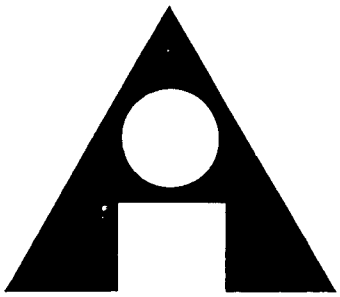
Granger,
Follow-up



IJCAI-77

5TH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE -1977

IJCAI-77 • PROCEEDINGS OF THE CONFERENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MASSACHUSETTS, USA
AUGUST 22 - 25, 1977

VOLUME ONE

Sponsored by the
International Joint Conferences on Artificial Intelligence

FOUL-UP
A program that figures out meanings of words from context

Richard H. Granger, Jr.
Department of Computer Science
Yale University
New Haven, Connecticut 06520

The inferencing task of figuring out words from context is implemented in the presence of a large database of world knowledge. The program does not require interaction with the user, but rather uses internal parser expectations and knowledge embodied in scripts to figure out likely definitions for unknown words, and to create context-specific definitions for such words.

INTRODUCTION

"I woke up yesterday, turned off my alarm clock, took a shower, and cooked myself two grimps for breakfast." What's a grimp? On the basis of this one story, it can't be determined. It seems, however, that a grimp must be some sort of food, in order for it to make sense in the context of the story. The fact that a grimp is a physical object, and thus a noun, is such a natural inference that it is taken for granted. The process by which these conclusions about "grimps" are arrived at is commonly called "figuring out a word from context". This process is modelled by the FOUL-UP program. FOUL-UP enables the SAM system (see Schank et al, 1975) to read stories such as the above, and to make an educated guess as to what a "grimp" should be. The methods employed by FOUL-UP are based on intuitions about how the analogous tasks are performed by people. Thus its abilities and limitations should reflect those of people. FOUL-UP provides the following purely practical assets to SAM: (1) it allows reading of unedited newspaper stories, which typically assume a large vocabulary, and (2) it creates dictionary definitions for unknown words, which would otherwise have to be done by hand for every word in SAM's dictionary. Other natural language-based systems have attempted to allow for the occurance of unknown words during understanding, (see Woods & Kaplan, 1971, and Winograd, 1971), but these usually involve having the user aid the system in defining the unknown word. FOUL-UP acts independently of the user, taking its information entirely from the internal representation of knowledge that exists in the SAM system.

This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored under the Office of Naval Research under contract N00014-75-C-1111.

BACKGROUND: THE SAM SYSTEM

1. Overview of SAM

FOUL-UP was designed as an integral part of the SAM system, so any discussion of its performance will involve some knowledge of SAM itself. The SAM system exists as three main modules, ELI (English Language Interpreter), TOK (Tokenizer) and APPLY (Script Applier). Every input sentence is first parsed by ELI into a Conceptual Dependency (CD) representation of the sentence. Then TOK incorporates specific objects into memory pointers, for use in disambiguating later anaphoric references. Finally, APPLY, which contains the scripts, attempts to match every CD into the currently active script.

When an undefined word like "grimp" occurs in a sentence, it is ELI that will first encounter it, and it will "foul up" the normal processing protocol, by calling FOUL-UP with the word that caused the trouble. Ideally, at this point FOUL-UP should attempt to guess the word's meaning by looking into the context of the surrounding script (e.g. "the Get-Up-In-The-Morning script"), to see what object was expected in the position that the unknown word "grimp" occurred in (see Riesbeck & Schank, 1976). The "position" of the unknown word does not refer to the surface position in the sentence, but rather the conceptual slot in the CD, corresponding to the unknown word's meaning. This process cannot be immediately invoked at the time of ELI's foul-up, for three reasons:
(1) although the script has already been entered, we have not yet APPLYed the CD conceptualization for the current sentence into the script, so we therefore do not yet have the specific local script information that we need to find the expected meaning of the unknown word;
(2) ELI has not yet finished its parse of the current sentence, so there does not yet exist a complete CD to match into the script;
(3) since ELI has not finished the parse, we do not yet know which slot of the CD will be filled by the unknown word, so we could not find the corresponding slot in the script, even if we could match the CD into the script.

FOUL-UP could, of course, attempt to complete the CD by itself, use ELI's expectations to find the correct slot for the unknown word, match the CD into the script, and thus guess a possible meaning for the word. However, with a few modifications, the normal operation of the SAM system will do all this for us. Thus FOUL-UP makes a note of the current ELI expectations for future reference, and then proceeds to pass back a "place-holder" for ELI to use in constructing the current CD. This place-holder is so constructed as to satisfy any current ELI expectation, so when control returns to ELI, it will automatically put this place-holder into the slot that the unknown word belongs in. The expectations that ELI has for such incoming words consist of both syntactic and semantic rules. Thus ELI ex-

pects the incoming word to be a noun, and a PP ("picture producer") (see Schank, 1975). All of ELI's expectations are such that the expected word will fit appropriately into the sentence in which it appears. ELI is not capable of ensuring that the word will fit appropriately into the story, or any other context larger than a sentence. It is the script applier (APPLY) that checks larger contexts, specifically, the script context of the story. We will call ELI's expectations "intra-sentence" expectations, while APPLY's are called "inter-sentence" expectations.

2. Script Context

Once control has been thus returned to ELI, and it has incorporated the place-holder into the CD, then control continues through the rest of the system as though the foul-up had not occurred. Meanwhile, FOUL-UP has notified the system that a place-holder exists in the current CD, so that it can be handled by each module correctly. Thus TOK is able to tokenize the place-holder correctly, and then APPLY is called. APPLY now has a CD containing a place-holder to attempt to match into the script. It performs a partial pattern-match, allowing for the existence of the place-holder, and noting which script-slot corresponds to the place-holder slot in the CD.

At this point in the processing, all three of the above necessary processes have been performed: a slot has been found for the unknown word, a CD has been created, and that CD has been matched into the script. Now FOUL-UP is called back in by APPLY. Since the CD has been matched into the script, there is a conceptualization in the script which corresponds to the CD. This script conceptualization contains attributes associated with each slot in the CD, including that of the place-holder corresponding to the unknown word. These attributes are ordinarily used to ensure that the conceptualization appearing in a given slot is compatible with the larger context surrounding that slot. Thus, for example, it expects to see a food as the object of the instrumental script "cook" in the optional "Make-Breakfast" scene of the "Get-Up-In-The-Morning" script. FOUL-UP looks at the slot in the script corresponding to the place-holder, and notes the attributes associated with that slot. Then it compares these inter-sentence expectations with the intra-sentence expectations that ELI had for the slot. It combines the compatible set of attributes into a dictionary entry for the unknown word, and writes this definition onto a file, called the Primer, which the parser from then on uses as part of it's dictionary. Thus if the word "grimp" appears again in a similar context, SAM will "remember" it as being a noun and a PP, and having appeared before as a food in a "Breakfast" story, so it will successfully fill the appropriate slot in a CD, and will correctly match into the script.

OPERATION OF THE FOUL-UP PROGRAM

1. Annotated Run-Time Output

The following represents actual annotated run-time output of the FOUL-UP program running concurrently with the SAM system. The input to the system is the simple story:

"Friday, a car swerved off Route 69. The car struck an elm."

It is given that the system knows all the words in the story except "elm". The APPLY module has a script called $Vehaccident, which contains the necessary real-world knowledge about vehicle accidents (see Cullingford, 1977). The first sentence in the story goes straight through the system without incident, since all the words are defined. The major effect of the first sentence is to activate the script "$Vehaccident", recognizing that this sentence indicates the beginning of an accident story. The following output comes from the processing of the second sentence.

```
>ELI<
The car struck an elm.
```

ELI has no definition for the word "elm".

```
>FOUL-UP<
FOUL-UP notifying TOK & APPLY of unknown word ELM
Obtaining CONINFO and SYNINFO from ELI
Returning to ELI with bogus place-holder:
(#BOGUS LEXVAL (ELM) REF (DEF))
```

FOUL-UP has now created a place-holder for the unknown word. It has also stored conceptual & syntactic expectations associated with the word. These will be used later, during FOUL-UP's construction of a definition for "elm".

```
>ELI<
((ACTOR (#PHYSOBJ TYPE (*CAR*) REF (DEF)) <=>
(*PROPEL*) OBJECT (#BOGUS LEXVAL (ELM) REF
(INDEF))) TIME (TIM2))
```

ELI creates as much of a CD from this sentence as it can, including the place-holder from FOUL-UP. All it really knows is: the actor is a car, and "strike" indicates a "propel".

```
>APPLY<
APPLY locating unknown PPs in top-level TOK atom
Bound script variable: &OBSTRUCTION to BOGO
```

APPLY does a partial pattern-match, and points out slot in the match which corresponds to place-holder in the conceptualization from ELI

```
>FOUL-UP<
Attempting to partially understand ELM
ELI expectations were:
((AND (OR (POBJ) (HUMAN)) (NP)))
```

ELI's expectations are looking for either a physical object or a human, and a noun phrase. These are simply intra-sentence expectations.

```
APPLY expectations were:
(Scriptrole &OBSTRUCTION in $VEHACCIDENT)
```

APPLYs inter-sentence expectations expect this slotfiller to play the role of &Obstruction in the script $Vehaccident. Now FOUL-UP will pick a compatible subset of these expectations, and construct a definition for the word "elm".

```
FOUL-UP guesses the following definition for ELM:
(PROG NIL
 (DEFPROP ELM T PP)                          (1)
 (DEFPROP ELM *PHYSOBJ* ISA)
 (DEFPROP ELM T NOUN)
 (DEC ELM                                    (2)
    (T (NIL ((SHAPE NP))  NIL
            (QUOTE (NOUNGROUP))
            (QUOTE (STARTNOUNGR))))))
 (DENG ELM                                   (3)
     (VAL (#PHYSOBJ FUNCTION
                ((<=> ($VEHACCIDENT
                       OBSTRUCTION MODFOCUS)))))
     (MARKER NOUN))
```

This definition is in three parts:
(1) put the properties PP, PHYSOBJ, and NOUN on the word, to fill ELI expectations;
(2) attach noun-grouper (see Gershman, 1977);
(3) define the conceptual structure that the word will build in a CD, saying that it is of the conceptual class #PHYSOBJ & plays the role of an obstruction in a car accident script.

2. Note: Context-Specificity of Definitions

After the above run of SAM, the resulting dictionary definition for "elm" would become a permanent part of SAM's dictionary. Let us now take a closer look at exactly what information is contained in the definition for "elm". At this point, "elm" has been defined as no more nor less than a physical object which plays the role of an obstruction in an accident script. This does not seem to reflect the true meaning of the word "elm" at all, but rather seems only to embody the meaning of the specific use of the word in this one story. Most words have more than one meaning or function, depending on the context they appear in. Since FOUL-UP is given only one context, the story itself, it can only guess the single meaning apropos of that single story. The reason it seems like a poor definition to us is because we in fact have seen the word "elm" many times before, and already have rather complete definitions for it, in many contexts. Note that if the sample run had contained the word "bogosity" in the place of "elm", ("The car struck a bogosity"), then FOUL-UP's definition may still have seemed sketchy, but it would be about as much as a person could have figured out from just that one exposure to the word. From the single use of "elm" (or "bogosity") in the story, it could be defined as a tree, or a telephone pole, or one of those large orange barrels often seen by the side of the road, or any number of other things. (It might seem at first that a "bogosity" could be another car, or a person, instead of an inanimate object. It is the first sentence, "A car swerved off Route 69" that causes APPLY (perhaps prematurely) to prefer the one-car accident track, in which bystanders and other cars play no part. This track is assumed in the absence of further information.) The only concrete thing we can say about a "bogosity" is the way it functions in the context of the script and story it appears in. That is precisely the information that a scriptrole conveys in the SAM system, and that is why FOUL-UP defined "bogosity" in terms of a scriptrole. In fact, without the inter-sentence expectations provided by the script, the definition would have been even weaker. Only ELI's intra-sentence expectations could have been used, and the definition would have simply said that an "elm" (or "bogosity") was a noun and a physical object.

CATEGORIES OF FOUL-UPS

1. Nouns

The FOUL-UP program works solely on the basis of intra- and inter- sentence expectations, built into the parser and the script applier, respectively. Thus any conceptualization which can be "expected" in this sense by SAM should be able to be figured out from context by the FOUL-UP program, and FOUL-UP should be able to use the available expectations to construct a context-specific definition for the word, regardless of its syntactic or semantic class. However, the examples so far given have dealt only with nouns. In this section, Foul-Ups of other parts of speech will be demonstrated, and it will be shown why each different syntactic class presents its own unique difficulties to the FOUL-UP program.

2. Anaphoric References

Consider the simple story:

"A car swerved off Route 69. The flivver struck a tree."

Again, we assume all the words are defined but "flivver", and we assume a run of the SAM system with this story as input. Once again, the first sentence presents no difficulty, and serves to invoke the script context which is necessary for setting up the inter-sentence expectations that FOUL-UP uses to figure out the meaning of the word. The flow of control of the SAM run of this sentence would be practically identical to that of the previous example. ELI's expectations are also similar. However, the inter-sentence expectations from APPLY are of a different form than those in the previous examples. APPLY still matches the CD for the second sentence into the $Vehaccident script, and still notes the attributes of the slot in the script conceptualization corresponding to the place-holder slot in the CD. Those attributes, however, are no longer in the form of a scriptrole specification. Instead, associated with that slot is a pointer, saying that the object in this slot is the same object

as that which appeared in the ACTOR slot of the previous sentence. This is obvious when we note that (1) the script must know that the object that "swerves off the road" in an accident is the same as the object that later is said to "strike" something, and (2) SAM as a whole has the capability to resolve general anaphoric references. Thus the second sentence in the story might have read simply "It struck a tree", and the anaphoric reference to "it" would be resolved as "the car" that appeared earlier.

When such a situation occurs, FOUL-UP takes advantage of the pointer to the referent, and constructs a definition based on this pointer. Such a definition will simply duplicate the definition of the word that filled the pointed-to slot, since the unknown word is presumably just another reference to the same object. Thus, the definition for an unknown word in this circumstance will be better than the "FUNCTION" definition of "elm" in the previous example. The definition will not simply give the function of the word in this script context, but rather will give as complete a definition as exists for the already defined word "car" which the unknown word is an anaphoric reference to. Thus in this example, the definition created for "flivver" will be identical to that for "car":

```
(PROG NIL
      (DEFPROP FLIVVER T PP)
      (DEFPROP FLIVVER *PHYSOBJ* ISA)
      (DEFPROP FLIVVER T NOUN)
      (DEC FLIVVER (T (NIL ((SHAPE NP)) NIL
                  (QUOTE (NOUNGROUP))
                  (QUOTE (STARTNOUNGR)))))
      (DENG FLIVVER (VAL (#PHYSOBJ TYPE (*CAR*)))
                  (MARKER NOUN)))
```

The "TYPE" specification in this definition is a reference to the entry "*CAR*" in a general dictionary which defines attributes of generic objects such as cars, and their uses in certain contexts. Thus anything that SAM knows about cars it now also knows about flivvers. Furthermore, it is likely that the term "flivver" refers to a specific subset of all cars, (e.g. "sedan", "jeep"). Therefore, the term is saved, and FOUL-UP gives the term to SAM's generator program (see Goldman, 1975) to use when referring to the car in this story.

3. Verbs

a. Structure-builders vs. Slot-fillers

Consider the story

"A car swerved off Route 69. The car hied into a tree."

The unknown word in the sentence, ("hied"), is recognized as being a verb by ELI expectations and by the fact that it ends in a regular verb ending. A simple place-holder cannot be straightforwardly constructed for the word, as was the case for nouns. This is because the verb in a sentence determines most of the structure of the Conceptual Dependency representation for that sentence. (see Schank, 1975, and Riesbeck & Schank, 1976) Essentially, the verb builds the framework outline for the CD, and sets up most of the expectations as to what will come next. Then the nouns build structures which fill slots in that verb-frame. This can be illustrated by looking at a parse of a sample sentence, "John went to New York". Partially "kernelized" CD representations for sentences will be used for convenience, to facilitate references to specific pieces of the conceptualization. The internal representations of CDs in ELI are entirely "kernelized" in this fashion. A partially kernelized CD for the above sentence follows:

```
K1: ((ACTOR (K2) <=> (PTRANS)
                OBJECT (K2) TO (K3)))

K2: (#PERSON FIRSTNAME (JOHN))

K3: (#LOCALE STATE (*NY*))
```

The word "went" builds the structure K1, with empty slots in the ACTOR, OBJECT and TO positions. The word "John" builds the structure K2, which simply fills in the ACTOR and OBJECT slots in K1, and "New York" builds K3, which fills the TO slot in K1. Although these constructs may look about the same size and complexity, it is in fact K1, the verb-frame, which drives the parse of the rest of the sentence. The verb itself sets up the expectations that a #PERSON should fill the ACTOR and OBJECT slots, and that a #LOCALE should fill the TO slot. Then when the words "John" and "New York" build such structures, those expectations dictate how they are incorporated into the CD. By contrast, the CD representation for the sentence "John hied to New York", assuming the unknown word "hied" is ignored, would simply be:

```
K1:  (#PERSON FIRSTNAME (JOHN))

K2:  (#LOCALE STATE (*NY*))
```

This representation is simply an unordered bag of noun groups, lacking the structure usually imposed upon it by the verb. Assuming that this kind of structure was built in the event of a verb foul-up, the later stages of processing would be quite difficult. This unstructured bag of noun groups cannot be matched into a script, at least not by the usual methods. The FOUL-UP program performs a four-step process to enable undefined verbs to be correctly defined in terms of the context in which they appear.

b. Step 1: A "Catch-All" Verb Frame

In the event that the unknown word in a sentence is a verb, FOUL-UP must create a verb-frame which is capable of expecting all the subsequent noun groups or other slot-fillers that may occur. Such a "catch-all" verb frame looks like the following:

```
((ACTOR (NIL) <=> (NIL) OBJECT (NIL) TO (NIL)
                        FROM (NIL)) TIME (NIL))
```

Furthermore, each of the slots in this CD-frame has associated with it a set of expectations that will accept virtually any conceptual class of noun that may appear. This ensures that all words in the sentence will at least be assigned some position in the CD. Note that the resulting CD, built by ELI and FOUL-UP, will become input to APPLY's pattern-matcher, when it attempts to match the CD into the script. Thus, it is crucial that the noun groups are put into the correct slots within the conceptualization, if the CD is to be recognized correctly by APPLY's pattern-matcher.

c. Step 2: Preposition Matching

This step consists of using knowledge of prepositions to tentatively place nouns into their appropriate slots. Thus the prepositions TO, TOWARDS, INTO, AT expect to fill the TO slot in the CD, while FROM, OFF, OUT expect to fill the FROM slot. Finally, if no preposition is present, the OBJECT slot is assumed. This heuristic is dependable up to a point, but difficulties arise when idiomatic uses of prepositions occur. As an example, assume the second sentence in the above story had read "The car caromed off a tree", where "caromed" is unknown to SAM. Then the conceptualization for "tree" would be put into the FROM slot in the CD, by the heuristic just stated for the preposition "off". However, the script pattern will be looking for the obstruction "tree" in the OBJECT slot, corresponding to the normal construction "The car struck a tree". Thus the position of the nouns in the CD must be thought of as tentative, and APPLY will often have to search more than one conceptualization before it finds the appropriate match.

d. Step 3: ACT Preference

The APPLY search can be limited to those conceptualizations in the script which contain at least the same noun groups as the input conceptualization, even if they're not in the correct slots. There are still likely to be two or more such potential matches in the script, which APPLY can not dependably decide between. Thus FOUL-UP uses an additional heuristic to aid the search, that of "ACT Preference". Based on the prepositional phrase that appears in the sentence, there is quite often a unique primitive ACT implied, along with a set of expectations for the slot-fillers. Thus if we see a sentence like "John hied to New York", we easily infer, even in the absence of a script or other large context, that the ACT in the CD for this sentence will probably be a PTRANS, because of the preposition "to" followed by a location. The prepositional phrases considered are those which contain a preposition followed by a noun group. The SAM system is able to classify noun groups into conceptual classes, such as #PHYSOBJ (physical object), #PERSON (human being), #CONCEPT (sub-clause), #LOCALE (place), etc. When certain prepositions occur in a prepositional phrase combined with certain conceptual classes of nouns, the primitive ACT in the sentence can often be predicted simply from this information. The following table is a segment of a table of such correlations, used in FOUL-UP to help predict what the ACT in the sentence will be. Given in the table are common prepositions on one axis, and common conceptual classes of nouns along the other axis. For each possible combination of preposition and noun in a prepositional phrase (e.g. "to" and "#PERSON" in "He gave the book to Mary"), the corresponding preferred ACT is looked up in the table (e.g. ATRANS), as well as the predicted slot in the CD for the noun to fill (the TO slot).

	#PHYSOBJ	#PERSON	#LOCALE	#CONCEPT	#BODYPART
OFF	PTRANS? FROM		PTRANS FROM	MTRANS MOBJECT	
BY	Script? INSTR	? INSTR	PTRANS LOC	? INSTR	? INSTR
TO		ATRANS TO	PTRANS TO		
INTO	PROPEL TO	PROPEL TO	PTRANS TO	MTRANS MOBJECT	INGEST TO

For each of these pairs of categories (preposition / noun class), sample sentences are abundant. For example, the pair (OFF / #LOCALE) could arise from the sentence "The man jumped off the building", and the pair (INTO / #CONCEPT) occur in the sentence "He looked into the incident". The idea of "preferring" a certain set of attributes when in the presence of partial information has been used in other systems (see Wilks, 1975). Note that the table is incomplete, and thus can not always provide enough information to allow the correct conceptualization to be chosen from the script. Further work is being done in this area.

e. Step 4: Matching

Recall the story that began this section. Based on the above table, the second sentence, "The car hied into a tree", would be represented as follows:

```
K1: ((ACTOR (K2) <=> (BOGUS-ACT LEXVAL (HIED)
                        PREFER (PROPEL)) TO (K3))

K2: (#PHYSOBJ TYPE (*CAR*))

K3: (#PHYSOBJ TYPE (*TREE*))
```

Note that the ACT "PROPEL" is preferred, and the conceptualization for the noun "tree" is put in the "TO" slot. When APPLY attempts to match this conceptualization into the script, it discovers one tentative match, which has the same noun conceptualizations, corresponding to "car" and "tree", but the one for "tree" is in the OBJECT slot in the script conceptualization, and in the

TO slot in the CD, so the match can only be considered tentative. (This is an example of the idiomatic usage of verb-preposition pairs mentioned earlier.) APPLY now looks at the information in the PREFER slot of the "BOGUS-ACT" place-holder, which says that the ACT should be a PROPEL. Since this is in fact the ACT in the script conceptualization, the match is considered successful, and the word "hied" is defined as building a PROPEL in this context. The definition that FOUL-UP creates for "hied" looks like the following:

```
(PROG NIL
 (DEFPROP HIED T VERB)                                    (1)
 (DEFPROP HIED *CONCEPT* ISA)
 (DC HIED
     (T (NIL (BOGACTFRAME) NIL                            (2)
             ACTORSUGG BYSUGG TOSUGG
             (OBJECT CONREQ) TIMESUGG MODESUGG
             (FROM CONREQ) (TO CONREQ))
        (NIL (THREEFRAME (QUOTE HIED)) NIL                (3)
             SUBJSUGG (OBJ SYNREQ) RECIPSUGG)))
```

Again, the definition is in three parts: part (1) puts the appropriate internal properties "verb" and "concept" on the word; part (2) defines the conceptual (semantic) structure-builders and the expectations for the subsequent slot-fillers; and (3) defines the syntactic expectations and slot-fillers (see Riesbeck and Schank, 1976). Note that all the information provided by the first three steps (verb-frame, preposition match and ACT preference) was in fact required by the matching process before the conceptualization could be recognized as a form of the pattern in the script.

f. Notes on how hard this all is

The process just described depends on the current state of knowledge of (1) prepositions and (2) conceptual classes of nouns. The theories behind each of these are quite primitive (no pun intended), and the above process for foul-ups of verbs is at a correspondingly early stage of development. The problem of resolving the case ambiguity of text prepositions has proved intractable in many natural language systems. General rules for prepositions are riddled with exceptions and special cases, and every preposition seems to have a myriad different uses. For an example of the diversity of uses of a single preposition ("for"), see (Hemphill, 1973). Similarly, the theory behind conceptual classes in SAM is considerably less developed than that of primitive ACTs. Other sets of conceptual classes and object primitives have been proposed in the past, and perhaps the SAM system (and FOUL-UP) would benefit from a re-examination of the current classification system for nouns. (See Weber, 1971, and Lehnert, 1977).

4. Adjectives

Consider the story:

"Friday, a car swerved off Route 69. The flibby vehicle struck a tree."

The meaning of the word "flibby" in the second sentence is not at all clear, nor can it be easily figured out from the context. In fact, it is not clear whether or not the adjective even changes the meaning of the sentence at all, since the sentence can be understood in this case by ignoring the unknown word. In fact, FOUL-UP attempts to ignore unknown adjectives wherever possible, and cannot guess meanings for them in general. The methods used for nouns and verbs totally break down when applied to adjectives. There are three closely related reasons for this:
(1) Adjectives are not expected in a sentence, in the sense of expectation we have been dealing with. Rather an adjective appears as a modifier, affecting the meaning of the noun following it.
(2) There exist no primitives for adjectives corresponding to the relatively comprehensive sets of primitives within the realms of nouns and verbs in CD representation.
(3) The conceptualization built by an adjective usually occupies an "extra" slot in a CD, attached to the slot filled by the noun which the adjective modifies. Thus there is no purely "top-down" way to guess a meaning for an unknown adjective, since it is inherently unexpected.

Note that the ideas of expectation, conceptual classes, and slot-filling are all closely related. The expectations in both ELI and APPLY are based on differentiation between various conceptual classes of objects, and the slots are filled on the basis of the expectations. Thus the lack of conceptual classes of adjectives causes the other difficulties, and they should not be viewed as independent problems.

CONCLUSION: LIMITING CASES

1. Syntactic Class Limitations

It was stated earlier that there might be no theoretical limitation on the classes of words that FOUL-UP could figure out from context, as long as there existed ELI and APPLY expectations for the conceptualizations built by a word. We have now seen, however, that different syntactic classes of words present varying degrees of difficulty to the FOUL-UP program. The difficulties arise depending on (1) the consistency and completeness of the existing representational schemes for a given syntactic class, and (2) the amount of information contained in the structures typically built by words in that syntactic class. Furthermore, we have seen that the issues of expectation on the one hand, and conceptual class on the other, cannot be easily separated. Thus the expectations for the members of certain syntactic classes are stronger than for others, depending on the relative thoroughness of the representation schemes for those classes. Thus strength of expectation is also a (dependent) factor contributing to the ease or difficulty of the FOUL-UP task within a given syntactic class.

Nouns are typically slot-fillers, builders of small structures containing relatively little of the overall information present in a given conceptualization. They also have a reasonably consistent and complete representation in terms of conceptual classes. Thus the process for figuring out unknown nouns from context is relatively straightforward. Verbs are builders of large structures which contain most of the expectations for a given sentence, and which supply most of the structure to conceptualizations. They also have a consistent and complete representation in terms of primitive ACTs. Thus they are more difficult to figure out from context than nouns. Adjectives are not well defined or consistently represented in CD, and they build structures of varying size and complexity. Furthermore, being conceptual modifiers, they lack pre-defined slots to fill in a conceptualization. Thus it is natural that they should be most difficult, if not impossible, to figure out from context.

2. Conceptual Class Limitations

Assuming that all words can eventually have their meanings represented in something akin to Conceptual Dependency representation, then there would still be limitations on FOUL-UP's ability to figure out words from context. To perform the process, a strong top-down context, like a script, is needed to provide the expected attributes of unknown words. Other large top-down frameworks, such as plans (see Schank & Abelson, 1977) or belief systems (see Abelson, 1973), should theoretically provide enough context to enable words to be figured out from those contexts. For example, consider the following "planny" story:

"John saw a menacing figure approaching his store. He reached into the desk drawer and pulled out a magnum."

The word "magnum" in this story is highly expected to be some kind of weapon or other protective device. That expectation could be set up by a plan-applier mechanism (PAM) (see Wilensky, 1976), and a version of FOUL-UP could theoretically use such expectations to guess the meaning of the unknown word. However, scripts are very explicitly defined sequences of actions, while plans are much less explicitly sequenced, and belief systems still less so. It is to be expected that the ability to figure out words from context will diminish as the strictly episodic nature (and thus the strength of the expectations) of the knowledge database diminishes. Thus scripty stories are ideally constructed for figuring out words from context, plans are less so, and belief systems still less. It is worthwhile to reiterate that the representations being discussed are intended as models of the human understanding process, and thus that the abilities and limitations of the programs are modelled after analogous processes in people. The FOUL-UP program was designed to simulate a known human ability, and to add this ability to the already existing SAM system. Since SAM is intended to model the human understanding process within the realm of script-based stories, it is encouraging to note that the limitations of the FOUL-UP seem to closely parallel the limitations of people's ability to perform the same task. Certainly a program intended to model humans will not be able to out-perform them, but rather should show the same sorts of abilities and shortcomings that people show. In that sense, the FOUL-UP program has demonstrated some psychological validity for the SAM system, and for Conceptual Dependency representation itself.

REFERENCES

1) Abelson, R. P. (1973). The Structure of Belief Systems. In R. C. Schank and K. M. Colby (Eds.) Computer Models of Thought and Language. W. H. Freeman, San Francisco, Calif.

2) Cullingford, R.E. (1977). Organizing World Knowledge for Story Understanding by Computer. Ph.D. Thesis, Yale A.I. Project, New Haven, Conn.

3) Gershman, A. (1977). Conceptual Analysis of Noun Groups in English. Paper submitted to the 5th International Joint Conference on Artificial Intelligence. Cambridge, Mass.

4) Hemphill, L. (1975). A Conceptual Approach to Automated Language Understanding and Belief Structures: With a Complete Disambiguation of the Word "For". Ph.D. Thesis, Stanford University, Stanford, Calif.

5) Lehnert, W. (1977). The Process of Question Answering. Ph.D. Thesis, Yale A.I. Project, New Haven, Conn.

6) Riesbeck, C.K. and Schank, R.C. (1976). Comprehension by Computer: Expectation-based Analysis of Sentences in Context. Yale Dept. of Comp. Sci. Research Report #78, New Haven, Conn.

7) Schank, R.C. (1975). Conceptual Information Processing. North Holland, Amsterdam.

8) Schank, R.C. et al. (1975). SAM -- A Story Understander. Yale Dept. of Comp. Sci. Research Report #43, New Haven, Conn.

9) Schank, R.C. and Abelson, R.P. (1977). Scripts, Plans, and Understanding. Lawrence Erlbaum Associates, Hillsdale, N.J.

10) Weber, S. (1972). Semantic Categories of Nominals for Conceptual Dependency Analysis of Natural Language. Stanford A.I. Memo AIM-172, Stanford, Calif.

11) Wilensky, R. (1976). Machine Understanding of Human Intentionality. Proceedings of the ACM Annual Conference. Houston, Texas.

12) Wilks, Y. (1975). Preference Semantics. In E. Keenan (Ed.), Formal Semantics of Natural Language, Cambridge U. P., Cambridge, England.

13) Winograd, T. (1971). Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. TR-84, M.I.T., Cambridge, Mass.

14) Woods, W.A. and Kaplan, R.M. (1971). The Lunar Sciences Natural Language Information System. BBN Report No. 2265. Bolt Beranek and Newman Inc. Cambridge, Mass.