Memory-Based Parsing

by

Michael Lebowitz

Department of Computer Science

Columbia University[1]

406 Mudd Building, New York, NY 10027

[1]Much of the research described here was done while the author as at Yale University, supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract N00014-75-C-1111.

MEMORY-BASED PARSING

by

Michael Lebowitz — Columbia University[1]

Department of Computer Science, 406 Mudd Building

New York, NY 10027

ABSTRACT

The development of robust parsing systems can be greatly expedited by the application of memory-based parsing techniques. Described here are the parsing techniques used by the Integrated Partial Parser (IPP), a system designed to read and generalize from large numbers of news stories. These techniques include top-down predictions generated from high-level memory structures, and simple bottom-up heuristics to handle language-specific problems. A detailed example is presented.

1. Introduction

One important aspect of a complete natural language understanding system, involves the determination of a conceptual representation from the input text. As part of the development of the Integrated Partial Parser (IPP), a system intended primarily to illustrate the process of learning and generalization, several powerful techniques for the extraction of conceptual meaning from text (hereafter "parsing") were developed. This was crucial for the success of the project, since in order for the program to learn by making interesting generalizations, it had to be robust enough to read large numbers of texts that it was not specially prepared.

In this paper I will describe in detail the parsing techniques used by IPP. These techniques are largely independent of the particular program for

---

[1]Much of the research described here was done while the author was at Yale University, supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract N00014-75-C-1111.

which they were developed. The reader interested in other aspects of IPP should refer to [8].

There are two key aspects behind IPP's parsing strategy. The first is the extensive use of top-down processing driven by predictions made from high-level memory structures. A small number of relatively simple, automatically activated, data-driven predictions perform most of the parsing of text. These predictions are quite robust, and carry out the bulk of the parsing task.

The second element of IPP's parsing scheme is to use a relatively small number of simple bottom-up recognition techniques, integrated into the top-down processing, to handle syntactic information. These techniques deal primarily with identifying noun phrases and the initial recognition of memory structures to activate the memory-based predictions. The simplicity of both the top-down and bottom-up techniques used in IPP is crucial for a system that must be robust enough to handle many texts. This strategy is particular successful for IPP, where it is normally only necessary to represent a story in terms of high-level structures.

To illustrate the goal of the parsing process in IPP, I will consider S1, a rather typical news story. (IPP's primary domain is international terrorism, which provides the examples used in this paper.)

S1 - UPI, 18 January 80, Lebanon[2]

A hijacker gunman seized a Middle East Airlines jetliner enroute to Cyprus today, ordered the plane back to Beirut, then surrendered after two hours of negotiations in which he demanded an investigation into the disappearance of a Lebanese Shiite Moslem leader.

The hijacker identified as Fuad Hamade, 19, a Lebanese Shiite Moslem surrendered to police after two hours of negotiations with government ministers and officials of Middle East Airlines.

The parsing task that must be carried out on this story is the relation of the various events in this story such as the hijacking, negotiations and surrender into a single, coherent memory representation. In addition, the various role fillers for these events must be determined. This story can be well-understood in terms of high level memory structures.

Figure 1 illustrates the level of detail that IPP's parsing process extracts from S1. (The slashes in the descriptions of role fillers separate information taken from different parts of the text.)

This representation illustrates that IPP has identified the main actions in the story -- the hijacking, surrender and negotiations -- along with who did what. As I will describe in this paper, this is done almost entirely with information from the memory structures instantiated, independent of the specific words used.

This story also illustrates, however, some limitations of the analysis provided by IPP. The demand of the hijacker, "an investigation into the

---

[2]All the stories used as examples that include a dateline and reference to a newspaper or the UPI wire are actual, unedited news stories. IPP does not require any special preparation of the stories it reads and adds to memory.

```
*(PARSE S1)

Story: S1  LEBANON

(A HIJACKER GUNMAN SEIZED A MIDDLE EAST AIRLINES JETLINER
ENROUTE TO CYPRUS TODAY *COMMA* ORDERED THE PLANE BACK TO
BEIRUT *COMMA* THEN SURRENDERED AFTER TWO HOURS OF
NEGOTIATIONS IN WHICH HE DEMANDED AN INVESTIGATION INTO THE
DISAPPEARANCE OF A LEBANESE SHIITE MOSLEM LEADER)

(THE HIJACKER IDENTIFIED AS FUAD HAMADE *COMMA* 19 *COMMA* A
LEBANESE SHIITE MOSLEM *COMMA* SURRENDERED TO POLICE AFTER
TWO HOURS OF NEGOTIATIONS WITH GOVERNMENT MINISTERS AND
OFFICIALS OF MIDDLE EAST AIRLINES)

Story Representation:

** MAIN EVENT **
EV1 =
 MEM-NAME     S-EXTORT
 ACTOR        HIJACKER GUNMAN / 19 YEAR-OLD FUAD HAMADE
 METHODS
    EV0 =
     MEM-NAME     $HIJACK
     ACTOR        HIJACKER GUNMAN / 19 YEAR-OLD FUAD HAMADE
     VEHICLE      MIDDLE-EAST JETLINER
     TO           CYPRUS
 RESULTS
    EV2 =
     MEM-NAME     GS-CAPTURE-TERRORIST
     OBJECT       HIJACKER GUNMAN / 19 YEAR-OLD FUAD HAMADE
 SCENES
    EV3 =
     MEM-NAME     G$-NEGOTIATE
     ACTOR        OFFICIALS
     OBJECT       HIJACKER GUNMAN / 19 YEAR-OLD FUAD HAMADE
 DEMANDS
    NIL
```

Figure 1: IPP representation of S1

disappearance of a Lebanese Shiite Moslem leader," is hardly typical of terrorist demands. It cannot be understood with the high-level structures IPP uses for representation. Note, however, that this does not imply a failure of IPP's parsing techniques. Rather, it suggests, in conjunction with the overall success of IPP, that the same techniques simply must be applied using

a more detailed set of knowledge structures.

In the remainder of this paper, I will look at the overall structure of IPP's parsing algorithm, followed by a detailed description of both the bottom-up and top-down techniques that are used. I will then present an extended example of IPP in action, and a comparison of this scheme to other AI parsers.

2. IPP's Overall Parsing Design

IPP employs a highly top-down parsing scheme. Memory-based predictions always take precedence over bottom-up processing that might otherwise be done. Issues such as the specific syntax of the text are considered only when needed to initiate top-down processing, or when they affect the application of predictions. This is similar to the philosophy of programs at Yale such as ELI [12], CA [1] and FRUMP [3] (see Section 7).

In order to understand a story, IPP must identify high-level memory structures needed to represent the events in the story. In IPP, events are represented in terms of Action Units (AUs) and Simple MOPs (S-MOPs) [8]. AUs describe concrete events such as shootings, deaths and bombings, while S-MOPs represent more abstract levels of action such as extortion and attacks on people. The aspects of these structures that are relevant to parsing are that AU's contain descriptions of typical fillers for each of their roles, and S-MOPs relate various AUs that serve as methods, results and scenes. Other necessary details will be presented as necessary. In the examples presented, the structures with an S- prefix are S-MOPs, and all other structures are AUs.

To complete the representation of a story, IPP must identify the proper

role fillers for each AU and S-MOP. This information is sufficient to allow the events being described to be recorded in memory so that they can be recalled. The Action Units and S-MOPs describe what happened, and the role fillers represent who was involved.

2.1. Top-down precedence

Top-down predictions are given precedence in IPP's flow-of-control to bottom-up considerations. S2 illustrates the advantage of this scheme.

S2 - UPI, 18 January 80, Lebanon

> A hijacker gunman seized a Middle East Airlines jetliner enroute to Cyprus today ordered the plane back to Beirut then surrendered after two hours of negotiations in which he demanded an investigation into the disappearance of a Lebanese Shiite Moslem leader.

Consider the processing of the word "seized" in S2. Out of context the word is extremely ambiguous, and even in the terrorism domain it can have several meanings — hijack a plane, take over a building, or kidnap an individual. Normal bottom-up processing would activate a mechanism to disambiguate the word by looking at the syntactic object.

However, in this case such a mechanism is clearly not necessary. The description of the actor as a "hijacker gunman" provides the clue as to the meaning of "seized". IPP accomplishes this through a memory-based rule that will be described in detail later. Basically, both "gunman" and "hijacker" have as part of their definitions pointers to hijacking (i.e., that is something they are known to do). This causes IPP to look for action words that describe this action. Since one of the definitions of "seized" describes hijacking, IPP selects that meaning, immediately disambiguating the word.

Expectations in IPP are implemented with requests -- test/action pairs of

the sort described by Riesbeck [11]. The most important group of these requests implementing the memory-based rules to be described in Section 3. Requests may look for lexical items with specified properties, Picture Producers [13] of a given type (to fill roles, generally), or new events that might be expected.

The use of requests and the priority given to expectations leads to IPP's top-level flow of control is shown in Figure 2.

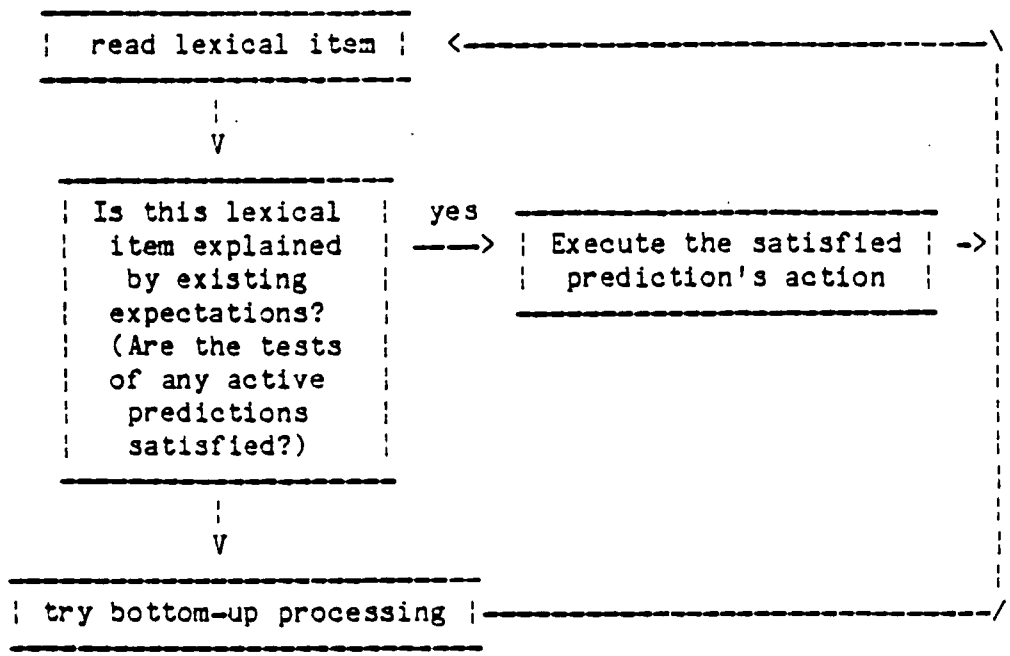


Figure 2: Top level flow of control

The flow chart in Figure 2 indicates that IPP reads a story one lexical item at a time (phrases of more than one word are allowed in the dictionary; henceforth any references to words also applies too multi-word phrases), checking to see whether any active requests are expecting each item. If so, the lexical item is processed according to that request; otherwise, the

bottom-up algorithm described in the next section is used.

While the processing of a story usually begins with largely bottom-up processing, this continues only until enough has been read to create top-down predictions of the sort that will be described in Section 3. In most news stories, this happens very quickly (as in S2 where "A hijacker gunman" was enough to create predictions), allowing knowledge of typical situations to quickly dominate processing.

2.2. Bottom-up processing

Even though top-down processing is extremely powerful, there are still times when there are no active predictions that explain a piece of text and so bottom-up processing is required. Bottom-up processing is especially important when reading the beginning of a story before a context has been established. The goal of bottom-up processing is to create a sufficient conceptual representation of a story to allow memory-based predictions to be made.

In IPP, bottom-up processing is based on a process-oriented classification of lexical items. The words in IPP's vocabulary are broken down into classes that depend on how they add information to the meaning of a story, and the manner in which they should be processed bottom-up. I will describe here the different word classes used by IPP, followed by the required processing for members of each class in bottom-up situations.

2.2.1. Word types

There are five different types of words used by IPP, each requiring different bottom-up processing.

Words in two of the classes provide most of the semantic content of stories. These are Event Builders (EBs), words whose definitions point to specific Action Units in memory, and Token Makers (TMs), that describe Picture Producers, and may point to associated Action Units. Typical EBs are "kidnapped", "hijacking" and "death". TMs include words such as "banker", "hijacker", and "747". Words in each of these classes, and particularly EBs, usually must be processed immediately in order to provide the context from which to create top-down predictions. Notice that while many verbs are EBs, and many concrete nouns are TMs, the classification is based on processing considerations, and class membership frequently crosses syntactic boundaries.

The next two classes of words tend to be less important, but do modify the conceptual content of a story somewhat. These classes are Token Refiners (TRs), words that add information to the conceptual Picture Producers specified by Token Makers, and Event Refiners (ERs), that alter some aspect of an event being described. Token Refiners are words like "old", "Basque" and "broken-down", while typical ERs are "unharmed", "severely", and "often". Words in these two classes are normally not processed until the concepts that they modify are identified.

The final group in IPP's classification of lexical items are Function Words (FWs). These words carry no semantic content of their own and instead provide information for processing. Function Words include articles, prepositions such as "to", "by" and "from" and auxiliary verbs including

"was", "were" and "been".

These five classes of lexical items comprise all those involved in IPP's bottom-up processing. The classes are summarized in Figure 3.

1) Event Builder — SHOT, HIJACKER, KILLING
2) Token Maker — EMBASSY, AMBASSADOR, OFFICIAL
3) Token Refiner — ARABIC, LEFT-WING, TWENTY-TWO
4) Event Refiner — ATTEMPTED, FATALLY, FAILED
5) Function Word — BY, AN, INTO

Figure 3: IPP word types for bottom-up processing

2.2.2. Bottom-up word processing

When an unpredicted lexical item is read, bottom-up processing is used. The processing depends on a lexical item's class. The top-level bottom-up processing algorithm is shown in Figure 4.

Most bottom-up processing revolves around Event Builders and Token Makers. The key aspects to processing EBs bottom-up include the instantiation of Action Units and S-MOPs. The details of this will be discussed in Section 4. One minor aspect of this process is that the short-term memory buffer containing any Event Refiners preceding the Event Builder can be processed fully when the EB is read. In a phrase such as "attempted hijacking ..." this means recognizing that the hijacking did not succeed.

The specific rules for processing EBs are as follows. Assuming there is not an outstanding S-MOP explaining the Action Unit (which would be handled

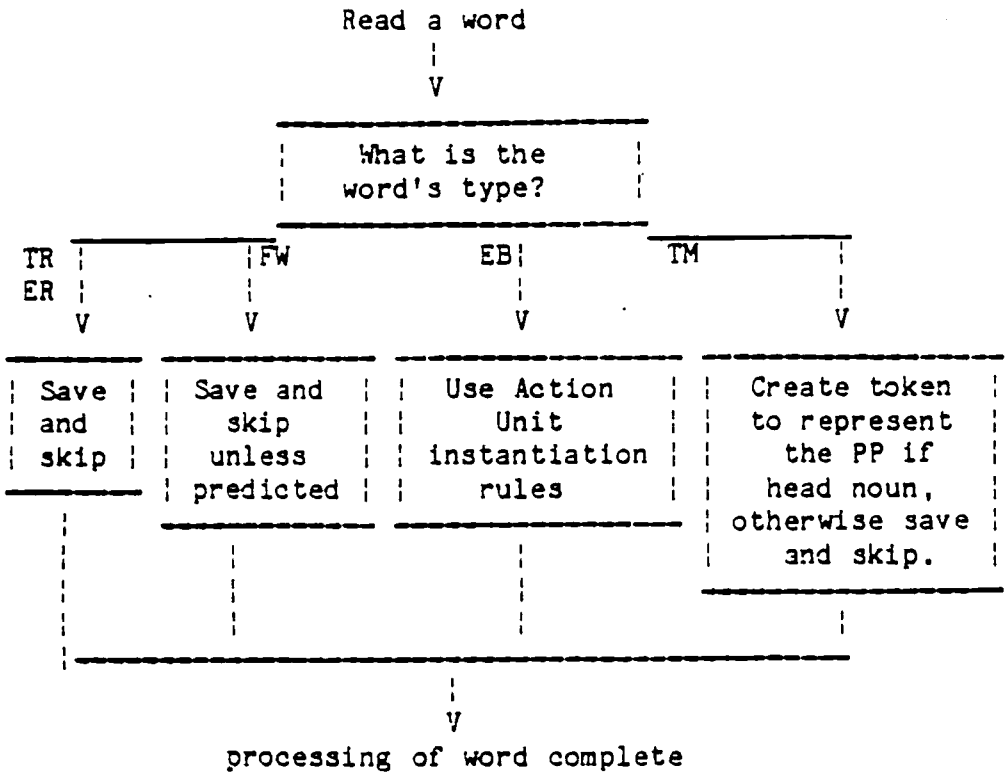


Figure 4: Basic bottom-up IPP flow of control

top-down) then IPP 1) instantiates the Action Unit (creates an internal symbol of this instance) (see Section 4), 2) gathers any modifying Event Refiners from short-term memory (see below), 3) tests any preceding Picture Producers (also saved in short-term memory) for possible slot-filling, 4) activates a request looking for role fillers (using the AU Role Filling Rule) (see Section 3.2.1), and 5), if the AU has a normal S-MOP, instantiates it, attaches the AU to it, and performs S-MOP processing (see Section 4.2).

This relatively simple EB processing is sufficient to identify enough of the actions in a story to create predictions to explain the rest.

Token Makers cause the creation of internal symbols (tokens) to be used to represent Picture Producers described in text (including any that may be

hypothetical). This involves the collection of any preceding Token Refiners that have been saved in short-term memory (in the manner described below), and their application to the token. This processing will be described more fully when I consider the problems of noun groups in Section 5.2.

Once a token has been created for the Picture Producer referred to by a TM, a test is made to see whether the top-down role-filling rule described in Section 3.2.1 is applicable. This rule causes the filling of roles in the memory structures created by Event Builder processing, thereby completing the representation of the story being read.

Both Token Refiners and Event Refiners are simply saved in short term memory buffers until a Token Maker or Event Builder, respectively is found. At that point TRs are applied to the Picture Producer being represented and ERs are used to modify the event description.

This "save and skip" strategy allows modifiers to be processed when most appropriate. Appropriate conceptual processing of a modifier is much easier when we know what is being modified. In English this requires waiting, as both TRs and ERs usually precede the words that they modify.

Function Words are most often processed in a predictive fashion. For example, certain Action Units specify how various prepositions should be treated. The $HIJACK script, for instance, specifies the Function Word "to" will indicate that the filler of the TO role of the script is to follow, and "from" does the same for the FROM role.

By and large, Function Words that are not predicted have little effect on IPP's processing. Prepositions and articles have very little intrinsic

meaning. They usually convey information that is either redundant or changes the meaning of the sentence in only a minor way, too subtle to have an effect at the level of understanding performed by IPP.

At IPP's level of understanding, articles make a contribution to delimiting noun phrases, but can otherwise be ignored. For example, "Armenian gunman" can be understood just as well as "An Armenian gunman". Similarly, prepositions that are not predicted by Event Builders can often be ignored. For example, in the phrase, "the terrorist ambush of an army general," semantic considerations identify the general as the victim of the ambush, without reference to the preposition "of".

The major exception to the general rule of ignoring Function Words, are auxiliary forms of the verb "to be" that indicate a verb is being used in the passive form. Passives are one of the few syntactic constructions that must be paid careful attention. The issue of passives will be discussed in Section 5.1.

In general, the bottom-up processing of IPP is relatively simple, and designed to allow top-down understanding to begin as rapidly as possible. It is the memory-based predictive processing of IPP that provides most of its power and robustness.

3. Memory-based predictions

As indicated in the previous section, IPP always gives precedence to top-down predictions during parsing. The key to using such a scheme is the ability to effectively create predictions early in the processing of a story, without storing an exorbitant amount of information with every word. This is done in

IPP by generation of most predictions from instantiated memory structures.

In this section I will look first at the advantages of generating predictions from memory structures, and then at the specific rules used by IPP to identify memory structures and create top-down predictions from them.

3.1. Why memory-based predictions?

In any particular domain there are a relatively small number of memory structures (S-MOPs and Action Units in IPP). In contrast, there are thousands of lexical items and an unlimited variety of syntactic constructions that can indicate the relevance of those memory structures. Thus, it is certainly desirable to make as much of the processing relatively independent of the specific text being read. This allows most of the necessary information for understanding to be organized around the small number of memory structures instead of the many lexical items. The parsing process need only rely on minimal information from specific lexical items, using them mostly to identify Action Units and Picture Producers.

As an example of this, since we know that a shooting involves a victim, it is more effective to store this information once with the Action Unit $SHOOT rather than many times with each of the lexical items that indicates a shooting occurred. This has the further advantage that if we learn a new piece of information about an AU, such as $SHOOT, it is not necessary to change the definitions of every relevant word. The information is concentrated where we would expect it, with the conceptual descriptions of the various events.

This concentration of information with the conceptual definitions of Action Units and S-MOPs is especially valuable in the design of an

understanding program such as IPP. It allows us to concentrate on providing detailed definitions of the memory structures to provide information to the program, with the large number of words being defined simply.

3.2. IPP's memory-based expectations

IPP uses two general rules to generate memory-based predictions, one concerning Action Units and the other S-MOPs, that largely guide processing. These two rules identify the roles various Picture Producers play in the relevant Action Units, and explain the Action Units in terms of the proper S-MOPs.

When an Action Unit is used to represent a story, it is important to identify how the various roles of the AU are filled by the characters in the story. Instead of using separate predictions based on the words that cause an Action Unit to be instantiated, IPP uses a prediction implementing a rule known as the AU Role Filler Rule that allows the determination of how the roles of each AU are filled. Action Units contain information describing typical role fillers. This is sufficient to determine how the various Picture Producers mentioned in a story fill the roles of Action Units by using top-down expectations largely independent of the specific description of the AU in the text.

The prediction from S-MOPs simplifies the recognition and explanation of Action Units. S-MOP definitions describe the Action Units that are likely to be found as part of the stereotypical situation they describe — methods, results and scenes. This information — employed by the S-MOP/AU Rule — is used to identify Action Units that appear later in the text, including those that may be described using ambiguous words.

3.2.1. The AU Role Filling Rule

Each Action Unit contains information about its various roles, and simple descriptions of the PPs that are likely to fill them, that allows memory-based predictions to be made. For instance, the $HIJACK script has five different roles — the ACTOR of the hijacking, normally a person associated with terrorist type acts, the VEHICLE that is hijacked, typically an airplane, PASSENGERS aboard the plane, a group of ordinary people, and the places the plane was going TO and FROM, usually cities or airports.

By using information about stereotypical role fillers it is often possible to identify the Picture Producers that fill each role, disregarding the specific way an Action Unit such as $HIJACK is presented in the text.

To illustrate this point, I will look at several terrorism stories involving shooting. In these stories the shooting is specified in a variety of different ways, but the details of the action specification can largely be ignored once the memory structure has been identified.

Shooting is represented in IPP with the AU, shown in Figure 5. The important aspect of $SHOOT here is that it includes information about the various role fillers that are expected in a description of a shooting. From this definition, it can be recognized that every time a shooting is mentioned, there may also be mention of the actor, the victim, the weapon used in the shooting, and the part of the body in which the victim was shot.

Each of the four stories below describes a shooting incident, and each time the shooting is presented in a somewhat different way. The first two stories both use the verb "firing" to mean shooting, but S3 has the pistols

```
(DEF-AU $SHOOT
  AU-TYPE           SCRIPT
  TEMPLATE          ($SHOOT ACTOR      !PROTO-TERRORIST
                            VICTIM     !PROTO-VICTIM
                            WEAPON     !PROTO-GUN
                            HURT-PART  !PROTO-BODYPART)
  NORMAL-S-MOP      S-ATTACK-PERSON
  PASSIVE-SLOT      (SUBJECT is ACTOR => VICTIM role
                     SUBJECT is WEAPON => WEAPON role]
```

Figure 5: $SHOOT definition

used as the direct object while the second example never mentions a weapon at all. The third story uses a completely different verb, "shot" and the fourth has no verb at all that indicates a shooting, but instead has a noun, "gunman" that strongly implies that event.

All four of these stories can be understood using a single, memory-based role filling rule, rather than specialized definitions for each action word.

S3 - UPI, 12 February 80, Italy

Red Brigades guerrillas firing silenced pistols killed a prominent judge with strong ties to the Vatican while he was at a conference on terrorism today.

S4 - UPI, 12 March 80, Puerto Rico

Three terrorists firing from a car attacked a US army vehicle carrying three military science professors early today on San Juan's busiest freeway.

S5 - Washington Star, 23 December 79, France

Terrorists armed with submachine guns yesterday shot and killed Turkish embassy press attache Yilmaz Kolpan, 31, on the Champs Elysees in a decades-old international feud.

S6 - New York Times, 11 November 79, Northern Ireland

A suspected Irish Republican Army gunman killed a 50-year-old unarmed security guard in East Belfast early today, the police said.

Each of the emphasized action words in these stories gives direct access to the Action Unit $SHOOT. From this it is immediately known that the text is likely to make reference to the actor of the shooting, who is likely to fit into a memory class of people we expect to do such things, the victim, who can be just about any person, and the weapon used.

Using this information it is a relatively simple matter to process all of these stories. In the first of these stories, the syntactic subject fits perfectly the stereotyped description of an actor for $SHOOT, the direct object "silenced pistols" is clearly the weapon, and "a prominent judge" must be the victim.

In the next story, the subject is again the actor of a $SHOOT, but this time the phrase after the verb "firing", "from a car attacked a US Army vehicle" does not describe any of the items involved in $SHOOT, and so for many understanding purposes can be largely ignored. Eventually there is something we are more interested in, "three military science professors", that is suitable for filling the victim role.

In S5, the verb that identifies $SHOOT is different — "shot". Nonetheless, it is still possible to pick out the same major elements of importance from the story — the actor (terrorists), weapon (submachine guns), and victim (Yilmaz Kolpan). (The final role presents some interesting noun group problems that will be discussed later.) This despite the fact that "shot" has different syntactic properties than "firing".

The point here is that the same roles must be filled for an Action Unit no matter how it is expressed in the text. Assuming we have a good idea of

what sort of filler is likely to be found, we can ignore any special peculiarities of the specific action words in the story and simply use this knowledge of the stereotypical situation to identify the role of each Picture Producer.

Memory-based role identification can also occur when an Action Unit is instantiated in ways other than a simple verb. In the final shooting example, S6, the Action Unit $SHOOT is identified primarily from a noun, not a verb. "Gunman" indicates a shooting can be expected, and the existence of one of $SHOOT's outcomes ("killed") strongly implies that there was indeed a shooting (see Section 4.1.3). Once this has occurred, the important roles, the actor and victim in particular, can be identified.

These stories indicate that a feasible plan of action for a parser is to assume that whenever an Action Unit is instantiated the necessary role fillers will basically fit the known stereotype. This allows it to simply look in the text for Picture Producers that fit the descriptions of any missing roles for the AU, and ignore the specific nature of the action words in the text. This plan is the one used by IPP, and is able to deal with all the above examples, as well as most other role filling problems. It is referred to as the AU Role Filling Rule.

> Whenever an AU is instantiated in a piece of text, assume its role fillers will fit the stereotypes in memory. Then check each new PP to see if it can be a role filler of the AU.
>
> This process is terminated at the next action word.

Figure 6: The AU Role Filling Rule

The termination test in the AU Role Filling Rule is worth noting. It

concentrates upon avoiding picking up irrelevant PPs as role fillers, rather than finding every conceivable filler. It relies on the fact that by using the S-MOP based inference rules described below, identifying one role of a PP fills is usually sufficient to infer all its roles. This can be seen in S7.

S7 - UPI, 12 June 80, Guatemala

Unidentified gunman shot and killed a leader of Guatemala's Christian Democratic Party in a street ambush early Thursday, authorities said.

The issue in this story is identifying that the Guatemalan politician was the person shot. The AU Role Filling rule easily identifies him as the person killed, and using an S-MOP (S-ATTACK-PERSON) we can invoke the inference rule that the person shot in an attack is also likely to be the person killed. Thus it is not necessary to extract directly from the text that he was the shooting victim. This illustrates how the AU Role Filler Rule can afford to be limited to the most reliable cases, as there are alternate, redundant sources of information to determine fully who did what. We need only extract directly from the text the role fillers that can be determined easily.

S8 is a typical story in which the AU Role Filling Rule is sufficient to explain the roles of all the PPs. I will use it to illustrate IPP's role filling processing.

S8 - UPI, 24 July 80, Bahrain

A Jordanian gunman hijacked a Kuwaiti jetliner with 112 passengers aboard Thursday and forced it to fly to Bahrain.

He released women and children passengers in Kuwait.

Hijackings tend to provide especially good tests of the AU Role Finding Rule, as there are a number of different roles to be found. Figure 7 lists the

various roles of $HIJACK, along with a brief description of a typical role filler.

```
ACTOR        [terrorist]
PASSENGERS   [group of people]
VEHICLE      [airplane]
TO           [city or country] [prep "to"]
FROM         [city or country] [prep "from"]
```

Figure 7: $HIJACK roles

Notice that the roles the AU Role Filling Rule is concerned with are only those particular to $HIJACK and that can be filled with Picture Producers. More abstract role fillers, such as the demands of the hijacker, are common to all forms of extortion, and hence processed at the S-MOP level. Also note that the description of the TO and FROM roles indicates that the prepositions "to" and "from" will often be used to distinguish those roles. This information holds without regard to the specific way $HIJACK is instantiated, indicating that information about the way a given natural language uses Function Words can be associated with Action Units.

Figure 8 shows IPP parsing the first section of S8, up to the point where $HIJACK is instantiated.

When it reads this story, IPP instantiates $HIJACK using a rule that will be described in section 4.1.2. Basically, it simply knows that the word "hijacked" indicates an occurrence of $HIJACK, with the syntactic subject (if it is a person or group) filling the actor role of that AU.

The important thing to notice here is that when $HIJACK is instantiated,

```
*(PARSE S8)

Story: S8  BAHRAIN

(A JORDANIAN GUNMAN HIJACKED A KUWAITI JETLINER WITH
112 PASSENGERS ABOARD THURSDAY AND FORCED IT TO FLY TO
BAHRAIN)

(HE RELEASED WOMEN AND CHILDREN PASSENGERS IN KUWAIT)

Processing:

A              : Function word - Token refiner - save and skip
JORDANIAN      : Token refiner - save and skip
GUNMAN         : Interesting token - JORDANIAN GUNMAN

Predictions - LOOK-FOR-GUNMAN-ASSOCIATED-AU
              FIND-GUNMAN-ASSOC-SIBLING

HIJACKED       : Word satisfies prediction

Prediction confirmed - LOOK-FOR-GUNMAN-ASSOCIATED-AU

>>> Instantiated $HIJACK structure

Predictions - $HIJACK-ROLE-FINDER REDUNDANT-AU-WORDS

*** Filling slot -> $HIJACK, ACTOR, JORDANIAN GUNMAN

>>> Instantiated S-EXTORT structure

Predictions - S-EXTORT-RELATED-AUS

*** Filling slot -> S-EXTORT, ACTOR, JORDANIAN GUNMAN
```

Figure 8: Instantiation of $HIJACK

a top-down expectation is activated implementing the AU Role Filling Rule. It is called $HIJACK-ROLE-FINDER in the output in Figure 8. This expectation will cause IPP to check each incoming PP to see if it fits any of the as yet unfilled roles of $HIJACK.

Figure 9 shows this expectation being used several times.

```
A               : Function word - Token refiner - save and skip
KUWAITI         : Token refiner - save and skip
JETLINER        : Normal token - JETLINER

Prediction confirmed - $HIJACK-ROLE-FINDER(VEHICLE)

*** Filling slot -> $HIJACK, VEHICLE, KUWAITI JETLINER

WITH            : Function word - save and skip

112             : Token refiner - save and skip
PASSENGERS      : Normal token - PASSENGERS

Prediction confirmed - $HIJACK-ROLE-FINDER(PASSENGERS)

*** Filling slot -> $HIJACK, PASSENGERS, 112 PASSENGERS

*** Filling slot -> S-EXTORT, HOSTAGES, 112 PASSENGERS

ABOARD          : Skip
THURSDAY        : Normal token - THURSDAY
```

Figure 9: Plane and passengers found

In the segment of processing shown above, IPP finds the PPs "Kuwaiti jetliner" and "112 passengers", and checks to see whether they fit the top-down expectation for role fillers of $HIJACK. In each case, they do. The jetliner matches the description for the vehicle role — it is a plane — and the passengers are a group of people, matching the description for the passenger role. Notice that subtleties such as the function word "with" indicating that the 112 people are inside the plane can be dispensed with, since what we are concerned with is who is being held hostage, and that can be determined by the AU Role Filler Rule.

Figure 10 shows the conclusion of the processing of the first sentence of S8, including the identification of the destination of the hijacking.

```
AND             : Function word - conjunction - save and skip
FORCED          : Dull verb - skipped
IT              : Skip
TO              : Word satisfies prediction

Prediction confirmed - $HIJACK-SYN-FINDER-TO
Predictions - GET-SYN-FILLER

FLY             : Dull verb - skipped

Prediction deactivated - GET-SYN-FILLER

TO              : Word satisfies prediction

Prediction confirmed - $HIJACK-SYN-FINDER-TO
Predictions - GET-SYN-FILLER

BAHRAIN         : Normal token - BAHRAIN

Prediction confirmed - GET-SYN-FILLER

*** Filling slot -> $HIJACK, TO, BAHRAIN

...  [the second sentence is processed in the same fashion]
```

Figure 10: Roles filled in $HIJACK

The TO and FROM roles of $HIJACK can be filled with similar conceptual items. Hence in some cases it does become necessary to pay attention to simple syntactic rules, such as function words specifying the role to be filled. However, once again it is possible to organize this information under a few memory structures rather than many lexical items.

In the output above we can see IPP find the function word "to", which is also part of the top-down role filling expectation (as part of the description of the to role), found twice. The first time it is not followed by a PP that can fill the TO role of $HIJACK, but the second time it is, and so IPP fills that role with "Bahrain".

IPP's processing of S8 as shown above produced the representation shown in Figure 11, which successfully explains all the PPs in the story.

```
Story Representation:

** MAIN EVENT **
EV1 =
 MEM-NAME     S-EXTORT
 ACTOR        JORDANIAN GUNMAN
 HOSTAGES     112 PASSENGERS / WOMEN AND CHILDREN PASSENGERS

 METHODS
    EV0 =
     MEM-NAME     $HIJACK
     ACTOR        JORDANIAN GUNMAN
     VEHICLE      KUWAITI JETLINER
     TO           BAHRAIN
     CARRYING     112 PASSENGERS / WOMEN AND CHILDREN
                  PASSENGERS

 RESULTS
    EV2 =
     MEM-NAME     GS-RELEASE-HOSTAGES
     ACTOR        JORDANIAN GUNMAN
     OBJECT       112 PASSENGERS / WOMEN AND CHILDREN
                  PASSENGERS
```

Figure 11: Final representation for S8

The examples in this section have illustrated that in situations where we have a great deal of knowledge about what to expect, such as the stories in IPP's domain, it is possible to explain the roles of most PPs in stories without making use of detailed knowledge about the specific words involved. While it is possible to construct examples that will not be correctly understood using the AU Role Filling Rule, use of domain-dependent information accurately allows IPP to be successful in a large percentage of stories describing relatively stereotyped situations.

3.2.2. The S-MOP/AU Rule

S-MOPs, once instantiated by IPP, are used as a source of predictions concerning Action Units that may appear in the text. In particular, a prediction is made that any of the Action Units serving as an S-MOP's methods, results and scenes might be found in the text. When these AUs are found, their roles as part of the S-MOP immediately explain their presence in the story as part of an abstract stereotypical situation represented by the S-MOP.

S9 illustrates the kind of situation for which knowledge of an S-MOP's related AUs is used in processing.

S9 - UPI, 21 April 80

A bomb exploded in front of the Iraqi Culture and Information Ministry in Baghdad killing an Iraqi policeman and injuring several people, the Kuwaiti news agency quoted Baghdad radio as saying today.

The radio said the bomb was placed by an Iranian agent who was arrested by another policeman near the building.

In this story, the mention of a bomb explosion immediately causes the S-MOP S-DESTRUCTIVE-ATTACK to be instantiated, in a manner described in Section 4.2. This in turn creates the prediction that any of S-DESTRUCTIVE-ATTACK's other methods, scenes, and results may be described in the story. Then when the Event Builders, "killing", "injuring" and "arrested" are found, they are analyzed as meaning Action Units that should be considered part of the S-DESTRUCTIVE-ATTACK. The details of this identification will be described shortly.

This S-MOP based parsing by IPP leads to the following representation.

Notice in Figure 12 how each of the three Action Units, CAUSE-DEATH,

```
*(PARSE S9)

Story: S9  IRAQ

(A BOMB EXPLODED IN FRONT OF THE IRAQI CULTURE AND
INFORMATION MINISTRY IN BAGHDAD KILLING AN IRAQI POLICEMAN
AND INJURING SEVERAL PEOPLE THE KUWAITI NEWS AGENCY QUOTED
BAGHDAD RADIO AS SAYING TODAY)


(THE RADIO SAID THE BOMB WAS PLACED BY AN IRANIAN AGENT
WHO WAS ARRESTED BY ANOTHER POLICEMAN NEAR THE BUILDING)

Story Representation:

** MAIN EVENT **
EV1 =
 MEM-NAME     S-DESTRUCTIVE-ATTACK
 ACTOR        IRANI AGENT
 TARGET       CULTURE AND INFORMATION MINISTRY
 WEAPON       BOMB
 VICTIM       IRAQI POLICEMAN

 METHODS
    EV0 =
     MEM-NAME     $EXPLODE-BOMB
     ACTOR        IRANI AGENT
     TARGET       CULTURE AND INFORMATION MINISTRY
     BOMB         BOMB

 RESULTS
    EV2 =
     MEM-NAME     CAUSE-DEATH
     ACTOR        IRANI AGENT
     VICTIM       IRAQI POLICEMAN
     HEALTH       -10

    EV3 =
     MEM-NAME     CAUSE-WOUND
     ACTOR        IRANI AGENT
     VICTIM       MORE THAN 2 PEOPLE
     HEALTH       -5

    EV4 =
     MEM-NAME     GS-CAPTURE-TERRORIST
     ACTOR        POLICEMAN NEAR BUILDING
     OBJECT       IRANI AGENT
```

Figure 12: S9 parsed by IPP

CAUSE-WOUND and GS-CAPTURE-TERRORIST, in addition to $EXPLODE-BOMB, have been

instantiated and attached to the S-DESTRUCTIVE-ATTACK, explaining these events.

Figure 12 illustrates another advantage in using S-MOPs to explain the presence of Action Units in a story. The representation of S9 lists the Irani agent captured by the police as the actor in the bombing, killing and wounding, despite the fact that none of this is clearly stated in the story. Without using existing knowledge about attacks, this role of the agent would have to be inferred through a complex inferential chain involving understanding the part of the second sentence describing the placing of the bomb and using a rule that the person who places a bomb is responsible for its explosion.

IPP, however, uses a simpler rule. Based on its stereotypical knowledge of destructive attacks in the abstract captured in the S-MOP, it can infer that person who was captured by the authorities is likely the actor of the attack, who is in turn the actor of the method (the bombing in this case), and responsible for any of the results (the killing and injuring in S9).

The basic idea here is that by using the stereotypical information contained in S-MOPs it is possible determine the role fillers for many Action Units without specifically identifying them in the text, or using a complex inference process. This allows the AU Role Filling Rule to concentrate on avoiding errors, rather than making a large computational effort to identify every role filler.

Role filling inferences are very common and required throughout parsing. It is handled in a very clean fashion by IPP's use of S-MOPs.

The solution implemented in IPP for recognizing AUs involves a rule that examines the definitions of incoming lexical items, looking for words with meanings that correspond with any of an instantiated S-MOP's methods, scenes or results. The situation is illustrated schematically in Figure 13.

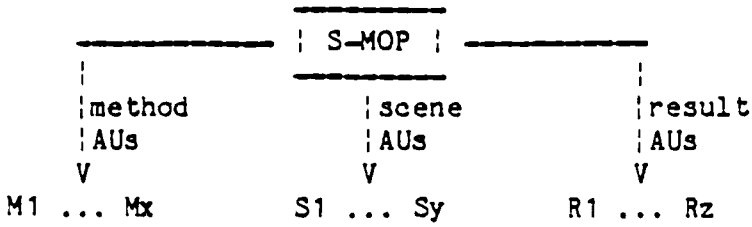


Figure 13: S-MOP related AUs

Figure 13 represents an S-MOP with x different methods, M1 - Mx, y scenes, S1 - Sy and z results, R1 - Rz. Whenever this S-MOP is instantiated, IPP's rule for identifying AUs will cause an examination of each new word to see if it has among its meanings a pointer to any of the Action Units M1 - Mx, S1 - Sy or R1 - Rz. If so, that meaning is assumed to be the correct one for that word, the AU is instantiated and attached to the S-MOP as a method, result or scene, as appropriate.

The specific rule, known as the S-MOP/AU Rule, is shown below.

> Whenever a situation represented by an S-MOP is identified, predict that its methods, scenes and results will be mentioned in the text.
>
> Use this prediction to specify the appropriate meanings for ambiguous words and to determine the relations between new action words and instantiated S-MOPs.

Figure 14: The S-MOP/AU Rule

This rule is a simple and yet extremely powerful way to apply knowledge

of abstract stereotypical situations in parsing. There are only a small number of abstract situations, S-MOPs, relevant to a particular domain. Once one has been identified, the analysis of many Event Builders, including complex and ambiguous ones, becomes very simple. Their underlying meaning and relation to the relevant S-MOPs can be determined all at once.

The advantage of the S-MOP/AU Rule, then, is that it allows the easy identification and explanation of Action Units described in text. In addition, it makes extremely simple an important kind of inference involving PPs with roles in more than one AU explained by a single S-MOP. A further example of the S-MOP/AU rule in action can be found in Section 6.

4. Identifying memory structures in the text

I have illustrated in the previous sections of this paper that it is possible to make useful top-down predictions based upon the memory structures used to represent a piece of text. However, for this process to be useful, it must be possible to instantiate some of these structures initially.

Specifically, many Action Units can be identified directly in news stories. I will present five specific methods that IPP uses to recognize AUs. The first four of these methods depend upon the ability of various words to include in their definitions pointers to Action Units in the same fashion as words point to Picture Producers. This will be discussed further below.

S-MOPs are generally inferred from Action Units. Certain key Action Units appear only in certain situations, and hence can identify the relevant S-MOPs.

4.1. Recognizing AUs

Since Action Units describe concrete events, they normally must be identified in order to represent what is actually happening in a story. Once the first AUs in a story have been identified, it is possible to infer more abstract structures, such as S-MOPs, which can in turn help recognize later AUs.

For texts of the sort IPP deals with, there are five distinct methods for identifying Action Units, including the S-MOP/AU rule, which is often crucial in the most difficult cases. The five AU identifying rules are listed in Figure 15. I will look at each in turn.

These five methods are the only ways that Action Units can be instantiated in IPP's scheme of memory-based parsing. While there are undoubtedly other situations where Action Units need to be inferred, perhaps from lower level primitives, limiting their instantiation to the five situations in Figure 15 allows for a clearly specified process, and seems to cover the large majority of situations.

At this point I will describe each of the five situations described in Figure 15 in a bit more detail, and explain how AUs are instantiated in each case.

4.1.1. Explicit mention

The most frequent way that Action Units are recognized in text is from words that unambiguously mention them. Many action words, the Event Builders (EBs) described in Section 2.2.1, have as their definitions pointers to particular AUs. These include obvious words such as "shot", "hijacked" and

1 - There is an explicit mention of an AU by an
Event Builder in the text.
(exs: A Serbian nationalist hijacked ...,
A twelve-year-old boy shot ...)

2 - A Token Maker with an associated AU is followed
by an Event Builder describing that AU.
(exs: A Russian hijacker took over ...,
Six gunmen attacked ...)

3 - A Token Maker with an associated AU is followed
by an Event Builder with a related AU.
(exs: Four bombs killed three bystanders ...,
The Moluccan hijackers negotiated ...)

4 - An Event Builder describing an AU predicted by an
instantiated S-MOP is found.
(exs: Terrorists shot and killed ...,
After hijacking a 747, the gunman released ...)

5 - There is a default AU specified by an S-MOP.
(exs: Three businessmen were killed by terrorists.
(The default is that they were shot.)
Members of the Red Brigades wounded six
executives ...
(The default is that they were shot in
the legs.))

Figure 15: Methods for recognizing AUs in a text

"negotiated". When such a word is found, the specified AU is instantiated. Many of these words are verbs, but other words that are syntactically nouns, such as "death", "negotiation" and "hijacking" also fall into this class.

It is not at all surprising that words should include AUs in definitions. Since Action Units are in effect "conceptual pictures," there is no more reason these pictures should not be identifiable by words than the Picture Producers of Conceptual Dependency [13]. Just as "book" or "Ronald Reagan" point directly to descriptions of concrete objects, words such as "hijacked" and "shot" point to Action Units.

Figure 16 lists a few words in English that point to each of several different AUs.

CAUSE-DEATH
murdered, slayed, dead, died, deaths, killed
slain, executed, assassination, assassinated

$HIJACK
hijacked, boarded, commandeered, piracy, diverted

GS-CAPTURE-TERRORIST
captured, arrested, surrendered

$SHOOT
gunned, shooting, shot, machinegunned, raked,
fired, submachine-gunned, firing, sprayed

$EXPLODE-BOMB
explosion, exploded, blew up, bombed, firebombed
bombing, blasted, detonated

Figure 16: Assorted EBs and the AUs they point to

Notice from Figure 16 how an AU like CAUSE-DEATH can be described by words with totally different syntactic properties - "deaths" and "killed" for example. However, they all point to the same Action Unit, and, using the AU Role Filling Rule, it is possible to determine the various important role fillers largely disregarding the specific action word in the story.

S10 is a typical example of the direct form of Action Unit instantiation.

S10 - UPI, 14 July 80, Mexico

Five students were shot to death and 15 others were wounded by hooded gunmen who sprayed submachine gun fire at the volatile San Carlos University in a mid-morning raid Monday authorities said.

In this story, "shot" unambiguously causes $SHOOT to be instantiated. In cases such as this it is not hard for an understander using AUs as a parsing

target to determine the actions in the story. The definition of "shot" is shown in Figure 17.

```
(DEF-EB SHOT
        AU           = $SHOOT
        USE-SUBJECT = if ACTOR then ACTOR role
        SYN-TYPE     = PP)
```

Figure 17: Definition of "shot"

This definition specifies that when "shot" is read, $SHOOT should be instantiated, the syntactic subject, if a person or group, should fill the actor role (in S10 the "unidentified couple" is the actor), and the word is a past participle (i.e., can be made passive).

In this particular example, "shot" is in the passive form, so that the use of the syntactic subject must be modified. Passives are one of the few syntactic constructions that must be paid close attention to, and their processing will be outlined in Section 5.1. The only point I would like to make here is that the processing of passives depends almost entirely on the Action Unit being built, and not the specific verb. So virtually any passive verb that points to $SHOOT, for example, will use the syntactic subject in the same fashion.

4.1.2. TM - EB combinations

Event Builders are not the only words that suggest the actions being described in a story. Many words that primarily describe Picture Producers, known as Token Makers (TMs), also have Action Units associated with them. Words such as "killer", "bomb" and "gunman" strongly indicate that CAUSE-DEATH, $EXPLODE-BOMB and $SHOOT or $SHOOT-ATTACK will be events mentioned in

the story.

It is not normally possible to instantiate an associated AU directly from the Token Maker, since there is no guarantee the AU describes an event that actually took place. For example, we might find a story such as S11.

S11 - In Rome today, three terrorist gunmen were sentenced ...

While there is an implication in S11 that the terrorists did shoot someone, that is not the event being described in the story, and should not be at the center of our representation. Furthermore, we would not want to expect to immediately hear about the things normally associated with a shooting, such as the victim or weapon.

Despite these caveats, any confirming evidence for a TM-related AU will cause the AU to be instantiated. This evidence can take two forms, one of which I will discuss in this section, and the other in the next.

The easiest way for a TM-associated AU to finally be instantiated is simply to find an Event Builder later in the story that points to one of the TM's associated AUs. The rule used in such cases, called the Predicted AU Instantiation Rule, is shown below.

When a Token Maker with one or more associated AUs is found, predict that an Event Builder pointing to one of those AUs will appear later, disambiguating in favor of such a meaning if necessary.

If such an EB is found, instantiate the AU.

Figure 18: The Predicted AU Instantiation Rule

S12 is an example requiring this rule's application.

S12 - UPI, 23 July 80, Lebanon

Gunmen today shot and killed Riyad Taha, president of the Lebanese newspaper publishers' association, and his driver in an ambush.

Taha, 54, a Shiite Moslem was shot as he was going to his office in predominantly Moslem west Beirut.

In S12, "gunman" has the associated AU $SHOOT. Since the definition of "shot" also points to this AU, the prediction from the Predicted AU Instantiation Rule is satisfied, and $SHOOT is immediately instantiated.

Of course in this case, the gain in using the information from the TM is rather slight, as "shot" could cause the instantiation of $SHOOT by itself (as in the previous section). However, the TM knowledge becomes more important when the confirming action word is less specific than in this case, since as mentioned in the definition of the Predicted AU Instantiation Rule, we always disambiguate words in favor of the associated AUs.

S13 illustrates this process.

S13 - UPI, 14 July 80, El Salvador

Heavily armed gunmen believed to be leftist guerrillas attacked the National University early Monday but were beaten back by army troops occupying the campus for the past three weeks witnesses said.

The word "attacked" is a general term that in the terrorism domain can describe a variety of different acts of violence -- shooting, bombing and so forth. In IPP it has several senses pointing to the different Action Units that are relevant. In S13, the word "gunmen" with its associated AU $SHOOT makes the disambiguation of this word a simple matter. So while neither "gunmen" or "attacked" by itself can cause the instantiation of $SHOOT, the combination can.

This disambiguation ability can also apply in situations in which a word has a secondary definition that would not normally be considered. So for instance, while "sprayed" does not normally mean $SHOOT, in the context, "gunmen sprayed ..." it certainly does. This can be recognized using the disambiguation rule described above, assuming $SHOOT is listed as a secondary meaning for "sprayed". (Learning that "sprayed" can mean $SHOOT the first time is a problem I will not discuss here.)

There is also a situation in which TM's with associated AUs can be useful even when the Event Builder found later provides an unambiguous pointer to the same AU. This comes about because in addition to knowing what AUs are related to an TM, we also know what role the PP described by the TM will play in the Action Unit. This can be quite useful in syntactically ambiguous situations.

Consider, for instance, the following two story openings.

S14 - An elderly man shot yesterday ...

S15 - Three gunman shot yesterday ...

In S14, we would undoubtedly assume that "shot" was being used in an implicitly passive sense (partially because it is not followed by an object), and that the man was actually the victim of the shooting.

For S15, on the other hand, since we know "gunman" normally acts as the actor of $SHOOT, we will assume that this is in fact the case in this story. Thus by using the memory-based knowledge of the AUs related to this Token Maker, it is possible to avoid any serious syntactic disambiguation.

Finally, the AUs related to a TM do not all have to be part of the word's

literal definition, as is the case for "hijackers." It is possible that a reader may learn over time that other events should be expected along with the definitional ones. For instance, in IPP, "gunman" has associated with it $KIDNAP and $HIJACK, among others, as well as $SHOOT.

So in situations where Event Builders do not provide unambiguous identification of Action Units, it is often possible to use Token Makers that have associated AUs to clarify the situation.

4.1.3. TM and related EB

Stories with TM-associated AUs frequently do not contain an Event Builder that explicitly points to the AU. Instead, the presence of the AU must be inferred from related events. For instance, it is immediately inferred from "the sniper wounded ..." that the wounding was a result of shooting, despite the fact this is not mentioned directly.

The rule used, called the Related AU Instantiation Rule, is presented below.

> If a Token Maker is found with an associated AU that has a role in a given S-MOP, predict that the other methods, scenes and results of that S-MOP will be mentioned.
>
> If one does, assume that the associated AU also took place.

Figure 19: The Related AU Instantiation Rule

This rule means that as well as predicting the AUs directly associated with a TM (as in the Related AU Instantiation Rule), we also expect EBs describing related AUs. Figure 20 illustrates the situation abstractly. Assuming a TM has been found with the associated AU X1 (which can be a method, result, or scene, although method is the most common), IPP begins to check

each new Event Builder to see of it describes any of the AUs M1 - Mx, S1 - Sy or R1 - Rz. If so, it assumes that both that AU and X1 did occur. This check is not too computationally taxing as the number of S-MOP-related AUs is not large, and any given Event Builder is likely to have only one or two possible AU definitions.

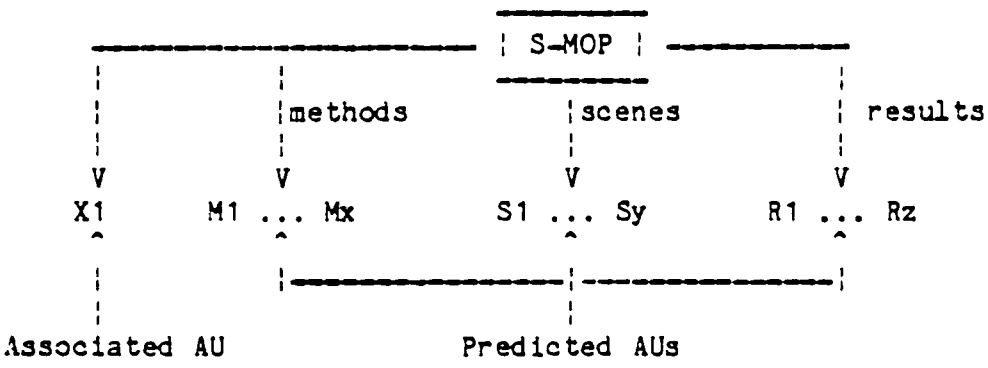


Figure 20: S-MOP related AUs

The Related AU Instantiation Rule is illustrated by the two examples below.

S16 - UPI, 3 February 80, El Salvador

Unidentified gunmen killed two people and wounded 15 others in a lightning attack on a church in downtown San Salvador, witnesses said Sunday.

S17 - Boston Globe, 17 January 1979, Lebanon

Six Moslem hijackers released all 66 passengers and nine crew members of a Lebanese airliner early today ending a seven-hour drama they had staged to protest the disappearance of a religious leader Imam Mousa al-Sadr.

In the first example, the combination of the TM "gunman" with its associated AU $SHOOT, and the EB "killed" with its related AU CAUSE-DEATH result in $SHOOT being inferred, despite never being explicitly mentioned in

the text. Since CAUSE-DEATH is a result of S-ATTACK-PERSON, and $SHOOT is a method, the Related AU Instantiation Rule can be applied, implicitly causing the inference to be made that the attack took the form of a shooting.

S16is similar, except the Action Unit specifically mentioned in the text, G$-RELEASE-HOSTAGES (from "released") is a scene rather than a result. Again the method, $HIJACK, can be inferred using the Related AU Instantiation Rule, since $HIJACK is an associated AU of the PP "hijacker".

4.1.4. S-MOP predicted AUs

The fourth way that AUs can be instantiated is from a combination of EBs and predictions from already instantiated S-MOPs. This is a result of the S-MOP/AU Rule described in Section 3.2.2. The only point I will reiterate here is that this rule can be important in some of the same cases where TM-associated AUs were important — when an EB's definition includes more than one AU that can be built, or for words that only imply an AU when strongly in context.

4.1.5. Defaults

The final way an Action Unit can be instantiated involves a case in which it is not actually mentioned at all in the text. There are situations for which we have default expectations about the AUs that are likely to be present, and if there is nothing to contradict that expectation, then we assume it to be correct.

IPP starts out with a small number of such defaults that are needed to understand news stories. An example would be that when we are told of an attack against a person (i.e., the S-ATTACK-PERSON S-MOP is instantiated), and do not hear about the results, then we assume the victim died. S18 is an

example of such a story.

S18 - UPI, 28 May 80, Italy

Two spectacular terrorist attacks in Rome and Milan Wednesday jolted hopes that police finally were winning against Italy's 10-year wave of terrorism.

In Rome a commando squad of five right-wing gunmen shot three policeman before scores of high school students.

...

If this story ended at this point we would assume that the three policemen had been killed in the attack. If the story specified their condition more explicitly, the result Action Unit would override the default. However, since none is specified, the default result, CAUSE-DEATH, for an S-ATTACK-PERSON is instantiated.

As described in [8], defaults can often be learned by IPP. For example, the program might generalize that shootings of businessmen in Italy usually result in their being wounded (and not killed). Such generalizations are then applied in the same manner as the built-in defaults provided to the program.

4.2. Instantiation of S-MOPs

In order to use the S-MOP/AU rule presented in Section 3.2.2, IPP must quickly instantiate S-MOPs as well as AUs, despite the fact that S-MOPs are not normally mentioned explicitly in a piece of text. We are not told directly that terrorists attacked a person. Instead we hear that they shot the person, or perhaps that they killed him, and we have to infer S-ATTACK-PERSON.

S-MOPs thus have to be inferred. This normally happens by inference from Action Units. The key determiners are Action Units that are are only used in a single S-MOP. So while an S-MOP like S-EXTORT can have an number of different

methods, an Action Unit such as $HIJACK virtually always occurs in service of S-EXTORT.

Thus when IPP reads a story such as S19, it can instantiate S-EXTORT as soon as it recognizes that "hijacked" is identifying the AU $HIJACK. Then that S-MOP can be used along with the S-MOP/AU Rule to explain the rest of the AUs in the story.

S19 - UPI, 30 June 80, Argentina

A well-dressed gunman Monday hijacked an Aerolineas Argentinas Boeing 737 jet on a domestic flight and demanded $100000 and enough fuel to fly to Mexico.

The man said to be in his twenties, released 45 passengers when the flight from the seaside community of Mar del Plata landed in Buenos Aires, but at least seven people plus crewmembers remained on the plane.

The plane, flight 601, was surrounded by air force police when it landed at Jorge Newbury metropolitan airport in Buenos Aires.

The S-MOP Instantiation Rule, then, is a simple one.

Whenever an Action Unit is instantiated that most commonly occurs as part of a single S-MOP, also instantiate that S-MOP.

Figure 21: S-MOP Instantiation Rule

This rule proves to be quite effective, as virtually all stories specify at least one appropriate Action Unit early on. If a widely used Action Unit is found first in a story, IPP simply waits until an S-MOP is instantiated, and then explains the AU in terms of it. One frequent example of this is the AU CAUSE-DEATH, which is a result of both S-ATTACK-PERSON (a direct attack), and S-DESTRUCTIVE-ATTACK, (where the deaths are a side-effect of the attack). So if a story starts "Three men were killed ...," IPP will wait and see

whether it continues "in a shooting ..." or "by a bomb explosion ..." before deciding what S-MOP CAUSE-DEATH is a result of.

The key to effective processing of a story by IPP, then, is to find as rapidly as possible an AU that specifies an S-MOP that can them be used to explain the rest of the actions that are described.

5. Language specific processing

Not all of the problems of text understanding can be handled by appealing to top-down context created from memory. There are some language-specific problems that must be dealt with. Three such issues are processing passive verb constructions, recognizing noun groups, and disambiguating words with multiple meanings. These three problems turn out to be the only language specific issues that IPP, must be concerned with.

IPP does not abandon its use of high-level memory structures in handling language specific problems. In fact, the unifying theme behind the solutions to the three problems I will discuss here is the way IPP continues to rely on high-level structures as much as possible. As with all of IPP's bottom-up processing, the goal is to allow memory-based processing to take over as soon as possible.

5.1. Passives

The passive construction in English is used to modify the role played by the syntactic subject of a transitive verb. The crucial nature of recognizing passives is illustrated by S20.

S20 - UPI, 15 July 80, El Salvador

Five guerrillas were killed Wednesday in a shootout at a Salvadoran grammar school where children were pinned to the floor by bullets blasting into their classroom.

If we used strictly semantic considerations to determine the role of the "five guerrillas" in the killing described in S20, we would undoubtedly conclude that they were the actors, due to our knowledge of the kinds of actions often performed by guerrillas. However, the passive nature of the verb "killed" overrides this possibility.

Recognizing most passive constructions is not difficult. IPP does so by saving auxiliary forms of the verb "to be" in a short-term memory buffer, and checking that buffer whenever a transitive Event Builder causes the instantiation of an Action Unit.

More difficult is the question of deciding what to do with the syntactic subject of an Event Builder that is passive. An Action Unit can have a number of different roles, and all we know initially from the fact that the verb is passive is that the normal slot filled by the subject is not correct.

One possibility would be to have specific rules associated with each Event Builder as to how role filling should be modified in the passive case. However, this would go against the goal of making processing as independent as possible of the specific language used.

Fortunately, there is another solution. It is illustrated by the set of stories below.

S21 - New York Times, 23 March 79, Guatemala

Manuel Colom Argueta, former mayor of Guatemala City and Guatemala's most popular leftist leader, was murdered by unidentified gunmen as he drove to his office in the Guatemalan capital this morning.

S22 - New York Times, 23 December 79, France

A Turkish official was slain by a machine-gun-wielding terrorist today on the Champs-Elysees.

S23 - UPI, 22 July 80, United States

A former Iranian diplomat who feared for his life because followers of Ayatollah Ruhollah Khomeini are "mad, really mad" was killed on his doorstep Tuesday by a man dressed as a postal worker.

These stories use different Event Builders to describe a person being killed. In each case the verb is passive. Despite the different verbs in the stories, the use of the subject in each is the same — it fills the VICTIM role of the CAUSE-DEATH Action Unit.

This example illustrates a general rule, confirmed by looking at a number of other stories with passive Event Builders. Knowledge about handling passives can effectively be stored with Action Units rather than individual EBs. Passive EBs that describe the same Action Unit use the syntactic subject in the same way. In the examples above, the use of the passive subjects of the EB's "murdered", "slain" and "killed" can be determined by refering to the AU CAUSE-DEATH, and not the individual words.

All the stories above are processed correctly by a rule that the syntactic subjects of Event Builders pointing to CAUSE-DEATH are used to fill the VICTIM slot of the AU. This allows an accurate passive rule to remain based on structures in memory.

Another situation worth noting involving passives concerns cases where the passive nature of the verb is not made explicit in the story. This case is also important, and requires the use of our world knowledge.

S24 is an example of such a situation.

S24 - UPI, 29 July 80, Unites States

> A Nationalist Chinese businessman killed by a bomb as he opened the front door of the home he was visiting apparently was the victim of a terrorist group seeking out Taiwanese officials, police said Tuesday.

The verb "killed" in S24 functions as if it were passive despite the lack of any form of the verb "to be". (The verb is technically part of a modifying clause.) Normally, the subject of killed indicates the actor of the CAUSE-DEATH Action Unit (as in "the terrorist killed ..."), but here it is the victim. As with the more conventional passives, we must recognize this case in order to correctly fill in roles. In fact, this turns out to be a very common construction in news stories, and is quite important for accurate understanding.

Recognition of this case depends upon access of high-level structures. Whenever IPP processes an Event Builder that has the potential of being made passive (e.g., normally takes an object), it checks to see whether the next word could initiate a noun phrase that might be the object of the Event Builder. If not, the EB is potentially passive, and IPP checks further to see whether this is actually the case.

IPP uses its knowledge of typical role fillers for the AU being instantiated to see whether the syntactic subject is a particularly good filler for the role normally filled by the subject of the EB (usually the

ACTOR role), thus overriding the hypothesis that the EB is passive. Otherwise, IPP assumes the construction to be passive and processing continues as if there had been a form of "to be" preceding the EB (e.g., "the businessman was killed").

It is the use of the Action Unit memory structure and its typical fillers that provides the crucial knowledge needed to distinguish the unspecified passive case ("the businessman killed ...") from a standard, active use of the same Event Builder ("the terrorist killed ...").

5.2. Recognizing noun groups

The processing of noun groups in English provides a difficult problem for any understanding system. The positioning of the head noun last, and the frequent use of noun-noun constructions contribute to the problem. One example, admittedly a bit extreme, that IPP encountered is shown in S25.

S25 - Boston Globe, 2 May 79

> Two men disguised as police officers yesterday kidnapped and shot prominent Cleveland Jewish community leader Julius Kravitz, chairman of the board of Pick 'n Pay Supermarkets and his wife Georgia in an abortive $1.5 million ransom plot, police said.

The problem in S25 is recognizing that "prominent Cleveland Jewish Community leader Julius Kravitz chairman of the board of Pick 'n Pay Supermarkets" describes a single individual, while "his wife Georgia" is someone else. Deciding the boundaries of a noun group, and how the parts go together can be quite a problem. Gershman [4] has considered this problem in some detail.

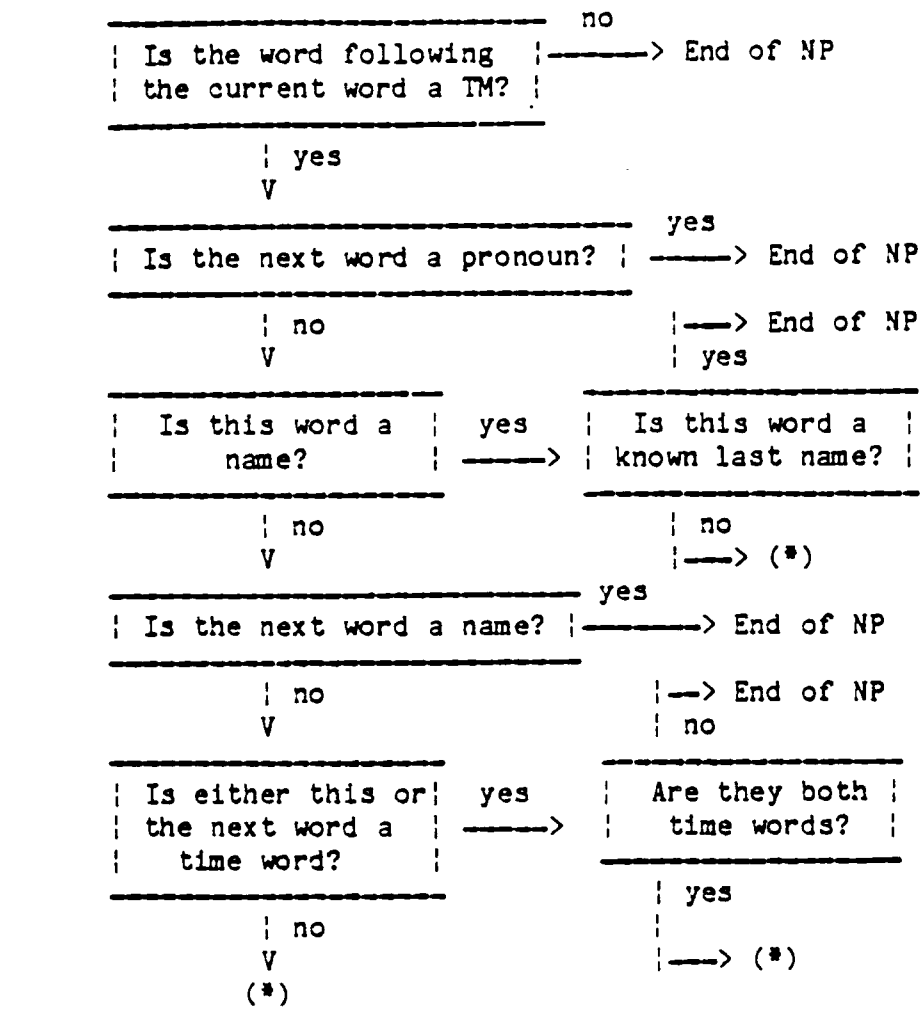
IPP concentrates on accurately determining the boundaries of noun groups. While this leaves some problems with creating the internal representation of

the Picture Producers described by noun groups, it does minimize confusion in processing the surrounding text, preventing parts of the noun group to be picked up by predictions as separate tokens.

The basic plan for IPP is to use simple semantic and syntactic rules to find the head noun in a noun group, using the save and skip strategy mentioned in Section 2.2.2, to save the words in the noun group until the head noun is found. Location of the head noun is performed using a series of simple heuristics. Their application is diagrammed in Figure 22.

While Figure 22 may appear complicated, in fact it is just the implementation of a series of simple heuristics. The first heuristic is, 1) a Token Maker followed by a non-TM is the head of a noun phrase. All the remaining heuristics apply to TM/TM (noun/noun) constructions. 2) Noun-pronoun constructions do not exist as part of the same simple noun group. 3) Last names end noun phrases. 4) Names appear only at the beginning of noun phrases (possibly in groups), hence a name following a non-name must be in a new noun group. 5) Time TM words do not combine with non-time TMs in the same noun phrase. These 5 simple heuristics suffice to delimit almost all simple noun groups.

At this point in the processing of the noun group, with the head noun located, it is possible to combine the previous words in the group (that were saved in a short-term memory buffer) with the head noun to form a representation of a conceptual Picture Producer. This part of the processing was not considered in great depth in the design of IPP, but does clearly involve the use of knowledge in memory about stereotypical Picture Producers.



Not end of NP; Save and skip word;

Go to next word

Figure 22: End of noun phrase determination

By using simple heuristics to locate the head noun, IPP is able to apply whatever information is available about the noun group at the most convenient time, once it knows what is being described.

The remaining major problem in delimiting noun groups is recognizing when two consecutive simple noun groups actually describe the same Picture Producer. This was the problem in handling the description of "Julius Kravitz" in S25.

Here again IPP uses simple syntactic and memory-based rules. It assumes any two noun groups not separated by an Event Builder or function word may be describing the same PP. Memory is then consulted to see if any of the properties of the two simple noun groups are in conflict. In the absence of any such conflict, the noun groups are assumed to be describing the same PP, and the information from them is combined.

With the Julius Kravitz example this rule indicates that "Julius Kravitz" and "chairman of the board" are potentially the same person. A memory check determines that a role such as "chairman" does not conflict with a name, and hence they are assumed to be describing the same person.

This contrasts with the processing of "and his wife Georgia". Again these could be describing the same person (as in "chairman of the board and member of the executive committee ..."), but this time the role and gender of the new noun phrase conflict with the previous and cannot be assumed to describe the same person.

The rules currently implemented in IPP for determining whether noun groups can be merged are quite simple. The focus has been on making the tests at the correct times. The eventual solution will require using knowledge about the features of people and objects. (See [2] and [14] for a discussion of the kinds of information available about people.)

As with the processing of passives, the solution to deciding whether noun groups can be merged is again one of using simple syntactic rules that allow semantic information to take priority when possible.

5.3. Resolution of ambiguity

One problem any natural language understander must address is the ambiguity of most words. With no restriction upon domain, nearly all common words have more than one meaning, and some, such as "got" and "ran" have literally scores. Some parsing systems have considered this problem to be the central one in understanding, and have had their design centered about a solution to it ([6, 15], for example).

IPP does not use a single solution to ambiguity. Instead several methods involving our knowledge of the world, simple syntactic rules, and the redundancy of language. These methods include 1) disambiguation by domain, 2) disambiguation by prediction, 3) syntactic disambiguation, and 4) skipping ambiguous words.

5.3.1. Disambiguation by domain

The primary method of disambiguating interesting words in IPP arose from the observation that very few significant words seem to be ambiguous within the terrorism domain. Once the domain of a story has been identified, it is often easy to pick single, appropriate definitions for words. For example, consider the word "sprayed" in S26.

> S26 - New Haven Journal Courier, 12 June 79, El Salvador
>
> Terrorists sprayed a car carrying a group of school teachers with automatic weapons fire Monday, killing three and wounding two, police said.

usually means something like "propel liquid in stream," but in this story, and in fact throughout the terrorism domain, its definition is a pointer to the Action Unit $SHOOT. Once a story has been identified as being in the terrorism domain, words like "sprayed" cease to be ambiguous. Other

words of this type include "divert" ($HIJACK), "fired" ($SHOOT), and "occupied" ($TAKEOVER).

For IPP, as currently implemented, this scheme of definition means many lexical items need have just one sense, since IPP deals with only a single domain. However, in the long run we would need lexical definitions sensitive to the domains that have been recognized. We might want words to have default definitions that are active unless certain domains have been found. I.e., "sprayed" would have the standard meaning, unless we were in the terrorism (or a similar) domain.

Domain dependent definitions will require that some interesting problems of domain delineation be solved. For example, for "sprayed" to mean "shot" in a domain implies that the domain covers acts of terrorist violence, but not vandalism or low-key protests, where "sprayed slogans on the walls" would become a reasonable possibility. Learning how such domains are delimited and recognized, as well as how the overlap among domains is handled is an important topic to be considered in future research.

Disambiguation by domain brings up the problem of recognizing an appropriate domain in the first place. This is not a problem that IPP deals with. Some ideas on domain selection can be found in [3, 9].

5.3.2. Predictive disambiguation

Disambiguation by prediction is another powerful tool. It has been used by many conceptual analysis systems. It can be applied in two basic ways. One is the actual prediction of certain words which might appear in the text. This is especially important for Function Words. As seen earlier, Action Units such as $HIJACK often fully specify how subsequent prepositions should

be processed. In general, most Function Words have no real meaning of their own, and can only be handled by proper predictions.

The second method of predictive disambiguation uses the rule that when we expect to see a word with a specified conceptual meaning, and a word is found with several meanings, one of the which matches the prediction, then it should be assumed that it is the intended meaning. This rule has been used in other predictive parsing systems such as ELI [12] and CA [1]. It can be used in conjunction with many of the memory-based predictions, such as the S-MOP/AU rule and the Associated AU Instantiation rule. This is an extremely powerful technique, and is in fact one of the main advantages of predictive processing.

Predictive disambiguation is necessary in S27.

S27 - Washington Star, 23 December 79, Rhodesia

Black gunmen attacked the suburban home of Patriotic Front guerrilla co-leader Robert Mugabe's sister on the first day of the Rhodesian truce yesterday and wounded two of her children.

The Event Builder "attacked" underspecifies the events it describes, even within the terrorism domain. It can refer to any of the Action Units $SHOOT, $EXPLODE-BOMB or $ASSAULT. However, in this case it is nicely disambiguated using the Associated AU Instantiation rule in conjunction with the disambiguation philosophy stated above. The Token Maker "gunmen" has $SHOOT as an associated Action Unit. From this the Associated AU Instantiation rule predicts that an Event Builder pointing to the same AU is likely to follow. Since one of the meanings of "attacked" does refer to $SHOOT, that meaning is selected as the correct one, thereby disambiguating the word.

It seems to be true in many cases that when an ambiguous word serves a

crucial role in a story it can be disambiguated by predictions such as those from the Associated AU Instantiation and S-MOP/AU rules.

5.3.3. Syntactic disambiguation

There is a moderately large group of lexical items that appear in the terrorism domain that are ambiguous in a rather simple fashion. They have multiple definitions, but each word sense is a different word type. For instance, Token Refiners such as "Russian" or "American" also have Token Maker senses. So we can have either "The Russian gunmen shot ..." or "The Russian shot ..." The similarity between the semantics of the senses in the example above is not coincidental. The senses of most of the words in this class have this property. One way to handle these words might be simply to have only one definition for each word and manipulate the meanings cleverly.

IPP adopts another solution, however. IPP definitions are always based around the processing to be done. In the example above, when "Russian" is acting as a TM we want to build a token. For the TR case, we should simply save and skip "Russian," since processing will be simpler if we wait for the head noun (see Section 5.2, above). Therefore, if it is not too hard to do so, we would like to use different definitions for the two cases.

Fortunately, in this case, as in many others, the disambiguation test is a simple syntactic test. We can determine with a high degree of certainty the sense of "Russian" to pick by simply looking at the next word. If the next word could be part of the same noun phrase, them we assume the TR sense, otherwise the TM sense must be the right one.

This kind of simple, one-word look ahead, disambiguation test is adequate for most of the syntactically ambiguous words used by IPP. In fact, the very

test described above handles the largest subclass of these words. By using tests of this kind, IPP avoids a great many other processing problems.

5.3.4. Skipping ambiguous words

The final technique IPP uses for disambiguation that I will describe here relies heavily on the fundamental redundancy of language. The technique is to try and skip words which are ambiguous, whenever possible. This counts on the fact that if a word is hopelessly ambiguous, there will be other words with the same meaning that get the point across.

An example of the practical nature of this strategy can be seen in S28.

S28 - UPI, 11 January 80, Corsica

Thirty nationalist gunmen who held 10 hostages in a hotel for two days surrendered peacefully early today and released their captives unharmed.

is a word that is extremely ambiguous. ("Hold" has 32 major definitions in Webster's Third New International Dictionary). In fact, it is so ambiguous that there will always be other clues as to what is going on. In this case, the definition of the word "hostage" itself indicates the relevance of the $TAKEOVER Action Unit.

It is feasible to save "held" in a short-term memory buffer rather than totally skipping it, so that words like "hostage" can check and be sure it was appropriate (and not a word like "shot"). However, an IPP style processor would never do extensive bottom-up processing on "held", due to the extreme ambiguity and corresponding lack of clear meaning.

A plausible explanation for the success of this strategy is that many of the more common words in English are used to convey shades of meaning, and the

less common, and usually less ambiguous words carry the bulk of the communicative load. At many levels of understanding, it is possible to pay little attention to the very ambiguous words.

6. A Detailed IPP Example

The following story, S29), will be be used to exemplify the parsing techniques described in this paper. In this paper I will only illustrate the parsing of the first sentence. Parsing repeated mentions of the same event, as in the second sentence, raises new problems, discussed in [8].

S29 - UPI, 23 July 80, Lebanon

Gunmen today shot and killed Riyad Taha, president of the Lebanese newspaper publishers' association and his driver in an ambush.

Taha, 54, a Shiite Moslem, was shot as he was going to his office in predominantly Moslem West Beirut.

The gunmen who opened fire at his car escaped.

IPP begins its processing of a story such as S29 with no specific expectations about what is likely to be described. Its first processing goal is to create a context allowing memory-based rules to be used. In the processing of this story, as with most news stories, it is possible to get an idea as to what the story is about almost immediately.

Processing begins in Figure 23. "Gunmen" provides the first source of top-down expectations. It is a Token Maker with several associated Action Units. Specifically, it is defined as the plural of "gunman", inheriting most of its properties from the definition of that word which has several associated AUs, as shown in Figure 24.

```
[PH: Initiation.  9-Sep-80  3:25PM]

 TOOLS TOPS-20 Command processor 4(560)-1
@IPP

IPP, ready  9-Sep-80 15:25:19

*(PARSE S29)

Story: S29  LEBANON

(GUNMEN TODAY SHOT AND KILLED RIYAD TAHA *COMMA*
PRESIDENT OF THE LEBANESE NEWSPAPER PUBLISHERS'
ASSOCIATION AND HIS DRIVER IN AN AMBUSH)

Processing:

GUNMEN          : Interesting token - GUNMEN

Predictions - LOOK-FOR-GUNMEN-ASSOCIATED-AU
              FIND-GUNMEN-ASSOC-SIBLING
```

Figure 23: TM with associated AUs

```
(D-SYN GUNMEN GUNMAN PLURAL]

(DEF-TM GUNMAN
        SUBTYPE        ACTOR
        ACTOR-TYPE     A-ROLE
        ASSOCIATED-AU ((SSHOOT          *TOKEN* = ACTOR)
                       ($SHOOT-ATTACK *TOKEN* = ACTOR)
                       ($KIDNAP        *TOKEN* = ACTOR)
                       ($HIJACK        *TOKEN* = ACTOR]
```

Figure 24: Definitions of "gunmen" and "gunman"

Reading "gunmen" causes IPP to activate requests that implement the Predicted AU Instantiation Rule and the Associated AU Instantiation Rule. In this case the first request, LOOK-FOR-GUNMEN-ASSOCIATED-AU, has a test examining incoming lexical items to see whether they have definitions pointing to any of the associated AU's for "gunmen" (i.e., $SHOOT, $SHOOT-ATTACK,

$KIDNAP or $HIJACK). If such an AU is found, it is instantiated with the token created for the conceptual Picture Producer "gunmen", used to fill the ACTOR role (as in the definition of "gunman" for each AU). FIND-GUNMEN-ASSOC-SIBLING implements the Associated AU Instantiation rule in the same fashion.

It does not take long for LOOK-FOR-GUNMEN-ASSOCIATED-AU to be satisfied by the text, as seen in Figure 25.

```
TODAY          : Normal token - TODAY
SHOT           : Word satisfies prediction

Prediction confirmed - LOOK-FOR-GUNMEN-ASSOCIATED-AU
```

Figure 25: Associated AU request satisfied

When "shot" is read, the Predicted AU Instantiation Rule is satisfied, since that word's definition includes a pointer to the script $SHOOT, one of the associated AUs of "gunman". This would happen even if IPP had several definitions for "shot", only one of which indicated an associated AU.

The confirmation of LOOK-FOR-GUNMEN-ASSOCIATED-AU causes $SHOOT to be instantiated, i.e., added to the description being built up for S29. In addition, the token created for the gunmen is identified as filling the ACTOR role of $SHOOT based on the definition of "gunmen". The identification of the first memory structure in the story results in further processing, shown in Figure 26.

Once $SHOOT has been instantiated, two separate types of processing take place — new requests are activated to create top-down context for

```
>>> Instantiated $SHOOT structure

Predictions - $SHOOT-ROLE-FINDER REDUNDANT-AU-WORDS
              $SHOOT-SYN-FINDER
```

Figure 26: $SHOOT instantiated

understanding later text, and a check is made to see if an appropriate S-MOP can be instantiated.

The first request shown activated in Figure 26, $SHOOT-ROLE-FINDER, implements the AU Role Filling Rule. As presented in Section 3.2.1, this rule uses the descriptions of typical role fillers included in the definition of $SHOOT to attempt to determine the roles of Picture Producers that follow in the text. (The definition of $SHOOT can be found in Figure 5.) In this case, IPP looks for PPs that could be construed as the victim, weapon, or wounded body part role of $SHOOT.

The second request activated by the instantiation of $SHOOT, REDUNDANT-AU-WORDS, looks for repeated mentions of that AU. This implements the idea that once an Action Unit has been mentioned in a text, it is likely to be mentioned again. Later mentions can provide additional details for our description of the event.

The final request activated at this point in the processing is $SHOOT-SYN-FINDER. It is based on the definition of the Event Builder "shot", and checks for the preposition "in" indicating the part of the victim's body that was shot.

Along with the activation of top-down predictions, the recognition of an

Action Unit causes a check to be made for potential S-MOP instantiation.

```
>>> Instantiated S-ATTACK-PERSON structure
Predictions - S-ATTACK-PERSON-RELATED-AUS
```

Figure 27: S-ATTACK-PERSON instantiated

As shown in Figure 27, the instantiation of $SHOOT does cause the selection of an S-MOP to be used in describing S29. $SHOOT, as shown in its definition, has a normal S-MOP, S-ATTACK-PERSON, that should be instantiated when $SHOOT is found, but not already explained by another S-MOP. This is feasible, since although the method of the S-ATTACK-PERSON S-MOP can be any of a number of AUs, the only S-MOP that $SHOOT is a method for is S-ATTACK-PERSON.

The final parsing action taken at this point is to activate a request implementing the S-MOP/AU rule. This request, S-ATTACK-PERSON-RELATED-AUS, checks incoming text for Event Builders that refer to any of S-ATTACK-PERSON's other methods, scenes or results.

At this point, considerable top-down context has been created by the instantiation of the memory structures S-ATTACK-PERSON and $SHOOT. There are three active semantic requests looking for $SHOOT's role fillers, additional mentions of $SHOOT, and other AUs related to S-ATTACK-PERSON.

IPP's processing of "shot" is complete at this point. What is worth noting is how quickly and how thoroughly it was possible to set up a context to be used as a source of top-down predictions. Almost all the rest of the

understanding of S29 is top-down to at least some degree.

The first example of the use of the context set up is shown in Figure 28, when "killed" is encountered.

```
AND          : Function word - conjunction - save and skip
KILLED       : Word satisfies prediction

Prediction confirmed - S-ATTACK-PERSON-RELATED-AUS

>>> Instantiated CAUSE-DEATH structure

Predictions - REDUNDANT-AU-WORDS CAUSE-DEATH-ROLE-FINDER
```

Figure 28: S-MOP/AU Rule satisfied

The S-MOP/AU Rule is used to identify "killed" as describing the CAUSE-DEATH Action Unit — a result of S-ATTACK-PERSON. This occurs in a similar fashion to the way $SHOOT was identified from "shot" using the Associated AU Instantiation Rule, by looking through its definitions for one that points to any of S-ATTACK-PERSON's related AUs. The action of this request causes CAUSE-DEATH to be instantiated and attached to the existing S-ATTACK-PERSON as a result. In addition, a new role filling request based on the AU Role Filling Rule is activated which attempts to fill the roles of CAUSE-DEATH.

Notice that the satisfaction of the S-MOP/AU prediction immediately specifies the causality between the shooting and the killing (from information built into S-ATTACK-PERSON), without any extensive inference process.

Another benefit of the attachment of this instance of CAUSE-DEATH to S-ATTACK-PERSON is that the role relation process, as described in Section 3.2.2, immediately determines that the actor of CAUSE-DEATH is the same as the

actor of $SHOOT, the gunmen. Again this is done simply, ignoring any syntactic considerations, relying on our knowledge of the stereotypical aspects of the situation.

```
RIYAD          : Token refiner - save and skip
TAHA           : Normal token - TAHA

Prediction confirmed - CAUSE-DEATH-ROLE-FINDER(VICTIM)
```

Figure 29: AU Role Filling Rule confirmed

In Figure 29, IPP begins the processing of a rather complex noun group. As soon as it has identified it as a name however, a token is built for the Picture Producer being described, and the role finding request that was activated during the instantiation of CAUSE-DEATH tests to see whether this token might fill any of the AU's roles. (The recognition of novel words that indicate names is not a problem dealt with by IPP. Work on this problem was done for the FRUMP program [3].)

The new token does satisfy the predicate describing the VICTIM role of CAUSE-DEATH, since it accepts any person or group not normally associated with terrorist acts. Nothing in particular is known about Riya Taha yet, except his name, but he assumed to be the victim. Using the relation among roles provided by S-ATTACK-PERSON, Taha is also identified as the victim of $SHOOT.

The remainder of the noun group, in Figure 30, provides additional information about Riya Taha.

"President" specifies a role that could feasibly be further describing

```
*COMMA*        : Skip
PRESIDENT      : Interesting token - PRESIDENT
   ----> Adding information to: RIYAD TAHA
OF             : Function word - preposition -
THE            : Function word - Token refiner - save and skip
LEBANESE       : Token refiner - save and skip
NEWSPAPER      : Interesting token - LEBANESE NEWSPAPER
   ----> Adding information to: PRESIDENT
```

Figure 30: Noun grouping

Taha, and so is assumed to do so. The phrase "of the Lebanese newspaper" is treated as a prepositional phrase modifying "president". IPP does not analyze such phrases deeply. It assumes that since the object of the preposition is an organization, it is specifying that the actor described by the token (Taha) has something to do with a newspaper in Lebanon.

```
PUBLISHERS'    : Token refiner - save and skip
ASSOCIATION    : Interesting token - PUBLISHERS' ASSOCIATION
   ----> Adding information to: LEBANESE NEWSPAPER
```

Figure 31: Further noun grouping

The remainder of the processing of the noun group describeing Taha, in Figure 31, specifies that the organization he is president of is not actually a newspaper, but a newspaper publishers' association, using a simple rule that two simple noun groups in succession ("newspaper" and "publisher's association") are describing the same token if they have no known properties that conflict. This additional information is added to the token for Taha.

The parsing of the rest of the sentence, in Figure 32, is rather superficial, since no predictions are satisfied and no words requiring extensive bottom-up processing are found, until the Event Builder "ambush" is

encountered.

```
AND          : Function word - conjunction - save and skip
HIS          : Token refiner - save and skip
DRIVER       : Normal token - DRIVER
IN           : Function word - preposition
AN           : Function word - Token refiner - save and skip
AMBUSH       : Word satisfies prediction

Prediction confirmed - S-ATTACK-PERSON-RELATED-AUS

>>> Instantiated $AMBUSH structure

Predictions - $AMBUSH-ROLE-FINDER REDUNDANT-AU-WORDS
```

Figure 32: End of first sentence

The first sentence of S29 ends with the mention of another Action Unit related to S-ATTACK-PERSON. Since "ambush" has as its definition a pointer to $AMBUSH, a method for S-ATTACK-PERSON, that AU is instantiated and attached to the S-MOP by the S-ATTACK-PERSON-RELATED-AUS request that implements the S-MOP/AU Rule.

The role filling for this AU must be done entirely on semantic grounds. Using the relation between $AMBUSH and S-ATTACK-PERSON, IPP can determine that the gunmen were the actors of the ambush, and Taha the person ambushed. Notice that had the story been, "Taha was shot and killed by gunmen in an ambush", IPP would fill the roles of $AMBUSH in a similar fashion, despite the different syntactic construction.

By the time the first sentence has been read, IPP has identified an instance of S-ATTACK-PERSON, with the method being $SHOOT and the result CAUSE-DEATH. The ACTOR of all these events is known to be a group of gunmen,

and a token has been created for the victim, the president of a publishers' association in Lebanon.

7. A Comparison to Other AI Parsers

In this paper I have discussed the methods that IPP uses to parse natural language. As a parser, it can be compared to other AI programs that transform natural language into internal representations.

The basic style of parsing performed by IPP has much in common with the conceptual analysis systems that originated with Riesbeck's ELI [11, 12] and include Birnbaum and Selfridge's CA [1] that were designed to create a Conceptual Dependency representation for sentences. Like these programs, IPP makes use of prediction, as implemented by a production-like system of test/action pairs (requests), to identify elements in text. Many of the refinements of IPP's noun grouping process are related to the work of Gershman [5].

Despite the underlying similarity between the conceptual analyzers and IPP, there is an important difference in the way predictive understanding is carried out. This involves the source of the predictions. In all the systems mentioned above, the primary sources of predictions are complex, procedurally oriented, word definitions. In contrast, IPP is able to create the majority of the predictions it needs from the memory structures identified to describe the story being read. This is done by rapidly identifying high-level memory structures, and using our knowledge of the stereotypical situations they describe to process later elements of the text.

The difference in the source of predictions contrasts even more strongly

with the conceptual analysis system of Small, the Word Expert Parser [15]. This program focuses on the problem of multiple senses of words, and uses very complex word definitions to achieve disambiguation and create a representation.

Another way that IPP contrasts with all these programs lies in the lack of total dependence on request-style processing in bottom-up situations. IPP relies on a procedural flow-of-control that uses declarative word definitions when there are no top-down predictions that explain a piece of text. It does not try and force request technology, which is well-suited for forward looking predictions, to handle non-predictive cases.

Another program that was influential in the creation of IPP was FRUMP [3]. This program was designed to be a robust skimmer of news stories. It has a novel design, consisting of two major modules, the Predictor, which creates top-down expectations based on the scripts it believes are relevant to the story, and the Substantiator, which attempts to confirm the expectations made by the Predictor. This design has enabled FRUMP to process hundreds of stories taken directly from the UPI news wire and produce representations based on "sketchy scripts".

The shared goal of robust understanding has lead both IPP and FRUMP to highly top-down, predictive designs. However, there are many important differences between the programs. The two most notable involve order of processing, and initial action representation.

IPP is a much more text driven program than FRUMP. It reads a story sequentially from beginning to end, creating top-down context, and using it to

process succeeding text. FRUMP instead moves its attention around in the text. It first looks for information to select a sketchy script that indicates what pieces of information should be searched for, and then goes back to try and find this information. In this phase of the processing, it looks for action words, and then goes back and forth looking at various words surrounding these words.

The initial representation of actions in FRUMP is also different from IPP. Rather than parse action words directly into their ultimate high-level representation, FRUMP first uses Conceptual Dependency [13] definitions, and then determines how the conceptualizations fit into the sketchy script being used to describe the story. While this allows additional generality in some cases, it also introduces a new level of processing that is often not necessary in IPP.

Parsers that first perform syntactic analysis on sentences have little in common with IPP. Augmented Transition Network parsers such as those described in [16, 18, 19, 7], as well as other parsers by Winograd [17] and Marcus [10], all treat syntactic analysis as a largely isolatable sub-part of the understanding process, that can be followed by semantic analysis. Since the syntactic structure of a sentence is of little use in itself, IPP does not explicitly determine it as a story is being read. It pays attention to syntactic details only when they affect the meaning being conveyed.

8. Conclusion

In this paper I have described in detail the parsing algorithm used by IPP, considering the novel elements of both its top-down, memory-based processing, and it bottom-up, heuristic techniques. The integration of these procedures has led to the creation of a very successful, robust understanding system.

IPP, writen in Yale/Rutgers/UCI LISP on a DECSystem 20/60, requires about 100K words of storage for the program and 3200+ dictionaries entries (including part of the generalization code). It was able to successfully process about 300 hundred stories, representing about 70 - 80% of the unedited stories it was tested on (all the relevant stories that appeared in local papers and the UPI newswire). It takes approximately 3 to 4 CPU seconds to process a typical story.

These statistics indicate that it is possible to develop a high-powered parser by making use of high-level memory structures to guide understanding.

REFERENCES

1. Birnbaum, L. and Selfridge, M. Problems in conceptual analysis of natural language. Tech. Rept. 168, Yale University Department of Computer Science, 1979.

2. Carbonell, J. G. Jr. Subjective understanding: Computer models of belief systems. Tech. Rept. 150, Yale University Department of Computer Science, 1979.

3. DeJong, G. F. Skimming stories in real time: An experiment in integrated understanding. Tech. Rept. 158, Yale University Department of Computer Science, 1979.

4. Gershman, A. V. Analyzing English noun groups for their conceptual content. Tech. Rept. 110, Yale University Department of Computer Science, 1977.

5. Gershman, A. V. Knowledge-based parsing. Tech. Rept. 156, Yale University Department of Computer Science, 1979.

6. Hayes, P. Some association-based techniques for disambiguation by machine. Tech. Rept. 25, Department of Computer Science, University of Rochester, Rochester, NY, 1977.

7. Kaplan, R. M. In process models for sentence analysis. In D. A. Norman and D. E. Rumelhart, Ed., Explorations in Cognition, W. H. Freeman and Company, San Francisco, CA, 1975.

8. Lebowitz, M. Generalization and memory in an integrated understanding system. Tech. Rept. 186, Yale University Department of Computer Science, 1980. PhD Thesis

9. Lehnert, W. G. Script selection. unpublished manuscript

10. Marcus, M.. A Theory of Syntactic Recognition for Natural Language. MIT Press, Cambridge, MA, 1980.

11. Riesbeck, C. K. Conceptual analysis. In R. C. Schank, Ed., Conceptual Information Processing, North Holland, Amsterdam, 1975.

12. Riesbeck, C. K. and Schank, R. C. Comprehension by computer: Expectation-based analysis of sentences in context. Tech. Rept. 78, Yale University Department of Computer Science, 1976.

13. Schank, R. C. "Conceptual Dependency: A theory of natural language understanding." Cognitive Psychology 3, 4 (1972), 532 - 631.

14. Schank, R. C. and Lebowitz, M. Does a hippie own a hairdrier? Tech. Rept. 144, Yale University Department of Computer Science, 1979.

15. Small, S. Word expert parsing. Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, La Jolla, California, 1979, pp. 9 - 14.

16. Thorne, J., Bratley, P. and Dewar, H. The syntactic analysis of English by machine. In D. Michie, Ed., Machine Intelligence 3, American Elsevier Publishing Company, New York, 1968.

17. Winograd, T.. Understanding Natural Language. Academic Press, New York, 1972.

18. Woods, W. A. "Transition network grammars for natural language analysis." Communications of the ACM 13 (1970), 591 - 606.

19. Woods, W. A. "Cascaded ATN grammars." American Journal of Computational Linguistics 6, 1 (1980).