LANGUAGE COMPREHENSION IN A PROBLEM SOLVER*

Douglas Wong**

Department of Computer Science
Brown University
Providence. Rhode Island 02912

Abstract

This paper describes BRUIN, a unified A] system that can perform both problem-solving and language comprehension tasks. Included in the system is a frame-based knowledge-representation language called FRAIL, a problem solving component called NASL (which is based on McDermott's problem-solving language of the same name), and a context-recognition component currently known as PRAGMATICS.

The intent of this paper is to give a flavor of how the context recognizer PRAGMATICS works and what it can do. Examples are drawn from the inventory-control, restaurant and blocks-world domains.

I Introduction

Over the years, it has been argued that there ought to be an AI system which can do both problem solving and language comprehension using the same database of knowledge [2,10, 11]. Such systems have not been built because researchers in these two areas have generally used rather different knowledge representations: the predicate calculus for problem solving, and some frame-like representation for language comprehension.

BRUIN is a unified AI system which can perform both problem-solving and language comprehension tasks. It is developed around Charniak's common knowledge representation FRAIL, a FRame-based AI Language. [3], The problem-solving component, called NASL, is based on McDermott's problem-solving language of the same name [7,8] The language comprehension component, which performs context recognition, is currently known as PRACMATICS. In addition, there is a natural language front-end which consists of a Marcus-type parser [6] and a semantic analyzer currently being developed by Graeme Hirst.

When a sentence is read by BRUIN, it is first parsed by the parser. The parse tree that results is then translated by the semantics module into a series of statements to be asserted into the FRAIL database. If the statement is a problem to be solved, then NASL will try to solve it. If the statement is an action which has been performed, then PRAGMATICS will try to recognize the context in which the action is being performed.

BRUIN is based on the following principles:

[Pi] The only knowledge which needs to be expressed in FRAIL is that associated with doing and choosing plans.

[P2] Given an arbitrary action, the context recognizer PRAGMATICS will recognize the action as either (a) a part (sub-act) of a plan; or (b) a way of accomplishing some higher goal.

For a rather simplified example, assume the following FRAIL definitions of the *build* and *magiciansaw-Victim* frames:[1]

```
; A step in the build plan is
; saw-step; the builder saws the lumber,
(frame:  build
    isa:  action
  slots:  (builder: must be a person)
          (lumber: must be a piece of board)
   plan:  (saw-step: builder saws the lumber)...)

/ A step in the magician-saw-victim plan is saw-step:
; the magician saws the victim,
(frame:  magieian-saw-vietim
    isa:  action
  slots:  (magician: must be a person)
          (victim: must be a person)
   plan:  (saw-step: magician saws the victim)...)
```

If the following statements are given to BRUIN,

Kim is building a house.
Kim is sawing a board.

then PRAGMATICS, noticing that Kim is sawing a board will recognize the board as the piece of lumber being sawed by the builder Kim in the *saw-step* of the *build* plan. On the other hand, if the statement were

Kim is sawing Lee.

then PRAGMATICS, noticing that Lee is a person, will recognize the sawing as the *saw-step* of a *magician-saw-victim* plan in which Kim is the *magician* and Lee, the *victim*

There is a rough correspondence here between the steps of "scripts" in language comprehension and the sub-goals of plans. In reality, because a plan is considered to be an integral part of a FRAIL frame, the knowledge available to BRUIN for performing inferences is more powerful than previous script formalisms.

This paper gives an overview of how the context recognizer PRAGMATICS works. PRAGMATICS can do all the examples presented in this paper if given the semantic representation for the English input. Except where noted, BRUIN's natural language front-end which translates the English into the semantic representation also works, but that process is beyond the scope of this paper. The domains which the examples are drawn from are (1) an inventory-control domain. (2) the restaurant-

"This research was supported in part by the Office of Naval Research under contract N00014-79-C-0592.

"Current Address: Department of Computer and Information Studies. Colgate University. Hamilton. New York 13346

*Throughout this paper, the FRAIL frames and the semantic representations will be given with an English-like gloss in italics. For a more formal treatment of these examples, the reader is referred to the author's thesis. [13].

dining domain and (3) the canonical blocks-world domain. All of these domains coexist in a single knowledge database and share some common knowledge. Although not described here, other domains which have been implemented include the operation, at the register transfer level, of a PDP8. and automatic programming in a very simple form based on Barstow's PECOS system. [1].

II Inventory Control

In this section, an edited version of a sample textbook problem whose solution requires the use of both PRAGMATICS and the problem solve NASL is presented. This problem is taken from the end of Chapter 7 of a textbook which deals with simple inventory control systems. [9].

(iI) A manufacturer used an injection molding machine in producing TV cabinets.
(i2) The standard time for production was 0.025 hours per TV cabinet.
(13) The injection molding machine operated for 80 hours per week
(i4) The manufacturer delivers 2000 cabinets per week.
(i5) The standard time for a setup change for the injection molding machine was 6.5 hours per lot.
(16) The setup change cost 3.50 dollars per hour.
(17) Storage of the TV cabinets cost 0.115 dollars per cabinet per week.
(i8) Calculate the economic lot size for inventory control of the TV cabinets.

This problem is interesting for several reasons. First. PRAGMATICS is needed to explicitly state that which is only implicit in the problem statement. In this case, PRAGMATICS must decide that the delivery of the 2000 cabinets, the setup change, and the production of the TV cabinets are steps in a manufacturing-business plan, a plan which itself is not mentioned. A side effect here is the realization that the TV cabinets being produced are the same ones which are delivered — a rather obvious, yet critical, inference. Second, it is in the nature of the economic lot size computation that NASL must choose between two similar equations when deciding how to compute the answer. Moreover, when an equation is decided upon. NASL must not use the same equation again when calculating the unknown variables of the chosen equation.

Before describing how PRAGMATICS understands this story, a sketchy introduction to inventory control and a quick solution to the problem will be presented.

A. The Economic Lot Stse

One of the decisions a manufacturer must make is how many items to produce in each of a series of production runs, which, taken together, will meet the demand for an item. This quantity is known as the economic lot size. The economic lot size is chosen so that it will minimize the total costs of meeting the demand for an item. There are a number of factors which may figure into the the determination of the economic lot size. Q — they are:

R The demand on the item,
S The cost of setting up a production run.
K The cost of keeping the item in inventory, and
P The rate of production for the item.

If the cost for setting up a production run is high, then decreasing the required number of production runs by increasing the lot size of a production run is desirable, since this would minimize the overall cost incurred. On the other hand, increasing the lot size would increase the average level of inventory in stock, thus increasing the overall cost.

There are two possible equations which may be used to calculate the economic lot size, *Q.* They are:

(EQ1)

and

$$Q = V^*Ir5" \quad (EQ2)$$

The difference in the two equations lies in an assumption about the production rate, *P.* In EQ1, *P* is considered to be infinite, or. in other words, the production of the good is instantaneous In EQ2, *P* is considered to be somewhat slower.

B. A Quick Solution

Given the above, albeit sketchy, knowledge of an elementary inventory control model, I will go through a quick solution to the problem. 1 chose EQ2 over EQ1 because information was given to me which would enable me to calculate the production rate. Once that decision is made, solving the rest of the problem is a matter of grinding through the calculations.

The production rate for *P* is calculated as follows:

p a *umr™p.rat£* ^0Q
rime *-to-produce-a- TV-cabxnet*

The demand. *P,* is the 2000 cabinets delivered, and the setup cost is

5 = *time-tosetup-lot x cost-pevhour*= $22.75.

Lastly, the carrying cost, *K,* is SO. 115, the cost of storage. Applying EQ2 to the values of *P, R. S,* and *K,* we end up with the economic lot size being

Q = V ^ ^ P " * - ^ r = 1452 units per Jot.

C. Pragmatic Deliberations

In this section, I will present the steps PRAGMATICS performs to do context recognition using the sample problem as an example. The semantic representation for the first sentence

(iI) A manufacturer used an injection molding machine in producing TV cabinets.

is

(manufacturer 1 produces 1 *the* TV-cabinets
on the injection-molding-machine)

Whenever an action is asserted into the database, PRAGMATICS will

[p1] Fetch from an auxiliary database a list of possible *context frames* and their *context links* to the just-asserted action.

The *context frame* represents a possible context for the asserted action. The *context link* is the statement which links the context frame with the action. For our example, one of the possible context frames is the *manufacturing-business* frame:

(frame: manufacturing-business
isa: business-operation

plan: (inventory-control-step:
manufacturer *performs* inventory-control on *the* product)
(setup-step: manufacturer sets up *the* machine-used *to produce* product)
(production-step: manufacturer produces *the* product *on the* machine-used)
(storage-step: manufacturer stores *the* product)
(delivery-step:
manufacturer delivers *the* product)...)

In this case, the context frame and corresponding link that PRAGMATICS will fetch is:

(manufacturer 1 operates a manufacturing-business)
{The production-step *of the* manufacturing-business *is* produces1)

based on the *production-step* of the *manufacturing>-business* plan.

The next step for PRAGMATICS is

[p2] For each of the context frames and links in the list, find ail the possible *context instances,* including a newly created context instance.

A *context instance* is a specific instance of a context frame. Because there aren't any *manufacturing-business* instances in the database, the only possible context instance is a created instance, *manufacturing-business 1* defined as:

(manufacturer 1 operates manufacturing-business 1)

with the link

(The production-step *of* manufacturing-business 1 *is* produces 1)

The next step for PRAGMATICS is:

[p3] Validate each context instance and link.

This means making sure that the context instance is consistent with what is already known about the frame. In this case, *manufacturing-business 1* has no *production-step* and the inclusion of *produces]* as the *production-step* of *manufacturing-business]* is consistent.

PRAGMATICS's next step is this:

[p4] Fill in the rest of the context.

This involves Ailing in slots of the context instance where necessary. For the manufacturing example:

{The machine-used *in* manufacturing-businessl *is the* injection-moiding-machine)
{The product *of* manufacturing-businessl *is* TV-cabinet)

At this point, there might be several valid context instances, so PRAGMATICS must

[p5] Choose and assert the best validated context instance, if any.

In the current version, the choice is made in a rather ad hoc fashion. That is, PRAGMATICS will choose the context instance with the fewest new instances and fewest statements necessary to establish the context. In the single choice given as the context for (il), there is one new instance, *business-manufacturxngi,* and four statements. Choices which are equal with respect to the above criteria will effectively eliminate each other: therefore it is possible that there may be no best choice. For the statement

(manufacturer 1 produces 1 *the* TV-cabinets *on the* injection-molding-machine)

there is only one choice so PRAGMATICS will assert:

(manufacturer 1 *operates* manufacturing-businessl)
(The production-step *of* manufacturing-businessl *is* produces 1)
(The machine-used *in* manufacturing-businessl is *the* injection-moiding-machine)
(The product *of* manufacturing-businessl *is* TV-cabinet).

It should be noted that once the action

(manufacturer 1 *operates* manufacturing-businessl)

is asserted, PRAGMATICS will go to work on it too This procedure terminates when PRAGMATICS stops creating new actions. There are no context frames for

(manufacturer 1 *operates* manufacturing-businessl).

Once again, the steps PRAGMATICS takes are these:

[pi] Fetch from an auxiliary database a list of possible *context frames* and their *context links* to the just-asserted action.
[p2] For each of the context frames and links in the list, find all the possible *context instances,* including a newly created context instance.
[p3] Validate each context instance and link.
I p4] Fill in the rest of the context.
[p5] Choose and assert the best validated context instance, if any.

What happens if a context instance already exists for an action as in the semantic interpretations for (i4). (i5), (i7), and (i6)? Each of these cases is similar so 1 will only present what happens for:

(i4) The manufacturer delivers 2000 cabinets per week.

whose semantic representation is, in part,

(manufacturerl delivers 1 TV-cabinets)

In step [pi], PRAGMATICS comes up with the context frame

(manufacturerl operates a manufacturing-business)

and the context link

(7Vie delivery-step *of the* manufacturing-business *is* delivers 1)

In step [p2], PRAGMATICS finds two context instances. One is *manufacturing-business],* created earlier, and the other is a newly created instance, *fnanufacturing-businessB,* I will refer to these as choice 1 and choice 2, respectively. Therefore, step [p2] results in:

Choice 1:
(The delivery-step *of* manufacturing-businessl *is* delivers 1)

Choice 2:
(manufacturerl operates manufacturing-business2)
(The delivery-step o/manufacturing-business2 iff delivers 1)

Steo fp3] of PRAGMATICS validates the choices and step [p4J adds

(77ie product o/manufacturing-business2 *is* TV-cabinet)

to choice 2. Note that choice 1 need not add any new context because it is already known that *manufacturing-business]* is manufacturing *TV-cabinets.*

The last step, [p5], must now deal with the two choices:

Choice 1:

(77M delivery-step 0/manufacturing-business 1
is delivers 1)

Choice 2:
(manufacturer 1 *operates* manufacturing-business2)
(The delivery-step 0/manufacturing-business2
is delivers 1)
(The product 0/manufacturing-business 2 is TV-cabinet)

The first choice is the overwhelming favorite because it doesn't create a new instance and it needs only one formula to establish the context.

III Restaurants

In the introduction, it was stated that one of the principles which BRUIN is based on is:

[P1] The only knowledge which needs to be expressed in FRAIL is that associated with doing and choosing plans.

A corollary of [P1] is:

[P1a] All other knowledge is deduced by inference where possible.

[P1] and [P1a] are designed to try to meet the objection that the knowledge expressed in frames and frame-like representations, such as scripts, is mostly adhoc. [5].

To illustrate this approach, two stories in a typical language comprehension domain, the restaurant-dining domain, are presented. The first story is:

(rl) Jack picked up a menu.
(r2) He ordered a hamburger.
(r3) He ate.

and the second story is an example of Wilensky's [12]:

(w 1) Will a was hungry.
(w2) She picked up a Michel in guide.
(w3) She got into her car.

They pose several interesting problems to a language comprehension system. The context for (rl) should include the fact that Jack is dining at a restaurant. Sentence (r2) tells us what food Jack will eat, so that what he is eating in (r3) is clear. In the second story, the same mechanism used for understanding (rl) can be applied to understand (w2). It should be noted that PRAGMATICS does not deal with the problem of pronoun reference. That is primarily the job of BRUIN's semantics analyzer which is not discussed here.

A. A Plan for Restaurant Dining

Before going into the stories themselves, I will develop a frame for restaurant dining which will be used to understand the first story. At the top level is the *restaurant-dining* frame itself.

(frame: restaurant-dining
isa: action
plan: (enter-step: *The* diner enters *the* dining-loc)
(decide-step:
The diner decides *on*
'(The food *to eat in this* restaurant-dining))
(order-step. *The* diner orders
(*That* decided *in the* decide-step))
(eat-step: *Ihe* diner eats
(That decided *in the* decide-step))
(take-check-step: *The* diner takes *the* check)
(pay-step: *The* diner pays *the* dining-loc
the amount *of the bill))*

Each step in the plan may be thought of as a goal to be achieved. The step which is of particular interest here is the *decide-step.* The *decide* frame represents a decision being made, which in *restour ant-dining* would be what food to eat.

(frame: decide
isa: action
slots: *(The* what *slot must be some* formula)
(The decided *slot may be* anything)...
plan: (*There is a*
(read-step: agent reads *the* reading material)
if
(The possibilities-list *in the* reading material
is what *is being decided))...)*

If there is some *reading-material* which lists the possible ways *what* can be decided upon, then read the *reading-material.* A partial read frame might simply be:

(frame: read
isa: action
plan: (pickup-step:
The agent picks-up *the* reading-material)...)

Examples of *reading-material* include objects such as books, menus and Michelin guides. A menu, for example, lists the possible foods one can eat in a restaurant-dining frame. Thus the menu frame is defined as:

(frame: menu
isa: reading-material
facts: (*The* possibilities-list *in a* menu *is*
'(The food *eaten in a* restaurant-dining *frame)))*

B. The Simple Story

As mentioned earlier, one of the difficulties posed by the first story is how to get the context of restaurant dining from the first sentence. Given the above frames, this can be easily done by PRAGMATICS. A rather straightforward translation of the first sentence:

(rl) Jack picked up a menu.

is:

(Jack picks-up 1 menul)

When *picks-upl* is asserted, PRAGMATICS uses the read frame to create the *read]* context:

(Jack readsl menul)
(The pickup-step 0/readsl *is* picks-upl)

One can read something to make a decision. Using the *read-step* of *decide,* and the definition of *menu,* PRAGMATICS hypothesizes that Jack is reading the menu to decide what to eat in some restaurant-dining frame Therefore the context for *readsl* is:

(Jack decides 1
'(The food *eaten in a* restaurant-dining *frame))*
(The read-step 0/decides 1 *is* readsl)

This decides; action fits really nicely as the *decide-step* of a created restaurant-dining action. As a result, PRAGMATICS will further assert:

Jack *goes* restaurant-dining 1 *at some* dining-loc)
The decide-step *of* restaurant-dining 1 *is* decides 1)

In addition, during validation of the *restaurant-dining 1* action; *re staur ant-dining 1* becomes bound to *restaurant-dining* in the *decides 1* action, hence *decides 1* becomes:

(Jack decides 1 *'(The* food *eaten in* restaurant-dining 1))

Note that PRAGMATICS is able to obtain the proper context for (rl) without having menu explicitly mentioned in the restaurant-dining frame at all

For next sentence:

(r2) He ordered a hamburger.

which translates to:

(Jack orders 1 hamburger 1)

PRAGMATICS recognises that *order1* is the *order-step* of *re st our ant -dining 1.* Furthermore, since the *order-step* of *restaurant-dining i*s defined as:

(The diner orders *'(That* decided in *the* decide-step))

validation of this context will find that the *diner* Jack has decided on a hamburger. Thus, the statement

(The item, decided *in* decides 1 is hamburger 1)

is also added to the context for *orders 1.*

The next sentence, "He ate", presents no new difficulty.

C. Poor Hungry Willa

A careful reading of the solution to the last story will reveal that the same mechanism used to understand why a menu is being picked up can be used to understand why a Michelin guide is being picked up. All we need to know is that the Michelin guide contains a list of possible places to dine out at.

IV Blocks World

The last domain to be presented in this chapter is a typical problem solving domain, the blocks world. The knowledge from this domain has been encoded in the system to solve the usual blocks world problems including the typical "hard problem", the destructive subgoals problem which may be stated as follows:

(b1) There is a green block on a table.
(b2) A red block is on the table.
(b3) The red block has a blue block on it.
(b4) Put the red block on the green block and the green block on the blue block.

The problem solver NASL can solve this problem, using the solution presented by McDermott in [6]. Furthermore, using the FRAIL frames developed for the solution to the above problem, PRAGMATICS can also find the context for the following simple story:[1]

There is a green block on a table.
There is a red block on the green block.
Jack put the red block on the table.

whose semantic interpretation is as follows:

; *There is a green block* on a *table*
(A block named green 1)
(The color o/greenl *is* green)
(A table named table 1)
(greenl is on table 1)

; There is a red block on the green block.
(A block named red1)
(The color *of* redl is red)
(redl *is* on greenl)

; *Jack put the red block on the table.*
(A person named Jackl)
(Jackl achieves 1 *the state* '(redl *is* on table 1))

A couple of relevant frames from the blocks world are:

(frame:, block
facts: (to-do *the task*
(agent achieves *the state* '(clear block))
do the task
'(agent *does* clear-plan *for the* block))
(to-do *the task*
(agent achieves *the state*
'(space-for *the* block *on* x))
do the task
'(agent achieves *the state* '(clear x))))

(frame: clear-plan
slots: (cluttered: *must be a* block)
plan: *(It is true that there is a*
(clear-top-step *for the* clutterer:
agent achieves *the state*
'(the clutterer *is* on a bwtable))

((the clutterer *is on the* cluttered)
(the bwtable *is a* table)
and
(cluttered *is on the* bwtable))))

For the statement:

(Jackl achievesl *the state* '(redl is on table 1))

PRAGMATICS will find the context frame and link

(Jackl *does a* clear-plan *for the* cluttered *block)*
(The clear-top-step *for the* clear-plan is achieve 1)

as well as a context frame and link associated with achieving a conjunctive subgoal. The latter option will eventually be eliminated when selecting the best choice in step [p5]. This is because the conjuctive subgoal option does not specify the other goal, whereas the *clear-plan* options is completely specified. For expediency, 1 will focus on the clear-plan option.

Since there are no instances of *clear-plan,* the only option generated by PRAGMATICS is the created instance, *clear-plan 1,* In order to verify that *clear-plan 1* is a possible context. PRAGMATICS must verify that:

((redl is *on the* cluttered *block)*
(table 1 is a table)
and
(cluttered *is on* table 1))

is retrievable from the database. Verifying this condition reveals that *cluttered* is the *greenl* block, so that the context for *achievesl* is:

(Jackl *does clear-plan 1 for* greenl)
(The clear-top-step *for* clear-planl *is* achievel)

PRAGMATICS now tries to find the context for the action *clear-planl.* Noting that doing *clear-plan* is a way to do

(agent achieves *the state* '(clear block))

and that there is no competition from other possible contexts, PRAGMATICS will assert

(Jackl *achieves2 the state* '(clear greenl))
(The sub-act *of* clear-planl *is* achieves2)

Therefore, PRAGMATICS has determined that Jack is moving the red block on the table is to clear the green block.

V In Closing

In this paper, I have presented an overview of how an AI system can perform one important aspect of language comprehension using the same knowledge a problem solver uses. It is a system which uses a common frame-based representation for representing

[1] This example does not yet work with the English input presented for reasons which are irrelevant to this paper.

knowledge associated with making and choosing plans. The problem-solving component, NASL, uses this knowledge to solve problems in a variety of domains. The context-recognition component PRAGMATICS uses this knowledge to recognize actions as a part of a greater plan, or as a means of attaining some higher-level goal.

We have been able to take advantage of the organizational structure of frames to express plans and the knowledge used to choose plans in a nice structured way. This has helped us get around some of the organizational problems associated with traditional, unstructured production-rules systems. By stressing the use of plan knowledge, the procedural portion of the frame has become less ad hoc than many of its predecessors. Knowledge which can be inferred need not be explicitly expressed. Probably, the best aspect of the system is that it works for the variety of domains which have been expressed for it; inventory control, restaurant dining, the blocks world, automatic programming, and computer architecture. In addition, all the frames described in this paper coexist in a single database.

Some of the problems we are currently experiencing have to do with incomplete knowledge or implementation specifics of BRUIN. To list a few:
(1) There is a problem with the blocks-world example which I suppressed in that section. In reality, the PRAGMATICS would go off on a tangent when the *achieved* task is asserted. This is because the concept of *prerequisite* in NASL is too poor to warrant its inclusion in PRAGMATICS at this time.
(2) Because each domain was developed somewhat independently, there are frames in the system which should be rewritten to make them more useful and general
(3) The system has not been seriously tested on a large complex domain. Even its knowledge about inventory control is rather superficial.

There are also some aspects of the general problem of context recognition which PRAGMATICS cannot handle They are:
(1) PRAGMATICS only uses actions to recognize context, not objects or states. As a result, it cannot handle:

Willa was hungry. There were cookies in her room.

(2) PRAGMATICS can only perform bottom-up recognition Given the story:

Jack picked up a rope. He had to hang up the wash.

PRAGMATICS will not recognize his hanging up the wash as the context for picking up the rope. Picking up a rope suggests too many different contexts (i.e. to tie something up, to jump rope, to make a leash, etc.) which can only be resolved by the subsequent sentences.
(3) PRAGMATICS is designed to perform *first impression* recognition only. It cannot perform, what 1 call *contemplative* recognition. An example given by Collins, et al in [4] is

> He plunked down IS at the window. She tried to give him S2.50 but he refused to take it. So when they got inside she bought him a large bag of popcorn.

This story is difficult because the initial context clues are weak (compared to clues like "menu"). Moreover, the initial context suggested by the first two sentences is wrong and must be revised when the last sentence is read.

In conclusion, this system does what many researchers have said ought to be done, that is unify the problem solver with the language comprehender and it does so with a common knowledge representation which can express knowledge in many domains.

Acknowledgements

I want to especially thank Eugene Charniak fur his insightful comments and suggestions throughout this research and the writing of this paper. 1 would also like to thank Graeme Hirst for his opinions on the ad hocness (or was it odd hackness?) of the translations I wanted out of his "one-shot" semantic translator, and for his thoughtful comments on an earlier draft of this paper.

References

[I] David R. Barstow, "Automatic Construction of Algorithms and Data Structures Using a Knowledge Base of Programming Rules," Memo AIM-308 Stanford Artificial Intelligence Laboratory (1977).

[2] Eugene Charniak, "A Framed PAINTING: The Representation of a Common Sense Knowledge Fragment," *Cognitive Science* 1(4) pp. 355-394 (1977).

[3] Eugene Charniak, "A Common Representation for Problem Solving and Language Comprehension Information." *Artificial Intelligence,* (forthcoming).

[A] Allan Collins, John S. Brown, and Kathy M. Larkin, "Inference in Text Understanding," in *Theoretical Issues in Reading Comprehension,* ed. R.J. Spiro. B C. Bruce, and W.F. Brewer, Lawrence Erlbaum Associates, Hillsdale, N.J. (forthcoming).

[5] Jerry Feldman. "Bad-Mouthing Frames," pp. 92-93 in *Proceedings of the Workshop on Theoretical Issues in Natural Language Processing,* ed R Schank and B.L. Nash-Webber, (1975)

[6] Mitchell P. Marcus, *A Theory of Syntactic Recognition for Natural Language,* MIT Press, Cambridge, Mass. (1980).

[7] Drew V McDermott, "Flexibility and Efficiency in a Computer Program for Designing Circuits," AI-TR-402, Artificial Intelligence Laboratory, Massachusetts Institute of Technology (June, 1977),

[8] Drew V. McDermott, "Planning and Acting," *Cognitive Science* 2(2) pp. 71-109 (April, 1978).

[9] Powell Niland, *Production Planning, Scheduling, and Inventory Control,* Macmillian Co., London (1970).

[10] Chuck Rieger, "The Commonsense Algorithm as a Basis for Computer Models of Human Memory, Inference, Belief and Contextual Language Comprehension," pp. 180-195 in *Proceedings of the Workshop on Theoretical Issues in Natural Language Processing,* ed. R. Schank and B.L. Nash-Webber. (1975).

[11] Roger C. Schank and Robert P. Abelson, *Scripts, Plans, Goals, and Understanding,* Lawrence Erlbaum Associates. Hillsdale, New Jersey (1977).

[12] Robert Wilensky, "Understanding Goal-Based Stories," Report 140 Yale Department of Computer Science (1978).

[13] Douglas Wong, "On the Unification of Language Comprehension With Problem Solving." Ph.D. thesis, Department of Computer Science, Brown University (forthcoming).