical sentence in the same utterance, and their verb phrases have the same unique identifier, then the new utterance subsumes the old, in a state that spans both of them. In other words, it is a correction ellipsis. In this example, the state created spans the whole input, which is necessary in order for the request to be activated and responded to. This means that the spurious sentence-level interpretations are ignored, and the overall correct interpretation recognized, in the same way that local incorrect interpretations are ignored in a single sentence chart.

Same-utterance elaborations are handled in a similar way, since they are also coreferent. Same-utterance reformulations are also implemented, but they require an additional rule that concatenates several independent utterances into a single, utterance-spanning one if there is no utterance-spanning state. This can be looked on as the default case, used when no recognizable inter-sentential relationships can be found. This follows the priority described at the beginning of section 4.5.

```
  1    2    3    4    5

 RICH *PERIOD
           FOR  RICH *CR

!===! NOUN     T0762
!===! PROPER-NOUN     T0762
!===! NOUN-PHRASE COMPLETE     T0764
!===! NP-REF     T0766
!===! VERB-PHRASE COMPLETE     T0767
!===! ELLIPSIS     T0767
!---! VP-REF COMPLETE     T0768
!===! VP-REF COMPLETE     T0768
!===! VERB-PHRASE COMPLETE     T0770
!===! ELLIPSIS     T0770
!---! VP-REF COMPLETE     T0771
!===! VP-REF COMPLETE     T0771
!===! VERB-PHRASE COMPLETE     T0773
!===! ELLIPSIS     T0773
!---! VP-REF COMPLETE     T0774
     !===! PUNCTUATION     T0776
!=========! IMPERATIVE     T0777
!=========! IMPERATIVE     T0779

          !===! PREPOSITION     T0785
               !===! NOUN     T0786
               !===! PROPER-NOUN     T0786
               !===! NOUN-PHRASE COMPLETE     T0788
               !===! NP-REF     T0790
          !=========! VERB-PHRASE COMPLETE     T0791
          !=========! ELLIPSIS     T0791
          !---------! VP-REF COMPLETE     T0792
          !=========! VP-REF COMPLETE     T0792
          !=========! VERB-PHRASE COMPLETE     T0794
          !=========! ELLIPSIS     T0794
          !---------! VP-REF COMPLETE     T0795
          !=========! VP-REF COMPLETE     T0795
                    !===! PUNCTUATION     T0797
          !==============! IMPERATIVE     T0798
          !==============! IMPERATIVE     T0800
!========================! IMPERATIVE     T0802

 RICH *PERIOD
           FOR  RICH *CR

  1    2    3    4    5
```

Figure 4-11: Chart built for a correction ellipsis

Dialogue charts are made possible by the inclusion of semantics and reference and antecedence relations within the chart, and by the ability to span individual utterances in working memory. There is much room for further work. Currently, only multiple-part elliptical utterances have been handled in this fashion. It may be possible to extend this technique to also deal with other kinds of dialogue structures [Hobbs 79, Grosz 86].

5. Strengths, weaknesses, and future directions

This paper has presented a technique for efficiently interpreting the most frequently occurring class of elliptical utterance in natural language interfaces. This case frame based ellipsis resolution technique handles a wide range of ellipsis cleanly and efficiently. There are forms of ellipsis that it clearly cannot handle, but the known techniques that attempt to address these forms are currently prohibitively expensive for use in actual interfaces.

The distinctions between these different classes have been drawn with the aid of a taxonomy of intersentential ellipsis, which was derived from a study of naturally occurring dialogues. This taxonomy clarifies the relationships between different classes of ellipsis and ellipsis resolution techniques, and explains their existance in terms of surface antecedence, coreference, and case frame matching. The taxonomy is complete for antecedent ellipsis. It seems also to apply to discourse relations between full sentences, although this has not been fully investigated.

The strengths of the Psli3 approach have also been presented, as exemplified by its handling of dialogue elliptical utterances. The semantic chart parser handles all types of ambiguity, including ambiguous semantics and references, with a simple, clear mechanism. This extension of chart parsing is made possible by the uniform use of rules for all information processing in production systems. It retains the capability of a production system to closely integrate a large number of independent processes, and uses the continuous context of working memory to bind together separate sentences and utterances, and the different processes involved in interpreting a dialogue. The programming style it encourages leads to intuitively pleasing designs. The use of dialogue charts extends the method to elegantly handling ambiguous dialogue relations, although how general this extension is is not yet clear.

In addition to the strengths that have been discussed, there are also several weaknesses that should be mentioned. Aside from needing extension in essentially all dimensions, a really usable database interface would need to forget old utterances at some point. Psli3 does not do this because it is not clear when a database interface should forget an utterance. The only practical problem caused by retaining all old utterances is that the system will gradually slow down as the size of working memory increases[29]. Given a criterion for when to forget, the structure to be deleted could be marked as inaccessible in a top-down fashion, and then the working memory elements could be safely deleted.

The system as implemented in OPS5 requires about a minute to interpret a typical sentence. This should not be compared to a parse time for a single syntax tree, of course, since it includes all parses produced, disambiguation, and semantic interpretation. But it is too slow for a usable system. This has not been a matter of concern, since this system has been used only for research. The easiest way to speed up the system would be to use a faster OPS5-like production system, such as the non-LISP based version of OPS5 or OPS83. These have the disadvantage, however, of lacking the LISP development and debugging environment.

A more serious practical problem is the low-level programming currently required in OPS5. While the production system architecture provides very elegant and powerful operations, currently one finds oneself programming at the equivalent of the assembly language level, specifying high-level objects and

[29] The speed of the original version of Psli3 degraded sharply as old utterances accumulated in working memory. The rules were rewritten using advice from Brownston et al. [Brownston 85] so that the decrease in system speed is no longer a problem.

operations in terms of very low-level data structures and actions. There is an obvious need for an ‘‘OPS5 compiler’’, which would automatically translate high-level descriptions of sets and linked-lists and operations on them into OPS5 objects and operations.

Returning to Psli3 itself, there are a number of interesting possibilities for further research. Perhaps the most interesting extension of the system would be to apply the discourse-functional taxonomy and dialogue chart technique to sequences of full sentences. As already noted, full sentences sometimes carry out discourse functions similar to those in our taxonomy, as in this full sentence example of elaboration:

John swung his arm over his head. He swung it right into the window.

Here the same action is described twice, with an additional case added in the second utterance. Note that the two utterances do refer to the same action, because they are compatible and are not marked as being disctinct. This example may seem a bit contrived, but the same principles underly the interpretation of the more natural sequence:

John swung his arm over his head. It went right into the window.

As discussed above, it would be very interesting to integrate into this system the handling of the non-antecedent classes from our taxonomy, using plan-based techniques [Carberry 89, Litman 87]. This could produce a system capable, in principle, of handling all types of ellipsis.

As also mentioned above, the system should be extended to handle some form of dialogue structures [Grosz 86, Litman 87]. Such an extension would allow the system to deal with digressions, where context switching can affect the antecedents available for ellipsis resolution:

Show me an irregular bone.
Make it a little larger.
Some flat bones.[30]

The mechanisms to support discourse structure are already available, in the form of semantic frame manipulation functions and the recording of discourse relations and speaker connections.

Another area with much room for development is the system’s use of syntax. This parser implements left-to-right incremental semantics with the capability of creating real syntactic structures, although the syntactic structures used at this point are quite simple. An earlier version handled several examples of more complex syntax, including relative clauses, using a technique similar to the derived categories of GPSG.

In addition to full sentence syntax, the system could use a better representation for the syntax of ellipsis. Elliptical utterances have a syntax of their own, not derivable in a general way from full sentence syntax [Shopen 72]. I was initially displeased with arbitrary restrictions in the ellipsis resolution rules, until I realized that they were primarily due to surface syntactic constraints. That these restrictions are a purely syntactic phenomenon can be seen by the fact that they differ in different languages. While in English a lone pronoun is almost always in the accusative case:

Who’s going to the store?
    **** I.***

[30] Example courtesy of an anonymous reviewer.

Me.

in German, the syntactic case appropriate to the case filled is used:

Wer geht zum Laden?
Ich.
**** Mich.***[31]

Some uses of ellipsis even differ between British and American English. So while the sentence:

It doesn't work. Never has done.

is completely normal in British English, an American would always ellide done[32].

As part of this, the ellipsis resolution rules should be separated into rules implementing the syntax of ellipsis and rules handling the case frame matching. This would simplify the system overall, since otherwise the cross-product of the two rule sets would be needed. This modularization would also be applied to rules building full sentence representations. It would be similar in nature to the already implemented separation of goal handling, semantic network access, and semantics rules from the actual parsing rules.

A final unexplored area is the use of meta-language utterances, that is, utterances referring to the current dialogue, such as:

I meant to say flat bones.

These utterances require the ability to resolve ellipsis and to resolve references to dialogue objects such as user input, system output, and their components and interpretations. This approach seems quite promising for such utterances, since all the relevant dialogue objects are already represented and readily acccessible in working memory.

References

[Allen 80] Allen, J. F. and Perrault, C. R.
Analyzing Intention in Utterances.
Artificial Intelligence 15(3):143-178, December, 1980.

[Brownston 85] Brownston, L., Farrell, R., Kant, E., and Martin, N.
Programming Expert Systems in OPS5.
Addison-Wesley, 1985.

[Burton 75] Burton, R.R.
Semantic Grammar: An Engineering Technique for Constructing Natural Language Understanding Systems.
Technical Report 3453, Bolt Beranek and Newman, 1975.

[Carberry 85] Carberry, S.

---

[31] This is a word-for-word translation of the English example.

[32] ***Done*** is of course a type of anaphora, so the British version is not purely ellipsis. This brings up the whole question of the relationship and interactions between ellipsis and anaphora, and whether they are really as different as is generally assumed, another vast area for future investigations.

A Pragmatics-Based Approach to Understanding Intersentential Ellipsis.
In Proceedings of the 23rd Annual Meeting of the ACL, pp. 188-197, 1985.

[Carberry 89] Carberry, S.
A Pragmatics-Based Approach to Ellipsis Resolution.
Computational Linguistics, Vol. 15, No. 2, pp. 75-96, 1989.

[Carbonell 83] Carbonell, J.G., and Hayes, P.J.
Recovery Strategies for Parsing Extragrammatical Language.
American Journal of Computational Linguistics, Vol. 9, Nos. 3-4, pp. 123-146, 1983.

[Cellio 84] Cellio, M.J., and Carbonell, J.G.
The PLUME™ Reference Manual, 1984.
Available from Carnegie Group Inc.

[Clocksin 81] Clocksin, W.F., and Mellish, C.S.
Programming in Prolog.
Heidelberg: Springer-Verlag, 1981.

[Earley 70] Earley, J.
An efficient context-free parsing algorithm.
Communications of the ACM, Vol. 13, No. 2, pp. 94-102, 1970.

[Fillmore 68] Fillmore, C.
The case for case.
In Universals in Linguistic Theory, E. Bach and R.T. Harms (eds.), pp. 1-90.
Chicago: Holt, Rinehart and Winston, 1968.

[Forgy 81] Forgy, C.L.
OPS5 User's Manual.
Technical Report CMU-CS-81-135, Carnegie-Mellon University, 1981.

[Frederking 85] Frederking, R.E.
Syntax and Semantics.
Technical Report CMU-CS-85-133, Carnegie-Mellon University, 1985.

[Frederking 88] Frederking, R.E.
Integrated Natural Language Dialogue.
Boston: Kluwer Academic, 1988.

[Grosz 77] Grosz, B.J.
The Representation and use of Focus in Dialogue Understanding.
Technical Report 151, Stanford Research Institute, July, 1977.

[Grosz 86] Grosz, B.J., and Sidner, C.L.
Attention, Intentions, and the Structure of Discourse.
Computational Linguistics, Vol. 12, No. 3, pp. 175-204, 1986.

[Hendrix 77] Hendrix, G., Sacerdoti, E., Sagalowicz, D., and Slocum, J.
Developing a Natural Language Interface to Complex Data.
Technical Report, SRI International, 1977.

[Hobbs 79] Hobbs, J.
Coherence and Co-reference.
Cognitive Science, Vol. 3, No. 1, pp. 67-82, 1979.

[Jardine 83] Jardine, T., and Shebs, S.

Knowledge Representation in Automated Mission Planning.
In AAIA Computers in Aerospace IV, pp. 9-14, American Institute of Aeronautics and Astronautics, 1983.

[Kay 67] Kay, M.
Experiments with a Powerful Parser.
In Proceedings of the 2nd International COLING, COLING, 1967.

[Kay 73] Kay, M.
The Mind System.
In Natural Language Processing, Rustin, R. (ed.), pp. 155-188.
New York: Algorithmics Press, 1973.

[Litman 87] Litman, D.J., and Allen, J.F.
A Plan Recognition Model for Subdialogues in Conversations.
Cognitive Science, Vol. 11, No. 2, pp. 163-200, 1987.

[Marcus 80] Marcus, M.P.
A Theory of Syntactic Recognition for Natural Language.
MIT Press, 1980.

[Schank 75] Schank, R.C. (ed.)
Conceptual Information Processing.
Amsterdam: North Holland, 1975.

[Shopen 72] Shopen, T.A.
A Generative Theory of Ellipsis.
PhD Thesis, UCLA, 1972.

[Touretzky 88] Touretzky, D.S. and Hinton, G.E.
A Distributed Connectionist Production System.
Cognitive Science, Vol. 12, No. 3, pp. 423-466, 1988.

[Vonnegut 70] Vonnegut, Kurt, Jr.
Welcome to the Monkey House.
Dell, 1970.

[Waltz 77] Waltz, D.L., and Goodman, A.B.
Writing a Natural Language Data Base System.
In Proceedings of the Fifth IJCAI, pp. 144-150, IJCAI, 1977.

[Wilensky 80] Wilensky, R., and Arens, Y.
PHRAN: A Knowledge-based Approach to Natural Language Analysis.
Technical Report UCB/ERL M80/34, University of California, Berkeley, 1980.

[Winograd 83] Winograd, T.
Language as a Cognitive Process, Vol. 1: Syntax.
Addison-Wesley, 1983.

Table of Contents

List of Figures

List of Tables