3 The Absity semantic interpreter

A great interpreter ought not to need interpretation.
—John Morley[1]

3.1 Introduction

In this chapter, I describe the Absity semantic interpreter.[2] Absity meets five of the six requirements for an interpreter listed in section 2.5, and provides a foundation for further research in meeting the remaining requirement.

Absity is part of the artificial intelligence research project at Brown University that was described in section 1.3. It uses one of the project's parsers, Paragram *(see section 1.3.1)*, and the project's frame representation language, Frail *(section 1.3.2)*. The implementation to be described is therefore necessarily dependent upon the nature of these other components, as are many aspects of Absity's design. Nevertheless, in keeping with the goals of this work, the design has been kept as independent as possible of the representation formalism and the parser. The main ideas in Absity should be usable with other representations that have a suitable notion of semantic object and also, in particular, with other parsers, transformational or otherwise.

The organization of this chapter is as follows. In the first half, Absity is gradually built up, by explaining alternately a strategy and then its use in Absity. I then give some examples and some implementation details. In the second half, Absity is put on trial, and its strengths and weaknesses are evaluated.

3.2 Two strategies: Strong typing and tandem processing

In the design of Absity, we will make use of two features of Montague's formalism *(see section 2.2.2)*: a strong typing of semantic objects, and running syntax and semantics not just in parallel but in tandem. These strategies will allow us to simplify the system of semantic rules.

[1] MORLEY, John [Viscount Morley of Blackburn]. *Emerson*. New York: Macmillan, 1884.

[2] The name is short for "A Better Semantic Interpreter Than Yours".

By a typing of objects, you will recall, we meant that there are a number of different kinds of object, and each object is of one kind or another—that is, every object has a type. Further, there are some basic, or primitive, types of object, and other types of object are constructed from them; from these, yet other types of object may be constructed, and so on recursively. For example, let *X* and *Y* be object types. Then any new object constructed from an object of type *X* and one of type *Y* would be of a new type (call it *X+Y*), as would all objects so constructed. It need not be the case that any two types can combine to form another; there may be restrictions on what can combine with what. Clearly, conventional syntactic categories form an object typing like this. Montague also imposed a similar typing upon semantic objects in his formalism, and, in particular, a typing that was in one-to-one correspondence with the typing of the syntactic objects. So, all elements of any given syntactic type are interpreted as corresponding to objects of the same semantic type, and if two syntactic types combined to form a third, then the corresponding semantic types would combine to form the semantic type corresponding to the third. We also will adopt this procedure.

This makes the tandem operation of syntactic and semantic processing very easy. We can now, like Montague, also put our syntactic and semantic construction rules in one-to-one correspondence, so that when the parser constructs a new constituent with some particular syntactic rule, the corresponding semantic rule can be invoked to make the object that will serve as that constituent's representation, in the manner described above. Note, however, that we cannot simply have one semantic rule for EACH syntactic rule, as Montague did. Montague's syntax was very simple, and each of its rules was in fact a CONSTRUCTION RULE. But we cannot assume that this property is true of parsers in general—indeed, it couldn't be true of any but the simplest parser. We therefore need to determine which of the parser's rules or actions are construction rules and therefore require a corresponding semantic construction rule. In the grammar of the Paragram parser, the set of construction rules turns out to be exactly the set of base rules. This should not be surprising; in many transformational theories of grammar, including at least one of Chomsky's (1965: 132), base rules are those that determine the deep structure and the sentence's meaning, while transformations can only make changes to the syntactic structure that do not affect the meaning.[3]

By adopting these two strategies, we have implicitly satisfied the fourth of the requirements in section 2.5, that semantic processing be able to provide feedback to the parser. By having the semantic interpreter proceed in parallel with the parser and by ensuring that the representation of a partially interpreted sentence is always a well-formed semantic object, we ensure that the fullest possible amount of semantic information is always available for the parser to use.

[3]Similarly, in her synthesis of transformational syntax with Montague semantics, Partee (1973, 1975) observes that the semantic rule corresponding to many transformations will simply be the identity mapping.

Table 3.1. *Types in the Frail frame language*

BASIC TYPES	
Frame	`(penguin ?x)`, `(love ?x)`
Slot	`color`, `agent`
Frame determiner	`(the ?x)`, `(a ?x)`
OTHER TYPES	
Slot–filler pair = slot + frame statement	`(color=red)`, `(agent=(the ?x (fish ?x)))`
Frame descriptor = frame + slot–filler pair*	`(penguin ?x (owner=Nadia))`, `(love ?x (agent=Ross) (patient=Nadia))`, `(dog ?x)`
Frame statement or instance = frame determiner + frame descriptor	`(the ?x (penguin ?x (owner=Nadia)))`, `(a ?x (love ?x (agent=Ross) (patient=Nadia)))`, `(the ?x (dog ?x))`, `penguin87` [an instance]

3.3 The typing system of Absity

In the previous section, we decided upon the use of a strong typing on the semantic objects. I now introduce such a typing on the elements of the Frail language, and a correspondence between the typing and the lexical categories of English.

The basic elements of Frail are shown in table 3.1, with examples of each. The three basic types are (GENERIC) FRAME, SLOT, and FRAME DETERMINER FUNCTION.[4] The notation for a frame is a list with the frame name and a free variable; the variable is denoted by a question-mark prefix. Thus `(penguin ?x)` is the `penguin` frame (which may or may not have something to do with penguins, depending on whether the frame name, which is only an arbitrary symbol, has mnemonic significance for the human observer). A slot is denoted simply by its symbolic name—*e.g.*, `agent`. The notation for a frame determiner is a list with the function name and a free variable—for example, `(the ?x)`. Do not let the notational sim-

[4]If you don't find these self-explanatory, see sections 1.2 and 1.3.1.

ilarity of frames and frame determiners confound you; the two are very different kinds of object—a frame is a data structure, and a frame determiner is a function.

The types that are built from these basic types are SLOT–FILLER PAIR, FRAME DESCRIPTOR, and INSTANCE or FRAME STATEMENT. A slot–filler pair (sometimes *sf-pair* for short) consists of the name of a slot and a value to fill that slot.[5] The value will be a frame statement or instance (to be defined in a moment). The notation for a slot–filler pair is a list of the slot and the value, with an equals sign between them: `(color=red)`.

A frame descriptor is a complete description of a generic frame. It can be used by frame determiner functions when searching the knowledge base for a frame to match that description, and when creating a new instance of a frame of that description. A frame descriptor is composed of a frame with zero or more slot–filler pairs added; thus the frame `(penguin ?x)` is also a frame descriptor, and so is `(penguin ?x (color=red))`. Note that the variable in the frame is still unbound.

A frame statement is a complete knowledge-base access statement in Frail. It is formed by giving a frame descriptor as an argument to a frame determiner function; for example, `(the ?x)` plus `(penguin ?x (color=red))` gives (3-1):

(3-1) `(the ?y (penguin ?y (color=red)))`

When the two are combined, the free variable in the frame descriptor is bound to the variable in the frame determiner, for which a unique new name is automatically generated. (For simplicity of exposition, I will assume in most of the discussion that variable names are magically correct from the start.) A frame statement can be EVALUATED by Frail, and the result is a FRAME INSTANCE—the unique symbolic name of the particular instantiation of the frame described by the frame descriptor that the frame determiner function caused to be inserted into or retrieved from the knowledge base. For example, the result of evaluating (3-1) might be the instance `penguin87`, one of several instances of the `penguin` frame that the system happens to know about.[6] Because a frame statement can always be turned into an instance by evaluation, the type system of Absity does not distinguish the two, and one may appear wherever the other is allowed.

The correspondence between these semantic types and the syntactic categories of English is shown in table 3.2. This correspondence satisfies the requirements imposed by our application of rules in tandem. For example, a preposition corre-

[5] For those not up on typographic subtleties: A slot-filler (spelled with a hyphen) is that which fills a slot; a slot–filler (spelled with an en dash) is the combination of a slot and a slot-filler.

[6] For mnemonic purposes, instances created by Absity as it operates are given names formed by adding digits to the end of the frame name. Instances predefined in the knowledge base, on the other hand, may have been given names even more descriptive. Thus `red` is an instance of the `visible-color` frame, and in other circumstances might have been called, say, `visible-color34`.

Table 3.2. *Type correspondences in Absity*

SYNTACTIC TYPE	SEMANTIC TYPE
Sentence	Frame statement, instance
Sentence body	Frame descriptor
Proper noun phrase	Frame statement, instance
Pronoun	Frame statement, instance
Common noun	Frame
Adjective	Slot–filler pair
Determiner	Frame determiner
Noun phrase	Frame statement, instance
Preposition	Slot name
Prepositional phrase	Slot–filler pair
Subordinate conjunction	Slot name
Subordinate clause	Slot–filler pair
Verb	(Action) frame
Adverb	Slot–filler pair
Auxiliary	Slot–filler pair
Verb phrase	Frame descriptor
Clause end	Frame determiner

sponds to a slot, a noun phrase to a frame statement, and their product, a prepositional phrase, corresponds to a slot–filler pair, as required.

Clearly, table 3.2 makes some very strong claims about the nature of representation systems and language—that a noun in some sense IS a frame, a preposition IS a slot, and so on. The starting point for these claims is Charniak's paper "The case–slot identity theory" (1981c), which argues for the plausibility (and meaningfulness) of the theory that the senses of a verb may be identified with the generic frames of a frame representation, their cases with the slots of the frames, and the noun phrases that fill cases with things that can fill frame slots.[7] Many of the type correspondences of table 3.2 follow immediately from this. It is already asserted that a verb is a frame; if a case is a slot, then a case-flagging preposition must be a slot name; a case-filling noun phrase must be a frame instance; and a prepositional phrase must be a slot–filler pair, as must an adverb. Sentences acting as

[7] If you are unfamiliar with case theory, see section 1.1.2.

noun phrases can also fill cases, as in (3-2):

(3-2) Ross knew that Nadia sought revenge for the trout incident.

so they too must be frame instances—in particular, a sentence will be an instance of the frame that represents the verb of the sentence. Similarly, subordinate conjunctions—those such as *because* and *when* that conjoin subordinate clauses with main clauses—can be slots, so that they can combine with sentences to make subordinate clauses slot–filler pairs.[8]

The other types mostly follow as a corollary of the above. If a noun phrase is an instance, then the head noun must be a frame; noun modifiers—adjectives and relative clauses—must be things that qualify a frame; and the noun phrase's determiner must be something to instantiate a frame, *i.e.*, a frame determiner function. It seems reasonable that the type of a descriptive adjective may be the simplest kind of qualifier, namely a slot–filler pair; a prepositional phrase that qualifies a noun may also be a slot–filler pair, just like other PPs. A relative clause, however, must be of the same type as other sentences, an instance, or else our strong typing is ruined. This is not a problem, however, if we allow the combination of an instance with another instance to which it refers. For example, this noun phrase:

(3-3) the grey cat that Ross loves

becomes these two instances, one for the matrix NP and one for the relative clause:

```
(3-4)   (the ?x (cat ?x (color=grey)))
        and
        (a ?y (love ?y (agent=Ross) (patient=WH)))
```

where `WH` represents the relative pronoun *that*, a pointer to the first frame instance. These may then be combined into this Frail representation for the complete NP:

```
(3-5)   (the ?x (cat ?x (color=grey)
                    (a ?y (love ?y (agent=Ross)
                                   (patient=?x)))))
```

which asserts that the `cat` sought has two properties: that its `color` is `grey`, and that there is a `love` frame whose `agent` is `Ross` in which it participates as `patient`.

Proper noun phrases, such as *Dr Henry Kissinger*, correspond directly to an instance and therefore need no frame determiner. The same is true of pronouns. On the other hand, all common noun phrases will need a determiner, to turn a frame descriptor into an instance. We therefore make the reasonable assumption that all common NPs do have a determiner, even though it may be null; that is, the null determiner has an associated type and semantics. To see that this is reasonable, we need only consider these sentences (where ∅ marks a null determiner):

(3-6) Ross likes the cats.

[8] I do not treat COORDINATE conjunctions, such as *and* and *or*, in Absity.

(3-7) Ross likes ∅ cats.

(3-8) Ross likes all cats.

As we can see, ∅ in English often has much the same meaning as *all*. Now, strong typing requires ∅ *cats* to have the same type as the corresponding NP of the other sentences, a frame instance. But if ∅ contributed nothing, then the word *cats* would also be a frame instance, identical in all three sentences, implying that *the* and *all* also contribute nothing, an unacceptable conclusion. In the next section I will show how null words can be handled in Absity.

Verb qualifiers, both auxiliaries and adverbs, can be slot–filler pairs on the frame of the verb; their behavior is just like that of adjectives with nouns. A verb phrase is thus a frame descriptor, so we need to find a frame determiner somewhere to turn it into an instance when its construction is complete.[9] A convenient way to do this is to construe the end of the VP itself as corresponding to the frame determiner; that is, the clause end is to be to a verb phrase what a determiner is to a noun phrase. How this may be done will be shown in the next section.

3.4 Another strategy: Turning meaningful syntax into words

In section 3.3, I said that case-flagging prepositions correspond to Frail slots. But cases can also be flagged by syntactic position—by the fact of the filler being in the subject, object, or indirect object position of the sentence. Clearly, Absity needs to have semantic objects—slots—that correspond to these syntactic case flags. Up till now, however, we have tacitly assumed that semantic objects corresponded only to syntactic objects;[10] after all, most of a sentence's meaning comes from the words of the sentence and their grouping into syntactic components, and it seems natural, therefore, to build the meaning of a sentence by taking the individual word meanings and assembling them according to the dictates of syntactic structure. Including syntactic position seems to work against the goal of compositionality—the meaning of the whole would be more than a function of the meaning of the parts. Further, it seems to require us to complicate the semantic rules so that they can deal with the contribution of syntactic position, and thus appears to threaten the strong typing and tandem processing.

There is, however, an easy way around the problem—PRETEND that syntactic positions are words just like the real words of the sentence. In fact, we will carry out the pretense to the extent that we will require the parser to INSERT these pretend words into the sentence, and Absity to then treat them as if they had been in there all along. We will use three such words: *SUBJ*, *OBJ*, and *INDOBJ*, corresponding to

[9]I have implicitly equated verb phrases with clause bodies in this statement; the justification for this will become clear in the next section.

[10]The fact of being in subject position is not a syntactic object in the sense that a prepositional phrase is.

the syntactic positions of subject, object, and indirect object. Further, since they are case flags, we will deem our new words to be prepositions; we will sometimes call them PSEUDO-PREPOSITIONS to distinguish them from "natural" prepositions.

So, for example, a sentence like (3-9):

(3-9) Nadia gave Ross a pewter centipede for his birthday, because she knew that he didn't have one already.

will become, and will be parsed as, (3-10):

(3-10) SUBJ Nadia gave INDOBJ Ross OBJ a pewter centipede for his birthday, because SUBJ she knew OBJ that SUBJ he didn't have OBJ one already.

The insertion of pseudo-prepositions requires us to modify our syntax for a simple sentence slightly. Previously, the base rules of the Paragram parser included these conventional rules:

(3-11) $S \rightarrow$ NP AUX VP
$VP \rightarrow$ V [NP] [NP] PP*

The NPs in these rules are, of course, the subject, object, and indirect object. The new rules are these:[11]

(3-12) $S \rightarrow$ PP AUX VP
$VP \rightarrow$ V PP*

In the case of Paragram, the base rules of the grammar may be amended directly as shown, and it is straightforward to modify the transformational rules so that they insert pseudo-prepositions at the appropriate places.

The principle of inserting pretend words can also be used in two other cases. The first is the need to treat null determiners as "real". For this, we can just get the parser to insert the PSEUDO-DETERMINER *NULLDET* into any common noun phrase that needs it. Similarly, the VP-DETERMINER *CLEND* (for *clause-end*) can be added by the parser; the sentence punctuation itself serves as a clause-end marker for a major sentence.[12] Collectively, we call our new lexemes PSEUDO-WORDS.

[11]We retain the notion of a VP in the syntactic structure in order to handle verb complements and predicates.

[12]Paragram and Parsifal (Marcus 1980) also treat punctuation marks as syntactic objects, and so does Hausser (1984) in his extension of Montague semantics, so this approach is not unprecedented or unorthodox.

He hath found the meaning.
—William Shakespeare[13]

3.5 The semantic rules of Absity

The semantic rules of Absity follow immediately from the results of the preceding sections. Tandem processing means that each rule will simply combine two or more semantic objects to make a new one. What the objects are will be given by the corresponding syntactic rule, their types will be as given by table 3.2, and they will be combined as defined in table 3.1.

Some examples will make our semantic interpreter clearer. First, let's consider a simple noun phrase, *the book*. From table 3.2, the semantic type for the determiner *the* is a frame determiner function, in this case `(the ?x)`, and the type for the noun *book* is a kind of frame, here `(book ?x)`. These are combined in the canonical way shown in table 3.1—the frame name is added as an argument to the frame determiner function—and the result, `(the ?x (book ?x))`, is a Frail frame statement (which evaluates to an instance) that represents the unique book referred to.[14]

Next, consider *the red book*. A descriptive adjective corresponds to a slot–filler pair; so, for example, *red* is represented by `(color=red)`, where `color` is the name of a slot and `red` is a frame instance, the name of a frame. A slot–filler pair can be added as an argument to a frame, so *the red book* would have the semantic interpretation `(the ?x (book ?x (color=red)))`.

Now let's consider a complete sentence:

(3-13) Nadia bought the book from a store in the mall.

Table 3.3 shows the representation for each component of the sentence; note that in the table the basic noun phrases have already been formed in the manner described above. Also, we have inserted the pseudo-prepositional subject and object markers *SUBJ* and *OBJ*, and represent the clause end with a period. For simplicity, we assume that each word is unambiguous (disambiguation procedures are discussed in chapter 5); we also ignore the tense of the verb. Table 3.4 shows the next four stages in the interpretation. First, noun phrases and their prepositions are combined into prepositional phrases; their semantic objects form slot–filler pairs. Then, second, the prepositional phrase *in the mall* can be attached to *a store* (since a noun phrase, being a frame, can have a slot–filler pair added to it), and the prepositional phrase *from a store in the mall* is formed. The third stage shown in the table is the attachment of the slot–filler pairs for the three top-level prepositional phrases to the frame representing the verb. Finally, the period, which is translated as a frame

[13]SHAKESPEARE, William. *Pericles, Prince of Tyre.* 1608. I, i, 143.

[14]Recall that it is the responsibility of Frail to determine, with the help of the discourse focus, which one of the books that it may know about is the correct one in context *(sections 1.3.1 and 3.6)*.

Table 3.3. *Absity example*

WORD OR PHRASE	SEMANTIC OBJECT
SUBJ	`agent`
Nadia	`(the ?x (person ?x (propername="Nadia")))`
bought	`(buy ?x)`
OBJ	`patient`
the book	`(the ?y (book ?y))`
from	`source`
a store	`(a ?z (store ?z))`
in	`location`
the mall	`(the ?w (mall ?w))`
. [period]	`(a ?u)`

determiner function, causes instantiation of the `buy` frame, and the translation is complete.

The next examples show how Absity translates *yes/no* and *wh-* questions. We will use interrogative forms of the previous example:

(3-14) Did Nadia buy the book from a store in the mall?

(3-15) What did Nadia buy from a store in the mall?

Sentence (3-14) has almost the same parse as the previous example, and hence almost the same translation. The only difference is that the frame determiner for the clause is now `(question ?x)`, the translation of *?*, instead of `(a ?x)`; thus the translation is (3-16):

```
(3-16)  (question ?u
          (buy ?u
            (agent=(the ?x (person ?x
              (propername="Nadia"))))
            (patient=(the ?y (book ?y)))
            (source=(a ?z (store ?z
              (location=(the ?w (mall ?w)))))))
```

In a complete NLU system, it would be the responsibility of the language generator to take the result of this Frail call, which will be either `nil` or the matching instance or instances, and turn it into a suitable English reply, such as *No, she didn't.*[15]

[15]If the answer is negative, a helpful system might then proceed to look for an instance that is a near

Table 3.4. *Absity example (continued)*

SUBJ Nadia

```
(agent=(the ?x
  (person ?x (propername="Nadia"))))
```

OBJ the book

```
(patient=(the ?y (book ?y)))
```

in the mall

```
(location=(the ?w (mall ?w)))
```

---

a store in the mall

```
(a ?z (store ?z
  (location=(the ?w (mall ?w)))))
```

from a store in the mall

```
(source=(a ?z (store ?z
  (location=(the ?w (mall ?w))))))
```

Nadia bought the book from a store in the mall

```
(buy ?u
  (agent=(the ?x (person ?x
    (propername="Nadia"))))
  (patient=(the ?y (book ?y)))
  (source=(a ?z (store ?z
    (location=(the ?w (mall ?w)))))))
```

Nadia bought the book from a store in the mall.

```
(a ?u
  (buy ?u
    (agent=(the ?x (person ?x
      (propername="Nadia"))))
    (patient=(the ?y (book ?y)))
    (source=(a ?z (store ?z
      (location=(the ?w (mall ?w))))))))
```

match, and present that to the user:

(i) No, she didn't, but Ross did.

(The need to find near matches for a frame also occurs in disambiguation, and is discussed in sections 7.2.4 and 7.3.2.) If the answer is positive, a good reply generator would try to determine the salient part of the matching instance—often that which matched the indefinite `(a ?x)` in the query—to give a reply of the form:

(ii) Yes, at Bobby's Booktique.

For the *wh-* question, (3-15), we make use of the fact that `(question ?x)` returns the bindings of its free variables. The translation will be (3-17):

```
(3-17)    (question ?u
              (buy ?u
                  (agent=(the ?x (person ?x
                      (propername="Nadia"))))
                  (patient=?WH)
                  (source=(a ?z (store ?z
                      (location=(the ?w (mall ?w)))))))
```

Notice that *what* has been translated as the free variable `?WH`. As before, the call will return either `nil` or an instance list, but in the latter case the list will include the bindings found for `?WH`, *i.e.*, the book that Nadia bought. The reply generator would use the bindings to compose a suitable answer.

These examples have assumed that the parser has provided all the right information whenever it was needed. This assumption, however, is unrealistic; we must allow for the fact that many parsers (all parsers?), including Paragram, will not necessarily do the syntactic operations in the "correct" order for Absity. Consider, for example, the noun phrase *the cute rabbit*, to be parsed by this Paragram base rule:

(3-18) NP → DET ADJ N

The lexicon lists the semantic objects corresponding to the three words of the NP as follows:

(3-19)	the =	`(the ?x)`	*(frame determiner)*
	cute =	`(appearance=cute)`	*(slot–filler pair)*
	rabbit =	`(rabbit ?x)`	*(frame name)*

Now, strictly speaking, our approach requires that we combine these three in one operation, the result being a frame statement:

(3-20) `(the ?x (rabbit ?x (appearance=cute)))`

In fact, Paragram will not execute (3-18) as a unitary operation, but will proceed left to right, first pulling in the determiner, then the adjective, and then the noun; intermediate structures will be created in the process and transformational rules (without semantic counterparts) may possibly apply.

We therefore must add some new semantic types to Absity, types that will only be used "internally", whose necessity will depend upon the parser. For example, the partial noun phrase *the cute* will have the type:

(3-21) *frame-statement/frame-name*

which can be thought of as "a thing that will become a frame statement when combined with a frame name". (The notation is intended to suggest that of generalized phrase structure grammars (Gazdar 1982) or categorial grammars, in which *A*=*B* denotes an *A* that is missing its *B*.) The rule for the creation of this type would be this:

(3-22) *frame-statement/frame-name = frame-determiner + sf-pair*

The exact representation of these SLASH TYPES is unimportant, and a list of components is almost enough; thus *the cute* could be represented by (3-23):

```
(3-23)    [(the ?x) (appearance=cute)]
```

This isn't quite adequate, because there could be several sf-pairs, for several adjectives, as in *the cute brown*; thus we really need a list of lists:

```
(3-24)    [(the ?x) [(appearance=cute) (color=brown)]]
```

Note that this also implies another type formation rule, saying that we can keep adding new sf-pairs to a frame-statement/frame-name:

(3-25) *frame-statement/frame-name = frame-statement/frame-name + sf-pair*

Other slash types may be created as a given parser necessitates it.

A lewd interpreter? But come, I'll tell thee all my whole device.
—William Shakespeare[16]

3.6 Implementation details

The careful reader will have noticed that although there has been no direct contradiction, the Frail expressions that I have shown above have been rather different in form from those that I showed in section 1.3.1. In fact, what I have shown above is not really Frail at all, but a meta-notation for Frail that is translated into real Frail by a simple translator. There are two reasons for doing this:

- Frame determiners, as I said in section 1.3.1, are not part of Frail, but sit on top of it. Most of Absity's Frail calls are actually calls on the frame determiners.
- The syntax of Frail is messy and less compositional than its semantics. The meta-notation smooths over some of the cracks.

Ignoring, for the moment, considerations of focus, we may regard `(a ?x)` as asserting an instance and `(the ?x)` as retrieving one. We then have the following equivalences:

```
(3-26)    (a ?x (marmot ?x (owner=Nadia)))
          [instance:  marmot24
                     (marmot
                         (owner Nadia))][17]
```

[16]SHAKESPEARE, William. *The merchant of Venice*. 1596. III, iv, 80–81.

[17]In fact, this is also a meta-notation, provided in this case by Frail, for its even messier lowest-level representation:

```
(i)    (assert '(inst marmot24 marmot))
       (assert '(:= '(owner marmot24) Nadia))
```

where `marmot24` is a name that has been automatically generated, and:

```
(3-27)  (the ?x (marmot ?x (owner=Nadia)))
        (retrieve
          '(and (marmot ?x)
                (owner ?x Nadia)))
```

These equivalences are still not exact, since `instance:` returns nothing useful, while `(a ?x)` returns the automatically generated instance name; similarly, `retrieve` returns a binding list such as `((?x marmot24))`, while `(the ?x)` will return just the instance name.

Let's now consider focus. Frame determiners have in the present version a very simple idea of focus: it is simply a list of "current" instances *(see section 1.3.1)*. They operate as follows: The function `(the ?x)` translates its argument, as shown above, and sends the retrieval request to Frail. When the results come back, it checks each binding to see if that instance is in the focus list; it expects to find exactly one that is, and complains if it doesn't. The function `(a ?x)` first performs a retrieval in the same manner as `(the ?x)`. It then compares the bindings with the focus list and asserts a new instance; if some of the bindings were in focus, it restricts the instance to being one of these.[18] The plural frame determiners `(the-pl ?x)` and `(some ?x)` (not yet implemented)[19] are similar. The differences are that `(the-pl ?x)` expects either to find more than one match in focus or to match a set in focus; `(some ?x)` also prefers a set.

To avoid having to have lambda binding and lambda conversion in the system, Absity uses a simpler, back-door method to bind one noun phrase to another.[20] This mechanism is presently used in sentences that require equi-NP-insertion. For example:

(3-28) An obnoxious multi-national corporation wants to hire Nadia.

Sentence (3-28) is usually assumed to be derived from a form like (3-29):

(3-29) An obnoxious multi-national corporation$_i$ wants [an obnoxious multi-national corporation$_i$ hire Nadia].

The subscript on the NPs indicates their referential identity, and the equi-NP-deletion rule deletes the second occurrence, giving (3-28). Parsing the sentence requires the reverse operation, equi-NP-insertion, making a copy of the sentence subject to serve as the subject of the embedded sentence. This copy is BOUND to the original, to indicate referential identity of the pair.

[18]Since Frail can't do this, the function does it instead, keeping a note of the fact tucked away.

[19]Frail cannot yet handle plural determiners; see point 4 of section 3.8.

[20]DS Warren (1983) points out that first-order representations are not powerful enough for NLU, but higher-order representations seem to be too powerful. This back-door mechanism may be considered a nice compromise between the two. It may, however, prove to be insufficiently powerful for future developments in Absity, and lambdas may have to be introduced.

In Absity, this is done by keeping a table of the interpretations of all the NPs in the sentence. The Frail expression under construction does not contain the actual NP interpretations, but only pointers to entries in the table. Thus, two NPs that are bound to one another are represented by pointers to the same table entry. Pointers are, of course, replaced by the table entry when the surface form of the sentence interpretation is printed.

Our interpreter does it well.
—William Shakespeare[21]

3.7 Absity as the fulfillment of our dreams

In section 2.5 I listed in six categories the various qualities desired in a semantic interpreter. It is now time to pass judgment upon Absity with respect to these qualities. We will find that the desiderata are numbered so that Absity meets five of the six.

1. Compositionality. Absity is nothing if not compositional. Its semantic rules do little more than combine objects to make new ones, and have no power to ignore or modify semantic objects. In fact, as we shall see in section 3.8 when discussing its shortcomings, Absity is, if anything, a little too compositional.

2. Semantic objects. The frames of Frail have been suitable semantic objects, as predicted.

3. Not ad hoc. The rules of Absity meet our requirement that they be clean and general and not mangle semantic objects (*cf.* point 1 above). By using pseudo-words, Absity also allows the rules to be sensitive to the contributions to meaning of syntax.

4. Feedback for the parser. Absity is able to provide feedback by running in parallel—a fortiori, in tandem—with the parser, with its partial results always being well-formed semantic objects. I will show in chapter 7 how this property may be used in structural disambiguation.

5. Lexical ambiguity. I will show in chapter 5 how Absity supports a lexical disambiguation procedure.

6. Semantic complexities. This is the requirement that is as yet unfilled by Absity, which, as a new system, has not been developed to the point of perfection. In the next section, I describe some of the complexities of semantics that Absity can't handle and show that the prospects for overcoming these defects are good.

[21]SHAKESPEARE, William. *All's well that ends well.* 1602. IV, iii, 209.

3.8 What Absity can't do yet

Although it has the virtues listed in the previous section, Absity still falls short of being The Answer To The Semantics Problem. It does, however, provide a base upon which A More Nearly Perfect Semantic Interpreter can be built, and this is why I refer to it (Hirst 1983a) as a "foundation for semantic interpretation".[22] In this section, I discuss the ways in which Absity is not satisfactory and how its inadequacies may one day be cured.

Sometimes I will put some of the blame for Absity's deficiencies on Frail (and frame representations of the current state of the art in general). Frail, you'll recall from section 1.3.1, is an extensional first-order representation (Charniak, Gavin, and Hendler 1983). As such, it is inadequate for the representation of all of the concepts that natural languages such as English can express, for which a higher-order representation seems necessary (DS Warren 1983). (One such representation is the higher-order modal temporal intensional logic used by Montague (1973), which I mentioned briefly in section 2.2.2.) Absity, therefore, occasionally finds itself embarrassed by not having a suitable representation available to it for certain constructs of English. However, in some cases even if Frail were to be magically improved, it would not be straightforward to amend Absity to take advantage of it.

1. Intensions and opaque contexts. Intensions, obviously, are the first item in the list of things that cannot be represented in a purely extensional formalism.[23] For example:

(3-30) Nadia talked about unicorns.

This sentence is ambiguous: it has an extensional de re meaning, in which it says that there are some particular unicorns about which Nadia talked:

(3-31) "I met Belinda and Kennapod, the unicorns that live behind the laundromat, as I was walking home tonight," said Nadia.

and an intensional de dicto meaning, in which it simply says that unicorns were the topic of discussion:

(3-32) "I wonder whether I shall ever meet a unicorn," Nadia mused.

It cannot be inferred from the de dicto reading that any unicorns exist at all, only that the idea or intension of a unicorn does. This distinction is lost upon Frail, which could only represent the extensional meaning, making the inference that there was a particular set of unicorns of which Nadia spoke. A similar problem occurs with (3-33), a sentence used by Bruin *(see section 1.3)*:

(3-33) Ross ordered a hamburger.

[22]Hence also its earlier name, Cement Semantics.

[23]Although intensions CAN be represented in a first-order system; see McCarthy 1979.

The usual reading of this sentence is intensional; Ross is willing to accept anything that meets his criteria of hamburgerness. He has not ordered any particular hamburger, which would be the extensional reading, illustrated by (3-34):

(3-34) "Please bring me that hamburger that's up there, the third from the right on the second shelf," said Ross.

Nevertheless, Bruin required that (3-33) be translated extensionally, namely as (3-35):

```
(3-35) [instance: order34
                  (order
                     (agent Ross)
                     (patient hamburger61)]
```

This says that there is an instance of a hamburger, namely `hamburger61`, that Ross ordered, as in (3-34). Bruin inferred that the exact same hamburger was the one that was then served and eaten (Wong 1981b:157), but this is obviously an unsatisfactory situation.

The reason for this problem is that although Frail contains intensions, namely generic frames, its deductive section supports only reasoning with instances of frames. That is, there is no way to replace the instance `hamburger61` in (3-35) with the generic `hamburger` frame and still have a legal Frail statement that the reasoning component can manipulate. Frail can reason only about individuals, not abstract descriptions.

There has been little work as yet in AI on intensional representations. As soon as first-order representations are left in favor of representations such as the typed lambda-calculus, major problems, such as a lack of decidability, immediately occur (DS Warren 1983). For preliminary work on the topic see Maida 1982, Maida and Shapiro 1982, DS Warren 1983, and Rapaport 1985; for the detection and representation of intensional ambiguities, see Fawcett 1985 or Fawcett and Hirst 1986; for an opinion on why problems of intensions don't matter very much, see Hobbs 1985.

Since Frail does not provide the necessary support, Absity makes no attempt to handle intensions or opaque contexts. Even if intensions could be represented, however, they would be difficult for Absity, because, ironically, it is TOO compositional: the noun phrase rules always take an NP to refer to an extension, and once it is so construed, none of Absity's higher-level rules have the power to change it.

Exactly how this might be fixed would depend on the particular intensional representation. It is reasonable, however, to assume a representation like Montague's intensional logic (1973) with an operator that converts an intension to an extension at a given index. Absity might then treat all NPs as intensional, but add to each a flag that indicates whether the extension operator should be applied to the NP's representation when it is evaluated by the frame language. This flag could be a pseudo-word, and be "disambiguated" to either the extension operator or the identity operator by the same lexical disambiguation procedures described in chapter

5 for other words. An alternative that might be possible with some representations is to conflate this flag with the determiner of the NP. Thus, for example, the *a* of *a unicorn* would be regarded as an ambiguous word that could map to either an intensional or extensional frame-determiner. Whether this is possible would depend on the relationship between frame determiners and intensionality in the particular representation.

2. Stative and dynamic verbs, and habitual actions. Absity is able to deal with both DYNAMIC and STATIVE verbs, but its approach is not extensible to HABITUAL ACTIONS. A dynamic verb is one that describes an action:

(3-36) Nadia <u>wrote</u> an angry letter to the company president.

while a stative verb describes a continuing state:

(3-37) Nadia <u>knows</u> how to get to Boston.[24]

We have already seen that dynamic verbs can be represented by Frail instances:

(3-38)
```
[instance:  write34
            (write
                (agent Nadia)
                (patient letter61)
                (destination president44)]
```

With statives, however, there is no single instance (in the general sense of the word) of an occurrence of the action. This leads to the temptation not to represent statives with instances (in the Frail sense of the word), but rather to use the logic face of Frail and regard a stative as a continuing relationship between its AGENT and its PATIENT:

(3-39)
```
(know Nadia Boston-travel-method61)
```

Giving in to this temptation, however, presents an immediate problem of noncompositionality, and in fact the temptation is resistible, for instances are in fact quite adequate for representing statives:

(3-40)
```
[instance:  know34
            (know
                (agent Nadia)
                (patient Boston-travel-method61)]
```

[24]An easy way to distinguish the two is that stative verbs cannot take the progressive aspect (Quirk, Greenbaum, Leech, and Svartvik 1972: 94):

(i) Nadia <u>is writing</u> an angry letter to the company president.

(ii) *Nadia <u>is knowing</u> how to get to Boston.

As long as stative verbs are defined as such in Frail, with the appropriate axioms, we can use the same representation for both dynamic and stative verbs and maintain compositionality. Moreover, this is as it should be: a semantic interpreter should not have to worry itself about stativeness, since it is a feature of the meaning of the verb rather than of the word's use. Absity should therefore be able to handle both dynamic and stative verbs identically and let Frail act on each type accordingly. This is the approach that I have adopted.[25]

However, if we accept this analysis, we still have the problem of what to do with sentences that express habitual actions. Habitual actions may be thought of as the generic form of dynamic verbs, but are like statives in that they express a continuing relationship:

(3-41) Squirrels eat acorns.

(3-42) Squirrels pursue Nadia.

(3-43) IBM makes computers.

(3-44) Nadia visits Ross (daily).

Note that both the subject and the object of a verb expressing an habitual action may be either extensional or intensional; the examples above show all four possible combinations. I have already discussed problems of intension, so let's concentrate on (3-44), in which both NPs are extensional. On the one hand, it is clearly wrong to represent (3-44) as an instance, since it describes many individual acts of visiting; but on the other hand, it seems equally funny to say that Ross and Nadia stand in a relationship of visiting:

(3-45) `#(visits Nadia Ross)`

as this suggests that it is true at all points in time that Nadia is visiting Ross—an implication that is false even though (3-44) itself may be true at all points in time.

Because of these problems, we do not extend Absity's treatment of statives to habitual actions. For Absity to be able to handle habitual actions, we will need a suitable representation for them in Frail, and, in particular, a representation consistent with Absity's approach to compositionality. One possibility is a flag, similar to the one posited above for intensions, that is disambiguated appropriately. We will also need a way to decide whether a particular occurrence of a dynamic verb is habitual or not, so that this flag may be set.[26]

3. Predication and identity. In English, the verb *be*, in addition to being an auxiliary, has three primary senses: the PREDICATIVE *be*, which asserts a property; the IDENTITY *be*, which asserts the extensional identity of two intensions; and the

[25]Stativeness axioms have not yet been implemented in Frail, however.

[26]In English, it is probably safe to assume that an occurrence of a dynamic verb in the simple present tense represents an habitual action unless there is evidence to the contrary; *cf.* Quirk, Greenbaum, Leech, and Svartvik 1972: 85.

DEFINITIONAL *be*, which asserts the identity of two intensions. The predicative *be* generally takes an adjective phrase (which may reduce to a prepositional phrase) describing the property that the referent of the subject is being said to have:

(3-46) Ross <u>is</u> exhausted.

(3-47) Everyone <u>is</u> eager to see what God hath wrought.

(3-48) Of all the trees in England,
  Her sweet three corners in,
Only the Ash, the bonnie Ash
  Burns fierce while it <u>is</u> green.[27]

(3-49) Hi! handsome hunting man
Fire your little gun.
Bang! Now the animal
<u>Is</u> dead and dumb and done.
Nevermore to peep again, creep again, leap again,
Eat or sleep or drink again, oh, what fun![28]

(3-50) To the Puritan all things <u>are</u> impure.[29]

The *be* of identity takes a noun phrase and asserts its extensional identity with the subject NP:

(3-51) Ambition, in a private man a vice,
<u>Is</u>, in a prince, the virtue.[30]

(3-52) Custom, then, <u>is</u> the great guide of human life.[31]

(3-53) Brothers and sisters I have none,
But this man's father <u>is</u> my father's son.

The *be* of definition takes a noun phrase, and asserts its intensional identity with the subject NP; that is, it defines the subject:

(3-54) A classic <u>is</u> something that everybody wants to have read and nobody wants to read.[32]

[27]DE LA MARE, Walter John. "Trees". *The complete poems of Walter de la Mare*. London: Faber and Faber, 1969. 180. [*Peacock pie: A book of rhymes*. 1913.] Reprinted by permission of the Literary Trustees of Walter de la Mare and the Society of Authors as their representative.

[28]DE LA MARE, Walter John. "Hi!". *The complete poems of Walter de la Mare*. London: Faber and Faber, 1969. 268. [*Poems for children*. 1930.] Reprinted by permission of the Literary Trustees of Walter de la Mare and the Society of Authors as their representative.

[29]LAWRENCE, David Herbert. *Etruscan places*. New York: Viking Press. 1932.

[30]MASSINGER, Philip. *The bashful lover*. 1636. I, ii.

[31]HUME, David. *An enquiry concerning human understanding*. 1748.

[32]TWAIN, Mark. Attributed.

(3-55) A cynic <u>is</u> a man who knows the price of everything and the value of nothing.[33]

It was one of the features of Montague's PTQ formalism that it was able to handle both the *be* of identity and the predicative *be* with the same representation for each and with the same mechanisms as used for other verbs.

In frame terms, the *be* of identity asserts that two frame instances previously believed to be different are actually the same. For example, (3-53) says that `person245`, which represents **this man's father**, and `person112`, which represents **my father's son**, should be identified with one another, and anything known to be true of one is therefore now known to be true of the other. The *be* of definition similarly asserts that two generic frames are identical. The predicative *be* asserts the value of a slot in an instance or generic frame. Thus, if *exhausted* translates as `(tiredness=high)` and *Ross* as `Ross`, then (3-46) says that the `tiredness` slot of `Ross` contains the value `high`.

Frail has no innate mechanism for asserting the identity of two instances or two generic frames, and thus cannot at present handle the *be*s of identity and definition. It would be possible, however, to add a mechanism triggered by assertions of the form of, say, (3-56) and (3-57), that toured the knowledge base making the appropriate adjustments:

(3-56) Nadia is the captain of the football team.
`(same-instance Nadia captain21)`

(3-57) A stool is a chair with no back.
`(definition stool (chair (back=nil)))`[34]

Frail is able to handle the predicative *be*, though non-compositionally and with an awkward syntax. For example, the translation of (3-58) would be as shown:

(3-58) The block is red.
`(assert '(:= '(color block23) red))`

[33]WILDE, Oscar Fingal O'Flahertie Wills. *Lady Windermere's fan*. 1892.

[34]If the `stool` frame has not been defined previously, then this statement can just reduce to a basic frame definition:

```
(i)  [frame:  stool
        isa:  chair
      slots:  (back=nil)]
```

However, if the statement is intended to provide new information about a predefined frame, then Frail's very strict view of the ISA hierarchy *(see sections 1.2 and 1.3.1)* may result in a conflict between the old and new definition. In the present example, if the frames `stool` and `chair` had already been defined as siblings in the hierarchy with, say, `seat` as their parent, then the attempt to redefine `stool` would cause Frail to fail. On the other hand, if `stool` had been placed below `chair` in the hierarchy, there would be no problem.

Another problem, which may occur if `stool` is undefined in Frail when the sentence occurs, is that the word *stool* is also undefined in the lexicon. Neither Paragram nor Absity can yet handle such a situation with suitable grace.

This assigns the instance `red` to the `color` slot of `block23`. A generic predication has a slightly different syntax:

(3-59) The ash is green.

```
        (assert '(:= '(color (ash)) green))
```

Clearly, compositionality will be a serious problem in adding *be* to Absity, as the sentence itself will not correspond to an instance. A partial solution may be achieved by recognizing that the syntax of predications is different from that of other sentences that we have looked at above. Thus, we might add this rule to the base grammar:

(3-60) S → NP BE ADJ

where BE is a new syntactic category for *be*.[35] Let us then introduce two new semantic types: a PREDICATOR, which will be a basic type corresponding to the category BE, and a PREDICATION, formed from a predicator, an instance (the type of NPs), and a slot–filler pair (the type of adjectives). Thus, we could have the following translation:

(3-61) The block is red.

```
        (slot-value (the ?x (block x)) (color=red))
```

where `slot-value` is the predicator that is the translation of *be* and turns its arguments into the form required by Frail, as shown in (3-58) above. When intensions are added to Absity, as discussed earlier, (3-59) may be handled in a similar manner.

It should be pointed out that this solution is still problematic, because sentences now have two different semantic types: instances and predications. As long as we are dealing only with simple sentences like the examples above, we can turn a blind eye to the problem. However, when a predication is used as a subsentence, type conflicts will arise; compare:

(3-62) Ross knows that Nadia fed Daryel.

```
        (a ?x
           (know ?x
                 (agent=Ross)
                 (patient=(a ?y (feed ?y
                                      (agent=Nadia)
                                      (patient=Daryel))))))
```

(3-63) Ross knows that the block is red.

```
        (a ?x
           (know ?x
                 (agent=Ross)
```

[35]The category BE will also include a few other words that act similarly, such as *become*, *wax*, and *turn*. These words tend to imply change as well as predication, and I will not discuss this additional complexity; see Somers 1983: 258.

```
                  (patient=(slot-value (the ?x (block x))
                                              (color=red))))))
```

The translation of (3-63) is obviously unsatisfactory, and some method by which a predication can at least masquerade as an instance is needed.

We can extend the method we used for predicative *be*s to identity *be*s. These will require this new base rule:

(3-64) S → NP BE2 NP

BE2 is another new syntactic category that also contains *be*. Its type will be FRAME EQUIVALENCER, and together with two instances, one from each NP, it will form a FRAME EQUIVALENCE STATEMENT that will correspond to the sentence. For example:

(3-65) This man's father is my father's son.

```
(same-instance (the ?x (father ?x ...))
               (the ?y (son ?y ...)))
```

Here, `same-instance` is the translation of *be* when its syntactic category is BE2. When intensions are added to Absity, this rule should be able to handle the definitional *be* as well. It will then, however, be necessary to decide whether *be* translates to `same-instance` or `definition`; this could be done by the word disambiguation procedures that I will describe in chapter 5.[36]Note that the rule has the same difficulties with embedded sentences that the rule for predicative *be*s has.

4. Complex quantifiers. Frail has no method at present to represent sets or their cardinality. One can in Frail retrieve a set of instances with a given property, but all one gets is a list of bindings. For example, to find **the students whose fathers like cheese**, one could ask Frail for the following:

(3-66)
```
(retrieve
  '(and (student ?x)
        (likes (father ?x) cheese)))
```

The result, however, is simply a list of the various bindings of `?x` that satisfied the request:

(3-67) `((?x Ross) (?x Nadia) (?x Kennapod))`

There is no facility for representing a collection of instances as a Frail object, nor for representing a set intensionally by specifying a generic frame that describes its members, nor for representing the cardinality of a set. Thus Frail cannot represent as a semantic object any of the following NPs:

(3-68) the students whose fathers like cheese

(3-69) the integers

[36]See also Mallery 1985.

(3-70) seven red blocks

(3-71) all but five of Nadia's marmots

Thus complex determiners such as *all but five of the* can be neither parsed by Paragram nor represented by Frail; Absity therefore has no qualms about ignoring them too. Simple plurals pose no problem for Absity, except that Frail doesn't know what to do with them.

Sentences with QUANTIFIERS whose scope is wider than the noun phrase in which they occur cannot yet be handled by Absity:

(3-72) Every boy gave every girl three peaches.

Sentence (3-72) implies that there was not one giving event but rather a number of them, equal to the product of the number of boys and the number of girls. The over-compositionality of Absity is the main problem here: the universal quantifications have to be pulled out of the noun phrases so that their scopes are correct, but Absity is not amenable to that. In addition, the representation of such sentences in Frail is problematic; if we solved the previous problem, we could then generate a representation with a frame determiner that found all instances of `boys` and `girls` and asserted a `give` instance for each pair. But if there were many pairs, this would be both inefficient and counter-intuitive; rather, it would be preferable to assert the fact that there were these givings, and use it later to infer, for example, that Ross gave Nadia three peaches if that fact is needed. Possible solutions include the use of a representation that is vague with respect to scoping, as suggested by Hobbs (1983), and the use of QUANTIFIER STORAGE (Cooper 1983).

5. Inherent vagueness. Often language is INHERENTLY VAGUE, and it is left to the interpreter to supply that which is unsaid. Two particularly important instances of this are the words *have* and *with*, many uses of which are extremely vague. Consider:

(3-73) Ross <u>has</u> green eyes / a cold / a sinister laugh / intelligence / a book under his arm / . . .

(3-74) the girl <u>with</u> green eyes / a cold / a sinister laugh / intelligence / a book under her arm / . . .

At present, Absity relies on being able to determine the precise meaning of a word from its context. In the case of words like *have* and *with*, however, it seems wrong to say that they are ambiguous words with a large number of precise meanings; rather, they seem to be inherently vague, with their intent to be figured out from the nature of the entities they relate.

I believe that this could be handled properly with the present Absity and Frail by adding a mechanism that takes a vague sentence in the Frail representation and uses the definitions of the objects to resolve the vagueness.[37] In anticipation of

[37]Such a mechanism is described very briefly by Steinacker and Trost (1983).

such a mechanism, Absity simply translates all verb occurrences of *have* as `have`, and all NP-attached uses of *with* as `attr` (an abbreviation for *attribute*).

6. Time and space. The representation of time is an extremely difficult problem. While Frail can represent points in time and time intervals as frame instances, it does not support any form of temporal logic. Similarly, its ability to handle spatial location is very limited. Absity can handle the simple representations of time and space that Frail permits, and it is to be hoped that as better representations are developed, they will be suitably compositional and amenable to Absity's approach.

7. Moral and contingent obligation. As is all too typical of computer systems, Frail cannot even conceive of contingent obligation, let alone moral imperatives:

(3-75) Nadia <u>ought</u> to prevent Ross from driving until he sobers up a bit.

(3-76) You <u>ought not</u> to read Hirst's book if you are under 18.

(3-77) The wheelhouse vacuum chamber pump bearings <u>should</u> be oiled daily.

(3-78) Persons desiring an I–94 form <u>must</u> complete an IAP–66 form.

Again, it is to be hoped that as better representations are developed, they will be suitably compositional and amenable to Absity's approach.[38]

8. Negation and conjunction. Frail permits the assertion that a particular instance does not exist. Thus (3-79):

(3-79) Ross didn't kiss Nadia.

could be represented as (3-80):

(3-80)
```
(not (and (instance ?x kiss)
          (:= (agent ?x) Ross)
          (:= (patient ?x) Nadia)))
```

[38]In fact, Frail is able to represent contingencies in a manner suitable for use by Bruin's NASL problem solver (Wong 1981a, 1981b) *(see section 1.3)*. Thus, one might have the following translation:

(i) To get an I–94 form, complete an IAP–66 form.
```
(to-do (possess I-94) (complete IAP-66))
```

The same kind of information may also be represented in a script-like frame:

(ii)
```
[frame:  getting-I-94
   isa:  task
 slots:  (first-step (complete IAP-66))
         (second-step (receive I-94))
 facts:  (before ?first-step ?second-step)]
```

These representations are designed for manipulation by NASL, but are not suitable as targets for semantic interpretation.

Absity, however, is unable to handle (3-79), or negation in general. To do so, it would need a new frame determiner, `not a`, say, that could provide a suitable meta-notation for (3-80), as described in section 3.6, and a way of getting the negation into the frame determiner.

Absity also makes no attempt to handle the conjunctions *and* and *or* at any constituent level.

9. Noun modifiers. Absity considers all noun modifiers to be slot–filler pairs that qualify their head noun. This implies that the modifier is either an ABSOLUTE or MEASURE adjective (Ali 1985), and that (from Absity's perspective) the relationship between it and the noun is defined solely by the modifier. For example, if *red*, an absolute adjective, translates as `(color=red)`, then the relationship is that the modifier is saying something about the noun frame's `color` slot.[39] Similarly, the measure adjective *tall* translates as `(height=tall)`, though measure adjectives bring the additional problem (for Frail, not Absity) that their final interpretation also depends on the noun; a *tall jockey* is probably not as tall as a *short basketball-player*, for example. (Ali (1985) has constructed a system for the interpretation of such constructs.)

Problems arise with other classes of noun modifiers, such as ROLE adjectives and INTENSIONAL adjectives. A role adjective is one such as *annoyed* in *an annoyed look*, which may be glossed as **a look whose agent is an annoyed person** (Ali 1985; Ali's system could also handle these adjectives). An intensional adjective is one such as *alleged*; an *alleged thief* may not be a thief at all.

In general, the relationship between modifier and head noun is often controlled by both, and is sometimes not one of frame and slot–filler pair at all. For example, a reasonable representation of *computer science* is (3-81):

(3-81) `(a ?x (science ?x (object-of-study=computer)))`

This implies that the meaning of *computer*, when used as a noun modifier, is `(object-of-study=computer)`. But the same reasoning gives us many other meanings for the same word:

(3-82) computer maintenance
`(a ?x (maintain ?x (patient=computer)))`

(3-83) computer language
`(a ?x (language ?x (comprehender=computer)))`

(3-84) computer game
`(a ?x (game ?x (medium-of-play=computer)))`

It would be wrong simply to regard *computer* as a highly polysemous adjective. In fact, language is extremely productive in its creation of such noun groups (Downing 1977, Levi 1978, B Warren 1978), and it seems that the relationship between

[39]If it doesn't have such a slot, then there's trouble; if the adjective or noun is ambiguous, then making sure that the sf-pair fits the frame is a good heuristic for disambiguation; see section 5.3.2.

the head noun and the modifier does not inhere in the modifier, nor even in the head, but rather is constructed by inference on the meanings of both (Bauer 1979). While a frequently used noun group will generally become LEXICALIZED (that is, treated as a single word or "CANNED PHRASE" with its own entry in the mental lexicon), people generally have no trouble creating and comprehending novel forms (Downing 1977).

There is nothing in Absity at present for inferring or interpreting such relationships, and this is obviously a major gap. One interesting possibility for a solution comes from the work of DB McDonald (1982), who used for noun group interpretation a marker passer not dissimilar to the one I use in chapter 5 for lexical disambiguation. It may therefore be possible to employ the marker passer for noun group interpretation in Absity as well. In addition, Finin (1980) described an approach to noun group interpretation, based on a frame system not unlike Frail, that might be adaptable.

10. Non-restrictive noun phrase modifiers. Absity is not able to handle non-restrictive modifiers, whether appositive or not:

(3-85) Ross, <u>whose balloon had now deflated completely</u>, began to cry.

(3-86) Ross <u>in a bad mood</u> is a sight to be seen.

Ideally, (3-85) should be translated into two separate Frail statements representing the two sentences from which it is composed:

(3-87) Ross's balloon had now deflated completely.
Ross began to cry.

but Absity has as yet no mechanism for this.[40] Sentence (3-86) is problematic because it is unclear exactly how the NP *Ross in a bad mood* should be represented in Frail.

11. Adverbial *wh*- questions. Questions with an adverbial *wh*- can be difficult. In section 3.5, we saw that the translation is straightforward if the *wh*- is an NP:

(3-88) What did Nadia buy from a store in the mall?

```
(question ?u
    (buy ?u
        (agent=(the ?x (person ?x
            (propername="Nadia"))))
        (patient=?WH)
        (source=(a ?z (store ?z
            (location=(the ?w (mall ?w)))))))
```

We knew that the `?WH` was the PATIENT because the syntax of the sentence indicated that the pseudo-preposition *OBJ* was its case flag. However, adverbial *wh*-s contain their own case flag, and it is not always unambiguous:

[40]In anticipation of such a mechanism, however, section 7.3.2 discusses structural ambiguity resolution for such cases.

(3-89) Where did Nadia kiss Ross?

It would be wrong simply to translate *where* as `(location=?WH)`, because there are two different LOCATION slots for *kiss*: one a true case slot, the other a verb modifier. This is demonstrated by the two possible kinds of answer to (3-89):

(3-90) On the cheek.

(3-91) Behind the gym.

Thus the translation of *where* is not always unambiguous, even when the sentence verb is known; moreover, it is probably NOT amenable to the local disambiguation techniques of chapter 5, because context and discourse pragmatics are necessarily involved in determining exactly what is being asked *(cf. section 5.3.6)*.[41]

12. Words with apparently non-compositional effects. There are many words whose semantic characterization is problematic in any compositional semantics. These are words, usually adverbs, that serve to change the meaning of their matrix sentence (or constituent) in some way. Examples:

(3-92) Ross has been rummaging through Nadia's wastebasket <u>again</u>.

(3-93) Ross <u>even</u> suggested that Nadia spend the weekend at his apartment.

(3-94) Ross hasn't <u>actually</u> made it with Nadia <u>yet</u>.

(3-95) He hasn't touched her, <u>let alone</u> kissed her.

These sentences are simple enough to represent if the underlined words are omitted (or replaced by *or*, in the case of (3-95)). But it is by no means clear how the full sentences, let alone the underlined words, should be represented if the meanings of the full sentences are to be formed compositionally from the representations of the simple forms and those of the underlined words.

It is tempting, and not without motivation, to sweep these words under the rug of discourse pragmatics, along with other matters discussed in point 13 below; that is, to say that they have no semantics per se and Absity can simply ignore them. For example, in (3-92), the word *again* seems to mean "and this is a new occurrence of such an event, not any of the ones that I believe you've heard of before"—that is, it is a meta-level comment on the knowledge of the participants of the discourse. Similarly, in (3-94), we can gloss *actually* as "despite indications or beliefs to the contrary that you may have", and *yet* as an intensifier of the verb tense, with the conversational implicature (Grice 1975; Levinson 1983) that the situation described may not remain true.

This is not an entirely satisfactory situation, however. If Absity ignores these words, then we must posit some other process that deals with them, a process that must necessarily use the syntactic and semantic representations created by Paragram and Absity. The questions of methods of interaction and possible changes

[41] In any case, Paragram can't parse adverbial *wh*-s yet.

to Absity then arise. Moreover, if Absity's structures are to be used in machine translation between natural languages, then they are now lacking important information that must be carried over to the output; it is distressing and counterintuitive to have to posit a second representation in which such information is carried separately from the meaning itself and then somehow re-integrated with it.

My intuition is that if these words are to be handled at the semantic level, then they should be represented not as passive objects for composition, but rather (more in the original Montague style) as functions that take a semantic object (for a sentence or other constituent) and produce a suitable modification of it. It may be possible to do this without too much violence to the principles of Absity.

13. Topic, tenor, and subtleties of discourse. We have already *(section 1.1.1)* discussed the exclusion of matters of discourse from Absity, but it is worth mentioning briefly here some of the things we have lost. In particular, Absity ignores the shades of meaning conveyed by surface form and choice of words. For example, topicalization is ignored; (3-96) and (3-97) would receive exactly the same representation in Frail:

(3-96) Egg creams, I like.

(3-97) I like egg creams.

Irony is also lost on Absity; compare:

(3-98) While I was holding the baby, <u>it</u> wet itself.

(3-99) While I was holding the baby, <u>the little darling</u> wet itself.

Presumably, any anaphora mechanism working with Absity would resolve both *it* and the epithet (Hirst 1981a[1]: 13) *the little darling* as **the baby** and simply plug in the appropriate frame statement `(the ?x (baby ?x))` in each case. As with the matter discussed in the previous point, this sort of discourse subtlety must also be preserved in machine translation.

3.9 Conclusion

I have presented Absity, a semantic interpreter that works with the Paragram parser and the Frail frame system. Absity is clean and compositional, works in tandem with the parser, and provides feedback for structural disambiguation. Absity is still limited by the abilities of Frail and by its own over-compositionality, but should provide a firm foundation for further development of semantic interpretation. Table 3.5 shows some of the sentences that Absity can handle, and some that it can't. (Some of the examples assume the disambiguation mechanisms of chapters 5 and 7.)

Table 3.5. *What Absity can and can't do*

SENTENCES THAT CAN BE INTERPRETED
Nadia gave Ross a marmot for his birthday.
The fish that Ross loved loved the fish that Nadia loved.
Ross promised to study astrology.
Nadia wanted the penguin to catch some fish.
What did Ross eat for breakfast?
Did a computer destroy the world?
Ross knows that Nadia assembled the plane.
A seagull landed on the beach.
What does Nadia want from Ross?
An obnoxious multi-national corporation wants to hire Nadia.
SENTENCES THAT CAN'T BE INTERPRETED
Nadia resembles a pika. *(de dicto reading)*
Ross sleeps on the floor. *(habitual reading)*
Ross is exhausted.
The mysterious stranger was Nadia.
All but five of the students whose fathers like cheese gave three peaches to many of the tourists.
Ross ought to swim home tomorrow.
Nadia didn't see that Ross was creeping round the corner.
Computer games are devastating the youth of the nation.
Ross in a bad mood should be avoided.
Where did Ross buy the unicycle?
Ross hasn't actually made it with Nadia yet.