1 Introduction

Like all other scientists, linguists wish they were physicists. They dream of performing classic feats like dropping grapefruits off the Leaning Tower of Pisa, of stunning the world with pithy truths like $F = ma$ ... [But instead,] the modern linguist spends his or her time starring or unstarring terse unlikely sentences like "John, Bill, and Tom killed each other", which seethe with repressed frustration and are difficult to work into a conversation.
—Joseph D Becker[1]

When I took my first linguistics course, freshman transformational syntax, in 1974, we were taught that syntax was now basically under control. Sure, people still argued over particular transformations, and this was still all new and exciting stuff, but there was general agreement on the approach. Semantics, on the other hand, was a difficult and tenuous territory; no one yet understood what a semantic was. Semantics was said to have the same qualities as God or Mind—fun to argue about, but inherently unknowable. The study of semantics was left, therefore, until junior year.

Given linguists with attitudes like those toward semantics, it is not surprising that consumers of linguistic theory, such as researchers in natural language understanding, took semantic matters into their own hands. The result was approaches to semantics that were exemplary in their own terms but lacked a firm theoretical basis and hence were inadequate in their relationship to other aspects of language and to wider issues of meaning and representation of meaning. The best example of this is the dissertation of Woods (1967), which I will discuss in some detail in section 2.3.1.

But times are changing. Linguists are much braver than they used to be, and exciting new things are happening in the study of linguistic semantics. Probably the most important is Montague semantics (Montague 1973), which remained for several years a small and arcane area, but which has now attracted a large amount of interest. It is therefore time to start importing modern semantic theories into NLU and examining them to see how they can be applied, how they need to be adapted, and what their limitations are.

[1] Becker 1975: 70.

It is the goal of this work to do just this, with a particular emphasis on semantic interpretation. I will be using new approaches to semantics to help put NLU semantic interpretation onto a firmer and more theoretical foundation, and, in the framework thereby set up, to look at issues of lexical and structural disambiguation.

1.1 The problems

The problems discussed in this monograph can be divided into three distinct (but, of course, related) areas: semantic interpretation, word sense disambiguation, and structural disambiguation. In this section I explain each of these problem areas.

1.1.1 Semantic interpretation

By SEMANTIC INTERPRETATION we mean the process of mapping a syntactically analyzed text of natural language to a representation of its meaning. The input to a semantic interpreter is a parse tree, but we do not require that it represent a complete sentence; we allow well-formed subtrees such as noun phrases and even single words (labeled with their part of speech and syntactic features) as input. The output of a semantic interpreter is the meaning of the input text, or a suitable representation thereof. I will discuss in chapters 2 and 3 what such a representation might be; for the present, we will just observe that the natural language input text itself is a representation of its meaning, but not a "suitable" one.

I exclude from semantic interpretation all aspects of syntactic analysis; rather, I assume the existence of a parser that performs morphological and syntactic analysis upon an input text before it is passed to the semantic interpreter. This is not in any sense a logical necessity; systems have been built in which syntactic analysis and semantic interpretation have been completely integrated—*e.g.*, Riesbeck 1974, 1975; Riesbeck and Schank 1978; Hendrix 1977a, 1977b; Cater 1981, 1982. However, this approach becomes very messy when complex syntactic constructions are considered. Keeping syntactic and semantic analysis separate is well motivated merely by basic computer science principles of modularity. Moreover, it is my observation that those who argue for the integration of syntactic and semantic processing are usually disparaging the role of syntax,[2] a position that I reject (see

[2]"The theory of syntax is an artifact that cannot be used as a foundation for parsing; stereotypic patterns of lexical interactions cannot account for the highly particular nature of context and usage" (Small 1980: 12).

"Syntactic and semantic processing is *[sic]* done at the same time, with the primacy of semantics over syntax ... Syntax is used only when it helps in semantic analysis" (Gershman (1979: 11), describing Riesbeck's parser).

See also Schank and Birnbaum 1980.

also Charniak 1983b), and one which has been found to be unworkable[3] and, probably, psychologically unreal (Marslen-Wilson and Tyler 1980: 62–66; Tanenhaus and Donnenwerth-Nolan 1984) *(see section 2.4)*. This is not to say that parsing is possible without semantic help; in chapters 6 and 7 we will see many situations, such as prepositional phrase and relative clause attachment, in which the parser must call on semantic knowledge. In this book, I will show that syntax and semantics may work together well and yet remain distinct modules.[4]

Lytinen (1984) argues for a "compromise" position. It seems to me, however, that his position is much closer to the separation camp, in which he places the present work, than the integration camp. He states five principles regarding the interaction of syntax and semantics (1984: 4–5), and I am in agreement with about 4.3 of them:

1. Syntactic and semantic processing of text should proceed at the same time.

2. Syntactic decisions must be made with full access to semantic processing; that is, communication between syntax and semantics is high.

3. [Only] a limited amount of syntactic representation [need] be built during text understanding.

4. Knowledge about syntax and semantics is largely separate. Syntactic knowledge should be expressed in the parser's knowledge base as a largely separate body of knowledge, but this knowledge should have references to semantics, telling the system how semantic representations are built from these syntactic rules.

5. Semantics guides the parsing process, but relies on syntactic rules to make sure that it is making the right decisions.

---

[3]"The conclusion that must be drawn from [these experiments] is that if a semantic parser operates without a complete syntactic parse of its input, it is difficult, if not impossible, to prevent it finding readings which do not in fact exist" (Tait 1982, comparing the parsers of Boguraev 1979 and Cater 1981).

"It is fair to say that none of these 'semantically based' approaches has succeeded in producing anything like the clear, communicable framework that seems to be offered by a syntactic parser sitting together with a semantic interpreter. As a result, people are continuing to write new and better syntactic parsers, and more and more complete grammars to be used with them in two-part natural language processing systems. The advantages of modularity and portability to new application areas seem to outweigh any other arguments that there may be" (Mellish 1982a).

"[The] basic assumption [of Riesbeck's parser] that every part of the input sentence has to be specifically expected by some previously built structure does not always work" (Gershman 1979: 11). (Gershman adapted Riesbeck's parser by adding two new modes of processing, with much more syntactic knowledge.)

"The [not-much-syntax] models that have been presented to date ... are and will remain fundamentally inadequate to handle the range of grammatical phenomena well known and understood within the linguistics community for the last ten years" (Marcus 1984: 254).

See also chapter 4 of Lytinen 1984 for a critique of integrating syntax and semantics.

[4]Mellish (1982a) suggests the possibility of a system in which separate syntactic and semantic modules are automatically compiled into an efficient unified system. Hendler and Phillips (1981) suggest object-oriented computing to achieve this goal.

The system I will develop in this book is in full accord with principles 1, 2, and 5. I will not quite meet principle 4; rather, we shall see that the relationship between syntactic and semantic rules need not be quite so explicit. Principle 3 is usually true; the internal structure of a syntactic constituent at any level is almost never used once it and its semantic representation have been built, and it may be immediately discarded. There are, however, two good reasons for constructing a full parse tree anyway:

- **Programming:** It is easier to discard the whole tree at the end of the parse than do it piece by piece during the parse, and it is useful to retain the tree for purposes of debugging the program.
- **Theoretical:** Sentences in which apparently-closed constituents are re-opened are widely acceptable *(see section 6.2.5)*; the internal structure must be retained just in case.[5]

I also exclude from semantic interpretation any consideration of discourse pragmatics; rather, discourse pragmatics operates upon the output of the semantic interpreter. Thus, semantic interpretation does not include the resolution in context of anaphors or definite reference, or of deictic or indexical expressions, or the recognition and comprehension of speech acts, irony and sarcasm, metaphor, or other non-literal meanings.[6] These exclusions should not be thought of as uncontroversial; while few would advocate making speech-act interpretation part of semantic interpretation, Moore (1981) argues that definite reference resolution, as well as certain "local" pragmatic matters, must be resolved during semantic interpretation, and Plantinga (1986) argues that metaphor comprehension cannot be divorced from other aspects of language comprehension.

1.1.2 Word sense and case slot disambiguation

Many words of English have more than one meaning, and many quite common words have a very large number of meanings. Table 1.1 lists the words of English

[5]Moreover, there are sentences that require the surface form to be retained—for example those with SURFACE COUNT ANAPHORS (Hirst 1981a):

(i) When connecting the toe pin to the ankle rod, make sure that <u>the latter</u> goes underneath <u>the former</u>.

(ii) When the ankle rod is connected to the toe pin, make sure that <u>the former</u> goes underneath <u>the latter</u>.

and it may be that there are sentences that, similarly, refer into their structure. (I have not been able to find any examples.)

[6]For a discussion of anaphors, definite reference, and their resolution in context, see Hirst 1981a, 1981b. Deictics and indexicals are discussed by Fillmore 1975, Kaplan 1978, 1979, and Levinson 1983. Some useful starting points for reading about the role of speech acts in language are Searle 1969, Cole and Morgan 1975, Boyd and Ferrara 1979, and Levinson 1983. For work in AI on recognizing and understanding speech acts, see Allen 1979, 1983a, 1983b, Allen and Perrault 1980, Perrault and Allen 1980, Brown 1979, 1980.

Table 1.1. *Words with the greatest number of senses in the* Merriam-Webster Pocket Dictionary *(data from Amsler 1980:55–57)*

WORD	CATEGORY	NO. OF SENSES	WORD	CATEGORY	NO. OF SENSES
go	verb	63	take	verb	24
fall	verb	35	dead	adj	21
run	verb	35	good	adj	21
turn	verb	31	have	verb	21
way	noun	31	line	noun	21
work	verb	31	pass	verb	21
do	verb	30	touch	verb	21
draw	verb	30	dry	adj	20
play	verb	29	wing	noun	20
get	verb	26	draft	noun	19
form	noun	24	give	verb	19
make	verb	24	turn	noun	19
strike	verb	24			

that Amsler (1980, 1981, 1982a) found to have the greatest number of senses listed in *The Merriam-Webster pocket dictionary.* Any practical NLU system must be able to disambiguate words with multiple meanings, and the method used to do this must necessarily work with the methods of semantic interpretation and knowledge representation used in the system.

There are three types of lexical ambiguity: POLYSEMY, HOMONYMY, and CATEGORIAL AMBIGUITY. Polysemous words are those whose several meanings are related to one another. For example, the verb *open* has many senses concerning unfolding, expanding, revealing, moving to an open position, making openings in, and so on. Conversely, homonymous words have meanings with no relationship one to another.[7] For example, *bark* means both **the noise a dog makes** and **the stuff on the outside of a tree**. A word may be both polysemous and homonymous; the adjective *right* has several senses concerning correctness and righteousness, but also senses concerning the right-hand side.[8] There is no clear line between

[7]The terminology in this area can be a little confusing. Strictly speaking, since we are interested in written language, the homonymous words we are concerned with are HOMOGRAPHS, that is words where many meanings are associated with the same lexeme, though different meanings may have different pronunciations. For example, the vowel varies in *row* depending on whether it means **a line of objects** or **a commotion**, but this fact is of no consequence when dealing with written language. If we were concerned with speech recognition, the type of homonyms we would worry about would be HOMOPHONES—words that are pronounced the same but possibly spelled differently, such as *four* and *fore*. A HETERONYM is a non-homophonic homograph (Drury 1983).

[8]A common etymology does not preclude the senses being distinct enough to be considered homony-

polysemy, homonymy, and metaphor; today's metaphor may be tomorrow's polysemy or homonymy. For example, there is an obvious relationship between *mouth* in the sense of **a person's mouth** and in the sense of **the mouth of a river**, but for practical purposes they are quite separate concepts, and it is not clear into which category *mouth* should therefore be placed.

Categorially ambiguous words are those whose syntactic category may vary. For example, *sink* can be a noun describing a **plumbing fixture** or a verb meaning **become submerged**. Clearly, categorial ambiguity is orthogonal to the other types: the ambiguity of *respect* is categorial and polysemous, as its noun and verb meanings are related, but that of *sink* is categorial and homonymous, as its noun and verb meanings are not related. Categorial ambiguity is mainly a problem in parsing, and I will say no more about it in this monograph, except where it interacts with other types of ambiguity. (See Milne 1980, 1986 for a discussion of handling categorial ambiguity in a deterministic parser.)

Generally, verbs tend to polysemy while nouns tend to homonymy, though of course there are many homonymous verbs and polysemous nouns.[9] This is consistent with the suggestion of Gentner (1981a, 1981b) that verbs are more "adjustable" than nouns; that is, nouns tend to refer to fixed entities, while verb meanings are easily adjusted to fit the context, with frequent adjustments becoming lexicalized as new but related senses of the original verb.

Panman (1982) argues that although experiments, including his own, have shown that people's intuitions do distinguish between polysemy and homonymy, it is difficult and probably unnecessary to maintain the distinction at the level of linguistic theory. While it seems strange that a cognitively real linguistic phenomenon should have no place in linguistic theory, I too will make little use of it in this work. The semantic objects we will be using are discrete entities,[10] and if a word maps to more than one such entity, it will generally (but not always) be a matter of indifference how closely related those two entities are.

For an NLU system to be able to disambiguate words,[11] it is necessary that it use both the discourse context in which the word occurs and local cues within the sentence itself. In this book, I discuss how this may best be done in conjunction with my approach to semantic interpretation, although the techniques will not be limited to use solely within my approach.

mous in everyday modern usage.

[9]In general, adjectives show less ambiguity than nouns and verbs, and this is reflected in Table 1.1.

[10]There are those who would argue that this fact immediately damns the whole approach. But there are no well-developed models yet for any form of non-discrete semantics in AI, though current research in fine-grained connectionist systems may change this.

[11]It is not always the case that an NLU system need worry about disambiguation at all; in some applications, such as machine translation, it is acceptable to ask the user for help (Tomita 1984) or simply preserve the ambiguity in the system's output (Hirst 1981a[1]:68, fn 11; 1981b:90, fn 10; Pericliev 1984).

Ideally, an NLU system should be able to go beyond polysemy and into metaphor. In the most general case, this is an extremely difficult task, and I do not attempt it in this research. For a discussion of metaphor in NLU, see Russell 1976, Wilks 1977, Browse 1978, Hobbs 1979, Carbonell 1981, and Plantinga 1986.[12]

A problem closely related to lexical disambiguation is case slot disambiguation. Case theories of language are generally associated with the work of Fillmore (1968, 1977). In its most basic form, case theory views a sentence as an assertion whose predicate is denoted by the verb of the sentence and whose arguments are denoted by the noun phrases. For example:

(1-1) Nadia tickled Ross with a feather.

Here, *Nadia* is the AGENT of the verb *tickle*, and we say that *Nadia* FILLS THE SLOT of the AGENT CASE. Similarly, *Ross* fills the the PATIENT slot, and *a feather* is in the INSTRUMENT case. We say that the INSTRUMENT case is FLAGGED by the preposition *with*; the AGENT and PATIENT cases are flagged by subject and object position respectively.

There is no rigid one-to-one mapping between flags and cases, however; that is, case flags are not unambiguous. For example, *with* can also flag the cases MANNER[13] and ACCOMPANIER:

(1-2) Nadia tickled Ross <u>with</u> glee.

(1-3) Ross flew to Casablanca <u>with</u> Nadia.

Also, a case may have more than one flag, often varying with different verbs. For example, some verbs allow the INSTRUMENT in the subject position when no AGENT is specified:

(1-4) <u>The feather</u> tickled Ross.

Thus, different verbs take different cases and different flag-to-case mappings; however, there is still a great degree of regularity in case systems that we will be able to use.

This explanation of cases is greatly simplified, and a few extra points should be made. First, not all prepositional phrases are case-flags and fillers; PPs can qualify nouns as well as verbs. Second, an adverb can act as a combined case-flag and filler:

(1-5) Nadia tickled Ross <u>gleefully</u>.

[12]Also of interest here is the work of Granger (1977; Granger, Staros, Taylor and Yoshii 1983) on determining the meaning of an unknown word from context. If it has been determined that a particular instance of a word does not accord with its normal usage, techniques such as Granger's may be applied. Metaphor, of course, provides more semantic constraints than are available in the general case of a hapax legomenon.

[13]Strictly speaking, MANNER is not a case at all but a verb modifier; see footnote 15.

In this example, *gleefully* behaves exactly as *with glee* does in (1-2).[14] Third, subordinate clauses also exhibit case behavior, with the conjunction as the flag and the sentence as the filler:

(1-6) Because Ross couldn't bring himself to touch the geranium, Nadia put it in an old shoe box for him.

The word *because* flags the REASON case here. Fourth, there are good linguistic reasons (*e.g.*, Bresnan 1982b) for distinguishing between cases and certain VERB-MODIFYING PPs that describe such things as the time or place at which an action occurs:

(1-7) Nadia tickled Ross on Friday at the Art Museum.

In the present research, this distinction will not in general be necessary, and we will usually be able to treat all verb-attached PPs in the same way.[15]

Clearly, the assignment of a case to a flag depends on the meaning of the potential slot-filler; we know that in (1-4) *the feather* is not in the AGENT case because

[14]In English, we can think of the suffix *-ly* as a flag for the MANNER case. However, English morphology is not quite regular enough to permit a general morphological analysis of case-flags in adverbs; rather, we just think of both the flag and the case-filler being bundled up together in the word's meaning.

[15]An example of the linguistic distinction between case-fillers and verb modifiers is that PP verb modifiers are sensitive to adverb movement and may be put at the start of the sentence, while PP case-fillers must usually follow the verb unless topicalized. Thus it sounds strange to say (i) instead of (ii):

(i) *On the boat, Ross put his luggage.

(ii) Ross put his luggage on the boat.

where *on the boat* is in the LOCATION case, but one can say both (iii) and (iv):

(iii) On the boat, Ross was having fun.

(iv) Ross was having fun on the boat.

where *on the boat* is a PLACE verb qualifier. Also, modifiers may sometimes not come between the verb and true case-fillers. Thus one can say (v) but not (vi):

(v) Ross put his luggage on the boat on Tuesday.

(vi) *Ross put his luggage on Tuesday on the boat.

But:

(vii) Ross threatened Nadia with a wrench in the park.

(viii) Ross threatened Nadia in the park with a wrench.

(Barbara Brunson, personal communication). The most important difference, however, is the BIUNIQUENESS CONDITION (Bresnan 1982b): each case may appear at most once in a sentence, while there is no restriction on how many times each modifier type may appear. Thus one may not have two INSTRUMENT cases; if more than one INSTRUMENT has to be specified, conjunction must be used with a single flag; examples from Bresnan 1982b:

(ix) *Ross escaped from prison with dynamite with a machine gun.

(x) Ross escaped from prison with dynamite and a machine gun.

On the other hand, (xi) (also from Bresnan 1982b) contains three MANNERs, three TIMEs and two PLACEs:

AGENTs must be conscious animate entities. Thus the problem of determining which case slot a particular preposition or syntactic position flags is very similar to that of lexical disambiguation: in each, semantic information is necessary to decide which one of a set of meanings is to be assigned to a particular token. In the present research, I will show how the two tasks may indeed be accomplished by very similar mechanisms.

1.1.3 Syntactic disambiguation

Although many sentences of English have more than one parse,[16] there is usually a unique preferred parse for a sentence after semantics and discourse context are considered. For example, in (1-8):

(1-8) Nadia left the university on the wrong bus.

we do not take *the university on the wrong bus* as a single noun phrase; rather, we apply the knowledge that universities seldom ride buses. That is, there is a SEMANTIC BIAS to one of the parses. Bias may also come from context; the parse of a sentence such as (1-9):

(1-9) They're cooking apples.

depends on whether it answers the question *What are they doing in the kitchen?* or *What kind of apples are those?*

In addition, the English language often exhibits certain preferences—SYNTACTIC BIASES—in choosing among several possible parses. Thus (1-10) was judged silly by informants:

(1-10) The landlord painted all the walls with cracks.[17]

(xi) Ross deftly handed a toy to the baby by reaching behind his back over lunch at noon in a restaurant last Sunday in the Back Bay without interrupting the discussion.

Nevertheless, there are certain restrictions on using the same flag twice in the same way, even for modifiers; thus *with* can't be used twice in the same sentence for MANNER:

(xii) *Ross chaired the meeting with tact with dignity.

(xiii) Ross chaired the meeting with tact and dignity.

The apparent exception (xiv) may be explained as an elliptical form of (xv):

(xiv) Ross chaired the meeting with tact and with dignity.

(xv) Ross chaired the meeting with tact and Ross chaired the meeting with dignity.

See also Somers 1984 and Brunson 1986a.

[16]Church and Patil (1982) point out that some sentences can have parses numbering in the hundreds if semantic constraints are not considered.

[17]From Rayner, Carlson, and Frazier (1983), who took it to be semantically well-formed.

who generally said that though the intent was clear, it sounded like either the walls were painted with a crack-like pattern or that cracks were being used to paint the walls, readings that both have the prepositional phrase attached to the verb phrase of the sentence instead of to the object noun phrase. Similarly, (1-11):

(1-11) Ross baked the cake in the freezer.

was taken by informants to mean that the baking in some bizarre way took place in the freezer, rather than that the particular cake known to have once been in the freezer was baked in a conventional manner.[18] PPs starting with *by* often sound like passives even when a locative reading makes more sense:

(1-12) SHOESHINE BY ESCALATOR[19]

(1-13) Ross was told what to do by the river.

Sentence (1-12) seems to be saying that the escalator shines one's shoes, and (1-13) sounds like Ross heard voices in the running water; even though the meaning in which Ross received instructions from an unspecified person on the river bank makes more sense, the parse that treats *the river* as the deep-structure subject is still preferred.[20] The following are reported by Cutler (1982b) as "slips of the ear":

(1-14) You never actually see a forklift truck, let alone person.
(Perceiver attempted to access a compound noun, forklift person, *as if a second occurrence of* forklift *had been deleted.)*

(1-15) The result was recently replicated by someone at the University of Minnesota in children.
(Perceiver assigned NP status to the University of Minnesota in children, *cf.* the University of California in Berkeley.*)*

Cutler attributes such errors to the hearer; I am inclined to say, rather, that the error was the speaker's in creating a sentence whose structure misled the speaker. The main point, however, is that the speaker WAS misled into an anomalous interpretation consistent with the sentence structure, despite the availability of a sensible interpretation "close by".

The source of syntactic bias is disputed. Frazier (1978; Frazier and JD Fodor 1978) has suggested two principles for the preferred placement of a constituent whose role is ambiguous:

[18] Sentence (1-11) and its test on informants is due to Barbara Brunson.

[19] Sign at American Airlines terminal, LaGuardia airport, New York, November 1984.

[20] Marcus (1980: 228–234) argues that, at least in some cases, when syntactic and semantic biases conflict and neither is strong enough to override the other, the sentence is judged ill-formed. The argument is based on subtleties of well-formedness that vary widely over idiolects, and I am not convinced of the generality of the hypothesis.

- Right Association (also called Low Right Attachment, Late Closure, or Local Association): A new constituent is attached as low and as far to the right in the parse tree as possible.[21]
- Minimal Attachment: A new constituent is attached to the parse tree using as few non-terminal nodes as possible.

These principles predict many of the syntactic biases of English; Frazier (1978:115; Frazier and Fodor 1978) shows that they are inherent consequences of a two-stage parsing model she presents, and Milne (1982a) has shown them to be a natural consequence of Marcus parsing *(see section 1.3.2)*. However, the principles sometimes conflict, or interact in complex ways. In cases such as prepositional phrase attachment, when both a noun phrase and its dominating verb phrase could receive the PP, Low Right Attachment suggests that the NP (the lowest, right-most node) should take it, while Minimal Attachment prefers the VP because NP attachment allegedly requires an extra NP node above the resulting complex NP.[22;23] Sentence (1-10) shows that common sense is not always used to resolve the conflict, and Ford, Bresnan, and Kaplan (1982) and Crain and Steedman (1985) have proposed a different set of principles, which I will discuss in detail in sections 6.3.3 and 6.3.4.

Many sentences that are structurally unambiguous are, however, LOCALLY AMBIGUOUS: they contain a point at which, in left-to-right parsing, the parser could take one of several paths, and the information that determines which is correct occurs only later in the sentence. In the case of parsers with limited lookahead, such as Marcus parsers *(see section 1.3.2)*, the disambiguating information may be out of sight and a choice may have to be made without it. If this choice is wrong, the parser will eventually find itself off in entirely the wrong direction, unable to find any correct parse. A sentence that can do this to a parser is said to be a SYNTACTIC GARDEN-PATH SENTENCE,[24] in that it leads the parser "down the garden path"; the unfortunate parser is said to have been "GARDEN-PATHED". Many well-formed sentences are garden paths for the human mental parsing mechanism:

(1-16) The horse raced past the barn fell.[25]

---

[21]This principle was first suggested by Kimball (1973), and was modified by Frazier and Fodor; see Fodor and Frazier 1980 for discussion of the differences.

[22]This crucially assumes that noun phrases with PP modifiers are parsed as in (i) rather than (ii):

(i) $[_{NP}[_{NP}$ the noun phrase$]$ $[_{PP}$ with $[_{NP}$ the prepositional phrase$]]]$

(ii) $[_{NP}$ the noun phrase $[_{PP}$ with $[_{NP}$ the prepositional phrase$]]]$

Analysis (i) strikes me as dubious, and I use (ii) below.

[23]In Frazier's (1978) two-stage model, Minimal Attachment only occurs (in the second stage) if Right Association (a consequence of the first stage) fails to happen.

[24]In section 4.3.2, we will see that there are also semantic garden-path sentences.

[25]From Bever 1970:316.

(1-17) The old dog the footsteps of the young.[26]

Most people have trouble with these sentences the first time they see them.[27] Marcus (1980) argues that it is no shame for a computational parser to be garden-pathed by sentences that also trip humans up, and that such behavior is in fact necessary if claims of cognitive modeling are to be made for the parser. I also take this viewpoint.

To find which parse is the one preferred in each particular case, a parser needs help from both world knowledge and discourse context, as well as knowledge about preferred attachments. In this research, I develop a method of providing such semantic information for a parser—a method that works in concert with the semantic interpretation and lexical disambiguation systems that I also develop.

1.2 Frames

The concept that unifies the approaches to the problems described in the previous section is that of the FRAME as a semantic object. I am using the word *frame* in the conventional AI sense: a data item that contains a collection of knowledge about a stereotyped topic (Charniak 1976, 1981a), or represents a concept. A frame is usually structured as a set of SLOTS or ROLES that may contain VALUES; often, a DEFAULT VALUE obtains for a slot if no other value is specified. A value may be almost any type of object: a number, a boolean value, another frame (or frame instance—see below), or a homogeneous set of such values. A slot may be marked with restrictions on what sort of values it allows.

Here are some examples. The first set, shown in figure 1.1, is based (loosely) on the frames in Wong 1981b for the task of reading a menu in a restaurant. (The formalism is that for the Frail frame system—*see section 1.3.1.*) The first frame defined is `read`. Its second line says that it is a ("ISA") particular type of the `task` frame, and as a consequence INHERITS all the properties of that frame. Thus, since `tasks` already have a slot for the `agent` performing the task, it is unnecessary to define the slot again in `read`. On the other hand, `object` is defined at this level of the FRAME HIERARCHY (or ISA HIERARCHY), because it is not true that all `tasks` have this slot. It is also specified that the filler of this slot must be something that can be read, namely `reading-material`. The `facts` clauses, not shown in detail here, describe the actions involved in reading, starting off with taking the item to be read.

[26] I believe this example is due to Yorick Wilks.

[27] Their meanings are, respectively:

(i) The horse—the one that was raced past the barn—fell.

(ii) The footsteps of the young are dogged by the old people.

```
   frame:  read
     isa:  task
   slots:  object (required) (reading-material)
   facts:  (take ?agent ?object)
               ...

   frame:  possibilities-list
     isa:  reading-material
   slots:  type-of-possibilities

   frame:  menu
     isa:  possibilities-list
   facts:  (type-of-possibilities food)

instance:  menu34
     isa:  menu
```

Figure 1.1. Frail frames describing some of the knowledge necessary to understand the concept of reading a restaurant menu (from Wong 1981b).

The next frames define `possibilities-list`, `menu`, and also a particular INSTANCE of a `menu`, namely one given the arbitrary name `menu34`. Because `menu34` has `reading-material` as an ancestor in the hierarchy, it will be allowed to fill the `object` slot in any instance of the `read` frame. An instance is necessarily a leaf in the frame hierarchy, and no other frame may be defined in terms of it. Nodes that are not instances are said to be GENERIC.

The second example set, in figure 1.2, is from KRL (DG Bobrow and Winograd 1977, 1979). (I have modified it slightly for this exposition.) First the `BusinessTravel` and `Visit` frames are defined, and then an item that is simultaneously an instance of both is given. The `SELF` clause gives the frame's parent in the frame hierarchy, from which properties may be inherited. The following clauses list slots, giving their names and restrictions on their fillers; for example, the `visitor` slot can only be filled by an instance of the `Person` frame. `Event137` is an instance of the `Visit` frame, and particular values for the slots of `Visit` are given. It is also an instance of `BusinessTravel`; note that since a value is not given for the `mode` slot, it will take on the default value `Plane`. Because a KRL frame can be an instance of more than one frame, the frame hierarchy of KRL is a network, not just a tree as in Frail.

Thus we can think of a generic frame as a representation of a concept and an instance as one specific occurrence of the concept. Slots can be thought of as arguments or parameters of the concept, so that the filled slots of an instance describe or qualify that instance.

Many different frame systems have been developed in AI—another important one not mentioned above is KL-ONE (Brachman 1978)—and there is still much

```
[BusinessTravel UNIT
  ⟨SELF (an Event)⟩
  ⟨mode (OR Plane Auto Bus) DEFAULT Plane⟩
  ⟨destination (a City)⟩]

[Visit UNIT
  ⟨SELF (a SocialInteraction)⟩
  ⟨visitor (a Person)⟩
  ⟨visitees (SetOf (a Person))⟩]

[Event137 UNIT Individual
  ⟨SELF {
    (a Visit with
       visitor = Rusty
       visitees = (Items Danny Terry))
    (a BusinessTravel with
       destination = SanFrancisco) }⟩]
```

Figure 1.2. An example of frames in KRL, describing a trip from two different perspectives (from DG Bobrow and Winograd 1977).

controversy over how frames may best be organized and what their exact semantics should be (*e.g.* Brachman 1982, Brachman and Levesque 1982). Although the details of the present approach are tied to the Frail frame system (to be described in section 1.3.1), we try as far as possible to keep the basic ideas independent of any particular frame formalism.

The power of frame systems, the power that we will exploit in dealing with the problems described in the previous section, lies not merely in having a large hierarchy or network of frames but in being able to manipulate those frames and perform inferences upon them, prove theorems about them, do inductive, deductive, and abductive reasoning with them. A frame system is not just a static knowledge base containing a collection of pieces of information, but includes powerful procedures for using that information. Thus, when we say that we are using the concept of a frame as a semantic object, we are not simply defining a data structure; rather, we are adopting a whole system for representing, storing, retrieving, and using knowledge.

By using frames as semantic objects, we will be able to take the following approach to the problems under discussion:

- We will use a frame system as a well-defined semantics for natural language. By having a strong correspondence between lexical categories and elements of the frame system, we will be able to construct an elegant and compositional semantic interpreter.
- Because of this strong correspondence, well-formed subtrees of the input text will be mapped to well-defined items in the frame system, so whenever the parser

needs semantic help in dealing with a subtree, it can draw upon the full power of the frame system for assistance.

- Similarly, because individual words also correspond to frame system entities, lexical disambiguation can use the knowledge of the frame system. By constructing the semantic interpreter so that it deals, whenever possible, with semantic objects by reference only to their type rather than their content, lexical disambiguation may proceed independently of semantic interpretation and structural disambiguation and yet supply word meanings to the other processes when necessary.

1.3 Artificial intelligence research at Brown University

The work described in this book began as a component of the artificial intelligence research in the Department of Computer Science, Brown University.[28] This section describes that work briefly, so that the reader may better understand the present work and its place in the overall project.

The top-level goal of the project is to study the use of common-sense knowledge in both language comprehension and problem solving. To this end, a knowledge representation suitable for both these tasks has been developed (Charniak 1981a). This representation incorporates features of both frame systems and predicate calculus; it forms the basis of the Frail frame language (Wong 1981a, 1981b).

Frail is used by Bruin (Wong 1981a, 1981b), a system that can both understand simple stories, making inferences from the story when necessary, and solve problems. Here are three examples that Bruin can handle:

(1-18) A manufacturer used a molding machine in producing TV cabinets. The standard time for production was 0.025 hours per TV cabinet. The molding machine operated for 80 hours per week. The manufacturer delivers 2000 cabinets per week. The standard time for a setup change for the molding machine was 6.5 hours per lot. The setup change cost 3.50 dollars per hour. Storage of the TV cabinets cost 0.115 dollars per cabinet per week. Calculate the economic lot-size for inventory control of the TV cabinets.

(1-19) There is a green block on a table. A red block is on the table. The red block has a blue block on it. Put the red block on the green block while putting the green block on the blue block.

(1-20) Jack took a menu. He ordered a hamburger. He ate. He paid. He left.
What did Jack eat? What did Jack do before paying? Did Jack read the menu? Why did Jack take the menu? What did Jack pay for?

Bruin contained a parser based on that of Marcus (1980) and a primitive semantic interpreter based on that of Woods (1967, 1968) to translate the English input into

[28]Eugene Charniak, Principal Investigator. Others who have contributed to the project are Michael Gavin, Tom Grossi, Jim Hendler, Doug Wong, and the present author.

assertions or commands in Frail.[29] A module called Pragmatics monitored the input to the Frail knowledge base and made inferences about likely consequences, on the basis of simple world knowledge. In (1-20), the assertion would be made, from a script about restaurants, that what Jack ate was the hamburger he ordered. A problem solver named NASL, based on that of Drew McDermott (1978), could answer questions, either by looking directly for an assertion that provided the answer, as in the case of finding out what Jack ate, or by using problem-solving techniques to find a solution, as in the TV cabinet economic lot-size problem.

This system is presently being redesigned and extended. A new version of Frail has been developed (Charniak, Gavin, and Hendler 1983), and work is proceeding in such areas as context recognition and discovering causal connections (Charniak 1981b, 1982) and problem solving (Hendler 1985, 1986a). Improvements to the natural language front-end include a new parser, Paragram (Charniak 1983a), and the work described in this volume. Since this work makes extensive use of Frail and Paragram, it is important that the reader understand their design and intent. In the next two subsections I describe each in turn.

1.3.1 The Frail frame language

Frail is a frame language that incorporates features of first-order predicate calculus. This section describes the most recent version, Frail 2.1 (Charniak, Gavin, and Hendler 1983). Frail is based on the deductive information retriever of Charniak, Riesbeck, and McDermott (1980: 140–161); an earlier implementation is described by Wong (1981a, 1981b); and the motivation for its design is described by Charniak (1981a).

Frail consists of a knowledge base of "true" statements, together with functions `assert`, `erase`, and `retrieve` to add, delete, and retrieve statements. The function `retrieve` takes a statement with variables in it—one may think of it as a pattern to be matched—and returns all assignments of values to those variables that make the statement "true" in the knowledge base. The truth of a statement to be retrieved may be determined by its literal presence in the knowledge base or may be INFERRED using BACKWARD-CHAINING RULES. These rules, also asserted to the knowledge base, tell Frail about valid inferences; for example, the rule (1-21) says "Nadia likes anything that is warm and fuzzy", *i.e.*, "One way to prove that Nadia likes something is to prove that it is warm and fuzzy":

```
(1-21)   (←(likes Nadia ?x)
                (and (warm ?x) (fuzzy ?x)))
```

Frail also permits the assertion of frame information. One may assert that something is a frame or is an instance of a frame, or that a frame has a particular set of slots, each with a particular restriction on its filler. Since frame statements are

[29]The parser was implemented by Tom Grossi and the interpreter by Graeme Hirst.

```
[frame:  purchase                             The action of buying
   isa:  action                                is a kind of action.
 slots:  buyer (person)                       The buyer and seller
         seller (person)                       must both be people.
         bought-item                          No restrictions on what is sold.
         money (negotiable-instrument)        Cash, check, or credit card.
  acts:  pay-step                             The components of the action.
         (give buyer money seller)            Paying for the item.
         get-item-step
         (give seller bought-item             Getting the item.
             buyer)]

[instance:  purchase34                        A particular purchase action.
            (purchase
              (buyer Nadia)                   The slot fillers.
              (seller Ross)
              (bought-item marmoset34)
              (money twenty-dollars)]
```

Figure 1.3. A frame and an instance in Frail 2.1 describing (in very simple terms) the action of purchasing. Italics indicate comments. (Based on an example from Charniak, Gavin, and Hendler 1983.)

frequent in Frail, it provides a special syntax for their definition; an example may be seen in figure 1.3.

In the work described in this book, I make extensive use of FRAME DETERMINERS. Although frame determiners are not a facility in Frail 2.1, but are instead programmed on top of it, I use them as if they were part of the frame system (as they may indeed be in future versions), so it is appropriate to discuss them here. A frame determiner is a function that, like `assert` and `retrieve`, adds or gets formulas from the database but, unlike those basic functions, takes into account notions of CONTEXT and FOCUS. The focus is basically the set of entities that the discourse has referred to recently—see Hirst 1981a, 1981b for a more precise characterization—and is where the referents of pronouns and other definite references are sought.

The two main frame determiners are `the` and `a`, and their semantics are what their names suggest. The `the` function is like `retrieve`, except that before searching the database it first looks in the focus for a matching instance that is ACTIVE in context (Hirst 1981a, 1981b). (It is an error if there is not either exactly one match in focus, or none in focus but exactly one in the database.) This implies that `the` has available to it a representation of a discourse focus and that there is a process that dynamically maintains this focus.[30] The `a` function `asserts` the

[30]In the present implementation, there is only a very primitive model of discourse context to maintain

existence of a new frame instance (and returns as a value its name), but allows the possibility that it may turn out to be identical to a pre-existing instance; in particular, if there are frame instances of the same type in focus, `a` will assume that its argument is identical to one of them.

For example, suppose the following are in focus, each an instance of the frame suggested by its name:

(1-22) `penguin34, penguin87, catch22, kiss88`

Then the call `(the ?x (kiss ?x))` will return `kiss88`; the call `(a ?x (penguin ?x))` will return a new instance:

```
(1-23)  [instance:  penguin99
         one-of:   (penguin34 penguin87)]
```
[31]

and `(a ?x (restaurant ?x))` will return a new `restaurant` instance with no restrictions on its being identical with other restaurant instances in the knowledge base. In each case, the instance returned will be added to the focus, or, if already there, be marked as again very recent.

The frame determiner `question` is used in questions. It is very close to a simple knowledge-base retrieval function, looking for an instance that matches its argument and not considering focus at all. It returns a list of matching instances, and if its argument included any free variables, it also gives the bindings of those variables for each instance it found.

Other frame determiners include `some` and `the-pl`, which are the plural forms of `a` and `the`; each deals with a set of instances instead of a single instance.

1.3.2 The Paragram parser

The Paragram parser (Charniak 1983a) is a deterministic parser with limited lookahead. That is, once it has made a decision about the structure of the input or the lexical category of a word, it is unable to change that decision. Further, it is restricted in how far beyond the point of current processing it is allowed to look for information that might help its decisions. In these respects, it is modeled on Marcus's well-known parser Parsifal (Marcus 1980, Sampson 1983).

A Marcus-type parser uses a set of grammar rules, an input buffer of limited length, and a work stack. Each grammar rule is a pattern–action pair. When the input buffer matches the pattern of a rule, that rule is executed. An example of a pattern is (1-24):

this focus (and even that is not yet operative), since this is not the primary thrust of the present work. Ideally, the focus should allow for references to items only implicitly in context—if a car is in focus, then any of its parts, for example, may also be referenced. Later versions may take into account focus and topic determination. For a discussion of the issues involved and some approaches, see Hirst 1981a, 1981b or Grosz 1977a, 1977b, 1978, 1981. This approach has mild psychological reality; *cf.* Guindon 1985.

[31]Frail does not yet permit implementation of this construction; *cf.* section 3.8.

(1-24) `[= np] [=* to] [= tnsless]`

This pattern matches a buffer whose first element is a noun phrase, whose second is the word *to*, and whose third is marked as being tenseless. Rules are organized into PACKETS, and only rules whose packet is presently ACTIVE are considered. In addition, rules may be assigned PRIORITIES; if more than one rule matches, the one with highest priority is the one chosen.

The actions in a rule may include taking items out of the buffer and putting them on the stack, adding them to a partial parse tree that is on the stack, marking them with certain syntactic features, and so on. Rules may also activate and deactivate rule packets (including their own). As items are taken from the buffer to the stack or put back from the stack into the buffer, words from the input flow into or out of the buffer, so that it always contains the same number of items. An exception to this occurs during the processing of noun phrases, when the parser is allowed to look further ahead in the input than at other times.

Marcus parsers, being deterministic, stand in contrast to AUGMENTED TRANSITION NETWORK (ATN) parsers (Woods 1970, Bates 1978, Johnson 1983, Charniak and McDermott 1985). An ATN parser takes its grammar as a directed network whose nodes are states and whose arcs are labeled with tests that must be satisfied for their traversal and with actions that are performed if the arc is traversed. The parser itself is non-deterministic; if more than one arc may be traversed from a given state with a given input, it chooses one at random, and if it subsequently finds itself in a state from which it cannot proceed, it backs up and makes a different choice.

Although Paragram is a Marcus parser, there are several differences between it and Parsifal. First, although deterministic, Paragram is not "strictly" deterministic in the sense that Parsifal is; that is, unlike Parsifal, it is allowed in certain special cases to modify the parse tree that it is building. One of these is sentences that require lowering, such as (1-25):

(1-25) Ross believes Nadia to have invented the three-ring binder.

in which the object, *Nadia*, is lowered into the verb complement sentence so that it can be parsed as (1-26):

(1-26) Ross believes that Nadia invented the three-ring binder.

The second difference is in the type of grammar rules that each parser uses. Both parsers have grammars that are based quite closely on traditional standard transformational syntax (Chomsky 1965). However, Parsifal does not distinguish base-structure rules from transformational rules, but rather requires both types to be expressed in a single formalism (which is essentially a reverse transformation). On the other hand, like standard transformational grammars, Paragram maintains the distinction between the two types of rule, greatly simplifying the grammar. Unfortunately, the separation is not as clean as one would like, and the grammar writer cannot be blind to the fact that each base rule is actually converted by the grammar interpreter into a set of the other type of rule.

Third, Paragram can parse ill-formed sentences. If no rule pattern can be found that exactly matches the input buffer, Paragram attempts to find the nearest match, and proceeds from there. In the same situation, Parsifal simply gives up. I do not explicitly make use of this feature of Paragram in this work.

In using Paragram for the work described herein, it was necessary to modify its grammar somewhat. These changes will be described in the appropriate sections.

1.4 Policy statements

In addition to the biases mentioned elsewhere (such as my anti-anti-syntax position; *see section 1.1.1*), two other attitudes that pervade this book should be made explicit:

1. Psychological reality. The primary aim of the research is to build an artificial intelligence system; if the system performs as intended, then it has achieved its goal, regardless of whether or not any claims can be made for the system's being a model of human cognitive mechanisms. However, it is often a good strategy in AI to consider cognitive modeling anyway; finding out how people do something and trying to copy them is a good way to get a program to do the same thing (*cf.* Ringle 1983). Moreover, claims that can be made about the psychological reality of an AI program's mechanisms are interesting in their own right, even if they are not central to the research. For these reasons, therefore, we will where appropriate look at psycholinguistic research on how people deal with the language-understanding problems that we are trying to handle computationally.

2. Artificial difficulties. It is necessary in artificial intelligence to distinguish between genuine hard problems and artificial ones. A common form of argument in AI against the adequacy of a procedure or rule is to show that it cannot handle some particular hard case. Often, however, the cases cited are pathological ones that are not true counterexamples. For example, one can construct sentences with many horrible interacting ambiguities that will probably defy resolution by the mechanisms developed in this research. However, such sentences usually turn out to be "artificial"—that is, they are not sentences that would ever turn up in real, considerate discourse, and if, by accident, they did, they would probably defy resolution by the human sentence-understanding mechanism as well.

In general, people often misunderstand hard ambiguities. Sometimes their resolution mechanism notices the problem and applies conscious error-recovery procedures that we will not consider further herein. Sometimes the problem is not detected, and the mechanism confidently returns an answer that is wrong, or an answer that is correct but is so by luck rather than by science.[32] It would be wrong

[32]For examples of garden-path ambiguities that fool people, see section 4.3.2.

to expect an AI system to do better.[33] It is therefore necessary to be very careful when considering counterexamples to make sure that they are "genuine". At present, our only tool for this is our sharp, unbiased intuition.

[33]The reader familiar with it should note that my point here is NOT simply Wilks's (1975e) argument that "there will always be a counterexample for any AI system". For a discussion of Wilks's view, see Hirst 1976.