FOCUSING IN PLAN RECOGNITION

Norman F. Carver
Victor R. Lesser
Daniel L. McCue*

COINS Technical Report 84-23

*Digital Equipment Corporation
Continental Blvd. (MKO1-2/G09)
Merrimack, New Hampshire, 03054
(603) 884-5270

This research was sponsored, in part, by the External Research Program of Digital Equipment Corporation and by a grant from Rome Air Development Center.

FOCUSING IN PLAN RECOGNITION

ABSTRACT

A plan recognition architecture is presented which exploits application-specific heuristic knowledge to quickly focus the search to a small set of plausible plan interpretations from the very large set of possible interpretations. The heuristic knowledge is formalized for use in a truth maintenance system where interpretation assumptions and their heuristic justifications are recorded. By formalizing this knowledge, the system is able to reason about the assumptions behind the current state of the interpretations. This makes intelligent backtracking and error detection possible.

1.0 INTRODUCTION

An important issue for plan recognition in large search spaces is how to rapidly and accurately recognize the current plan based on the observation of a small number of plan steps. The need for this type of plan recognition has arisen in the POISE intelligent user interface system [CROF83, HUFF82, MCCU83]. Hierarchies of plans are used to specify typical combinations of user actions and the goals they accomplish. By recognizing a user's actions in the context of this model of possible actions, POISE is able to provide intelligent assistance to a user (e.g., agenda management, error detection and correction, and plan completion). Figure 1 describes a simple POISE plan that could be used as part of an intelligent assistant for an office automation system.

Plan recognition is a complex task since the recognizer may be forced to keep a very large number of plan interpretations under active consideration because of: 1) concurrency in user activities (i.e., loose constraints on the temporal ordering among plan steps); 2) sharing of plan steps among alternative plans; 3) the possibility that any partial plan might be continued by future user actions; 4) insufficient constraint information acquired from the observation of a small number of user actions to disambiguate among alternative interpretations. A key problem in the design of such

The descriptions of procedures/plans are specified in a formal language [BATE81] as depicted here:

```
!       The plan name and its parameters - the PROC clause

PROC    Complete-Purchase (Amount, Items, Vendor)

!       A textual description of the plan - the DESC clause

DESC    When the invoice is received, check with the requester to find out if the goods have
        been accepted and should be paid for or if they have been returned
        and so should be canceled

!       The steps involved and their relative ordering - the IS clause

IS      Receive-Invoice ´ Check-Goods ´ (Pay-For-Goods ! Cancel-Goods)

!       The constraints on the plan steps and objects - the COND clause

COND    (Check-Goods.Status = "Received") <=> WILL-EXIST Pay-For-Goods
        (Check-Goods.Status = "Returned") <=> WILL-EXIST Cancel-Goods

        Receive-Invoice.Items = Check-Goods.Items

        Receive-Invoice.Items = Pay-For-Goods.Items OR Cancel-Goods.Items

!       A definition of plan parameters - the WITH clause

WITH    Amount  = Receive-Invoice.Amount
        Items   = Receive-Invoice.Items
        Vendor  = Receive-Invoice.Vendor
```

This plan is one in a hierarchy of plans for purchasing in an office. The plan, Complete-Purchase, consists of three steps of which the final step is either Pay-For-Goods or Cancel-goods. The appropriate step is based on the acceptability of the goods received as specified with WILL-EXIST predicates in COND clause statements. Additional COND clause statements constrain the parameters of the sub-plans (e.g., the item being paid for must be the one referred to in the invoice). While this plan does not make use of it, the IS clause language provides for concurrency with the shuffle operator which leaves the relative ordering of the plan steps unspecified. The steps of concurrent top-level plans are implicitly shuffled.

Figure 1 - POISE description of Complete-Purchase plan

a plan recognizer is how to rapidly and accurately reduce the number of active plan interpretations in order for the system to be efficient and provide the most assistance to the user.

Our solution to this problem has been to develop a plan recognition architecture in which application-specific heuristic knowledge can supplement the constraint information in the plans. This heuristic knowledge is used by the focus-of-control strategy to quickly focus the search to a small set of plausible interpretations from the very large set of possible interpretations. The heuristics deal with the relative likelihood of alternative plans, the likelihood that plan steps are shared, and the likelihood of continuing an existing plan versus starting a new plan.

The heuristic knowledge has been formalized for use in a reason maintence system [DOYL79]. A formal system for representing heuristic assumptions has advantages for plan recognition systems. It becomes possible to reason about the assumptions behind the current state of the interpretation and why they were made. When new information is acquired which contradicts the current interpretation, the system can use this reasoning ability to recognize that the user has made an error or that the system has made an interpretation error. If an interpretation error has been made, an intelligent backtracking scheme can be used to decide what assumptions led to the invalid interpretation, how to undo these assumptions, and how to integrate the new information. This approach has the added benefit of allowing the system to explain to the user why it believes the user is carrying out a particular plan.

We believe that this approach not only applies to plan recognition, but to complex interpretation systems in general for it address issues which must be faced by any such system:

- The ability to exploit heuristic knowledge for control.
- The use of an intelligent focus-of-control strategy which can reason about context and the relationships between competing and cooperating interpretations when integrating new information.

- Adaptability to different applications and environments with changes to the heuristic control knowledge alone (i.e., without requiring changes to the underlying reasoning system).
- The use of an intelligent backtracking scheme.
- The capability of explaining its reasoning to users.

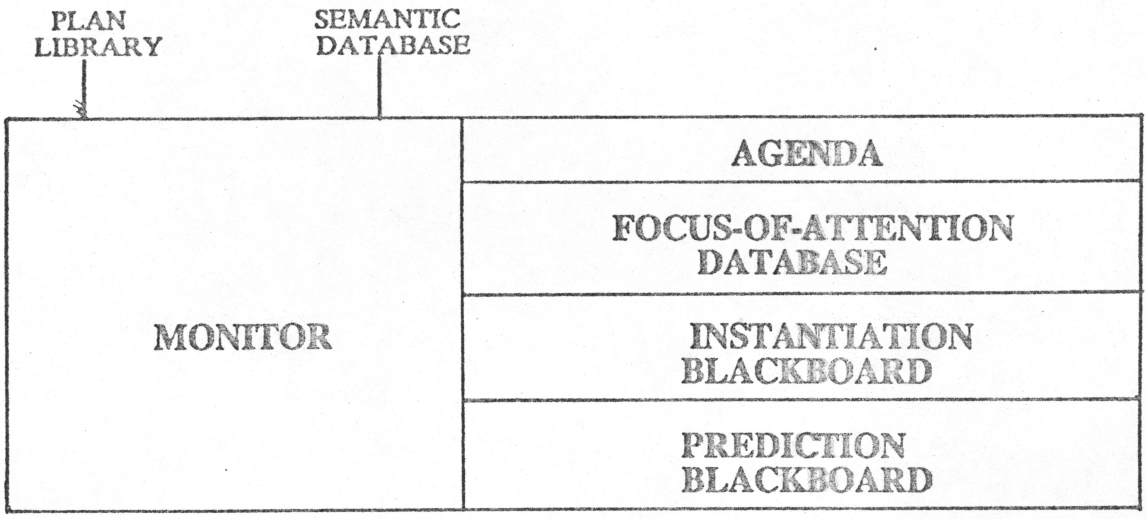
The remainder of the paper is organized into two sections. Section 2 discusses the plan recognition architecture and section 3 shows how the focus-of-attention strategy represents and uses the application-specific heuristic information to restrict the search.

2.0 PLAN RECOGNITION ARCHITECTURE

The plan recognition architecture consists of five components as depicted and explained in Figure 2. This architecture implements the basic interpretation machinery which checks syntactic and semantic validity of possible plan instantiations, i.e., plan step temporal orderings and attribute contraints. It also permits use of heuristic focus-of-control by providing a context mechanism. This allows interpretations which were previously considered unlikely to be recovered if new information invalidates the current likely interpretations.

When the monitor tracks a user action, it initiates the recognition process by attempting to integrate the user action into existing interpretations as follows:

- An instantiation of the primitive plan which represents the user action is placed on the instantiation blackboard.
- If a plan of this type was expected by any interpretation entries in the focus set, the monitor expands the predictions from the higher-level instantiation towards the instantiation of the action. This generates a hierarchy of instantiations on the prediction blackboard.
- When the predictions reach the level of the new instantiation, constraint values are checked. If the constraints are satisfied, the monitor integrates the user action instantiation into the higher-level plan instantiations. This is done by "abstracting" from the action instantiation up to the plan level of the predicting instantiation. The "abstraction" process creates plan instantiations and propagates constraint values. The new and the predicting



- MONITOR - the interpreter which tracks user actions and system events, instantiates plans corresponding to those actions and events, and propagates constraint information.[1]
- INSTANTIATION BLACKBOARD - a data structure where the monitor records potential interpretations of user activities as hierarchically related sets of partially instantiated plans
- PREDICTION BLACKBOARD - a data structure where the monitor "simulates" various predictions of future actions
- MONITOR AGENDA - a prioritized list of possible next steps for the monitor to take to expand interpretations. The monitor selects actions from this agenda to expand the interpretations that it is currently pursuing (i.e., its current "best" interpretations). The action of constructing an interpretation on the blackboard triggers the addition of new actions to the monitor agenda. If all agenda actions were executed, all possible (valid) interpretations would be generated.
- FOCUS-OF-ATTENTION DATABASE - a data structure where the monitor records its interpretations of user actions and the assumptions it made to arrive at those interpretations. The partial plan instantiations representing the current best interpretations are listed in the focus set.
- PLAN LIBRARY - contains data structures describing the plans. The monitor makes use of the temporal specifications of the sub-plans to predict the next steps in the plans and makes use of the constraint specifications when propagating attribute values between plan instantiations.
- SEMANTIC DATABASE - maintains a representation of the state of objects in the users world.

Figure 2 - Plan Recognition Architecture

[1] A full discussion of the constraint propagation mechanism is contained in [MCCU83].

instantiation structures are copied[2] and merged into a single structure which represents a new interpretation. The syntactically and semantically valid continuations of existing interpretations are posted to the focusing system as "can Continue" facts (see section 3).

- The monitor also checks to see if the user action could be the start of a new activity. It scans the plan library to determine whether or not the user action could syntactically start a new high-level plan.
- If a new plan could be started, the monitor "abstracts" from the action instantiation to higher-level plans, checking constraints and propagating constraint values until a top-level plan is reached. The plans which the user action can syntactically and semantically start are posted to the focusing system as "can STart" facts (see section 3).

To better understand how the basic plan recognition system and heuristic focusing interact, consider the following example. Assume that the interface is monitoring a purchase activity in progress. The state of the focus set and instantiation blackboard are as depicted in Figure 3. The plan, Complete-Purchase (see Figure 1), has as its first step, Receive-Invoice, whose only sub-step is a user action represented by the primitive plan, Receive-Information. The next step of Complete Purchase is Check-Goods whose only sub-step is the plan, Request-Information. Request-Information consists of the primitive plan, Send-Information, followed by the primitive plan, Receive-Information. The partial interpretation of a Complete-Purchase plan is on the instantiation blackboard. The focus set indicates that this Complete-Purchase plan instantiation is the most likely current interpretation of the actions seen so far. A Send-Information action has just occurred and its plan has been instantiated on the instantiation blackboard. This triggered the creation of an agenda entry, AE09, which, if executed, would generate the next level of abstraction for the given action (Send-Information.7). The sequence of actions leading up to this point has triggered the creation of other agenda entries some of which are pictured in Figure 3. Note that most agenda entries will not be executed if the focuser is correctly tracking user actions.

[2] This copy permits constraint values to be propagated and then easily retracted if the system decides to backtrack (i.e., revise previous decisions about which higher-level structure an instantiation is part of.)

Initial Focus Set

Current best: Complete-Purchase.6
Predictor: Complete-Purchase.6
Next step: Check-Goods
Agenda entry: AE06

Agenda

AE06 Predict Check-Goods from Complete-Purchase.6
...
AE09 Abstract Request-Information from Send-Information.7

Instantiation Blackboard **Prediction Blackboard**

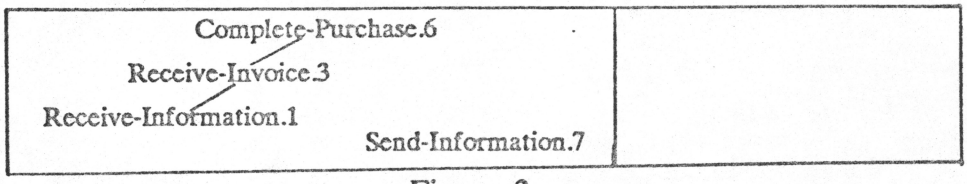


Figure 3

Agenda

...
AE09 Abstract Request-Information from Send-Information.7

Instantiation Blackboard **Prediction Blackboard**

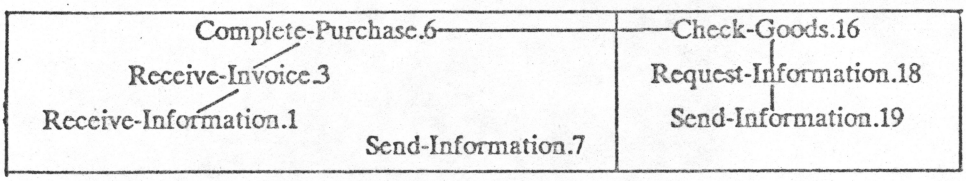


Figure 4

Final Focus Set

Current best: Complete-Purchase.25
Predictor: Request-Information.21
Next step: Receive-Information
Agenda entry: AE11

Agenda

...
AE11 Predict Receive-Information from Request-Information.21

Instantiation Blackboard **Prediction Blackboard**

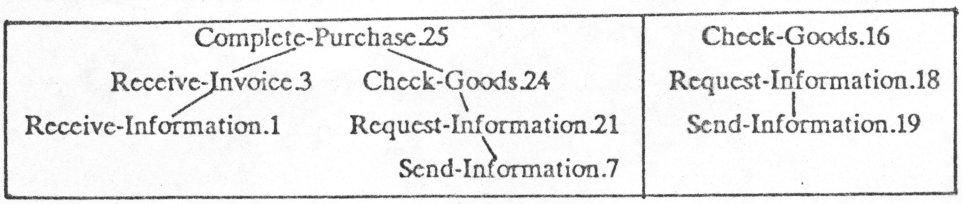


Figure 5

The monitor compares the instantiation, Send-Information.7, to the types of actions expected by the partial interpretations in the focus set. The monitor considers only those interpretations for the action which were expected by focus set interpretation entries. Through the use of pre-computed tables, the monitor detects a match between the plan type, Send-Information, and starting sub-step plan types of the plan, Check-Goods, which is expected next. The focus set entry also contains references to actions on the monitor's agenda which could be executed to generate the appropriate predictions for Check-Goods. The monitor instantiates those predictions on the prediction blackboard, removing the actions from the agenda. Constraint values are propagated down through the prediction instantiations shown in Figure 4. When the level of predictions meets the level of the instantiation representing the user action, constraint values are compared. In this case, Send-Information.19 has compatible parameter values with Send-Information.7. The action meets the expectation of the partial interpretation in focus. This example shows how focusing helps control the potentially explosive set of interpretations of primitive plans such as Send-Information by limiting the interpretations which are considered.

In Figure 5, the action, Send-Information.7 has been integrated into the partial interpretation. The abstraction process has created plan instaniations and propagated constraint values up to the level of the original prediction. Note that in Figure 4, the top instantiation in the interpretation structure is Complete-Purchase.6. The new instance of Complete-Purchase (Complete-Purchase.25) was copied from Complete-Purchase.6 when Check-Goods.24 was integrated. As previously noted, this copy operation permits the system to revise its decision that Check-Goods.24 should be part of Complete-Purchase.6. Focusing also eliminates the previous interpretation of Receive-Information.1, Complete-Purchase.6, from further consideration. While it might still be a valid interpretation (Send-Information.7 should really be interpreted differently or could be an error by the user), the focusing heuristics make it less

likely. If the focus set needs to be revised because later actions cannot be interpreted within the existing interpretation structure, the interface will reuse these "discarded" focus entries to pursue interpretations previously thought to be "unlikely."

3.0 FOCUSING

Focusing in POISE is a heuristic control mechanism which limits the interpretations of the user actions which the system considers to those interpretations deemed most likely. Heuristics about the likelihood of user actions are of three types:

1.) Application-specific - Derived from observations of the application environment. For example, "a single action is most likely to be taken to satisfy some part of a single goal." That is, sharing a single low-level plan among several high-level plans is unlikely. Also, "initiation of a new top-level plan is a less likely reason for executing some action than completing some plan already in progress." Since these heuristics are derived from a particular application (e.g., office systems), their relative importance (or usefulness) will depend upon the application. Other applications may have different characteristics.

2.) Plan-specific - The likelihood of specific actions being taken to satisfy particular goals (i.e., as steps of particular high-level plans) may be specified as part of the plans. For example, "action A is more likely to be part of plan X than plan Y."

3.) User Goals - The user may explicitly state goals (i.e., high-level plans with some or all of their parameters filled in) to be accomplished. Interpretations which match these explicit user goals may be considered extremely likely.

The focusing strategy uses these heuristics to make assumptions about the most "reasonable" interpretations to be expanded. The heuristics represent a form of nonmonotonic reasoning known as reasoning by default. A reason maintenance system is used for recording and maintaining interpretation assumptions along with their justifications. Assumptions are retracted with a dependency-directed backtracking scheme which identifies incorrect assumptions and proposes better assumptions. Assumptions can be retracted without having to undo unrelated interpretation work because the effects of changing a particular assumption can be inferred.

In order to formalize the heuristic knowledge, the facts and assumptions are represented as follows:

ST(k,X) — User action k can STart plan X.

C(k,Xu) — User action k can Continue plan instantiation Xu.

PE(k,Xu) — A Possible Explanation for user action k is as part of plan instantiation Xu.

SH(k,Xu,Yv) — User action k is SHared by plan instantiation Xu and Yv (i.e., the action fulfills parts of two goals).

GTR(k,Xu,Yv) — User action k has GreaTeR likelihood of fulfilling part of plan instantiation Xu than of plan instantiation Yv.

MLE(w,k,Xu) — A Most Likely Explanation for user action k in "state" w (i.e., after considering the sequence of actions, w) is that it is part of plan instantiation Xu. The interpretation entries in a focus set are the plan instantiations in the union of the MLEs in a particular state. A formal definition of Most Likely Explanation: IF PE(k,Xu) AND NOT THEREEXISTS Yv [PE(k,Yv) AND GTR(k,Yv,Xu)] THEN MLE(w,k,Xu).

where k stands for a user action, u, v, and w are sequences of user actions, X and Y are high-level plans, and plan instantiations are represented as Xu for the high-level plan underway (X) and the actions which partially fulfill it (u).

The heuristics are formalized as inference rules over these facts and assumptions. Three of the application-specific heuristics developed for the office application are:

H1 — A single user action is most likely to be taken to satisfy some part of a single goal so it is unlikely that the input action be shared by multiple high-level plans. This is implemented by stating that if no explicit assumption has been made that an input is shared by multiple interpretations, assume that it has not been shared:
IF PE(k,Xu) AND PE(k,Yv) AND CONSISTENT(NOT SH(k,Xu,Yv))[3]
THEN NOT SH(k,Xu,Yv).

H2 — Initiation of a new top-level plan is a less likely reason for executing an action than completing some plan already in progress:
IF C(k,Xu) AND PE(k,Xuk) AND ST(k,Y) AND PE(k,Yk) AND
NOT SH(k,Xuk,Yk) AND CONSISTENT(GTR(k,Xuk,Yk))
THEN GTR(k,Xuk,Yk).

---

[3] See [MCDE80] for an explanation of the nonmonotonic modal operator CONSISTENT.

H3 — Of two Possible Explanations for an input action, if one is assumed to be a Most Likely Explanation for a later user action then it has a GReaTer likelihood of being the Most Likely Explanation for this input:
IF PE(k,Xukj) AND PE(k,Yvk) AND NOT SH(k,Xukj,Yvk) AND
MLE(w,j,Xukj)[4] AND CONSISTENT(GTR(k,Xukj,Yvk))
THEN GTR(k,Xukj,Yvk)[5]

The focusing algorithm is as follows:

- Use the "can Continue" and "can STart" facts posted by the monitor during the basic interpretation process (see section 2) to post "Possible Explanation" assumptions for the user action.
- If there are no "Possible Explanations" for the action, either an interpretation error or a user error has occurred. Backtrack through the assumptions which have been made to find one which could result in an alternate interpretation (previously disregarded because it seemed unlikely) which would explain the current action. Retract this assumption, revise the interpretation of the previous actions and repeat the interpretation process for the current action.
- If an action has more than one "Possible Explanation," apply the heuristic rules to generate relative likelihood assumptions.
- Post the resulting "Most Likely Explanations" for the user action and propagate the results of this latest interpretation back to earlier interpretations.

To understand the focusing algorithm more fully, an example is given in Figure 6. The example considers only the syntactic interpretation of actions, ignoring constraint information. Since there is only one possible interpretation for the first user action, a, the focusing process is straightforward. The "Most Likely Explanation" for action a (see fact 3) is the only interpretation in focus. The second user action, b, can be interpreted as continuing the plan begun by action a or as the start of a new plan (facts 5 and 7). One of the heuristics formalized above, that continuing an interpretation is more likely than starting a new plan (H2), results in interpreting this action as a continuation of plan X rather than the start of plan Y (see facts 9 and 10). This interpretation of action b results in a new "Possible Explanation" for action a being posted (fact 11). This must be done

[4] Implicitly, w includes k and later user actions.
[5] This is a slightly simplified version of the actual rule.

Plan Grammar: X = a b d... Y = b c...

User Actions: a b c...

The sets of facts and assumptions for each successive "state" (string of user actions) are:

a	ab before backtracking	ab after backtracking	abc
1. ST(a,X)	4. C(b,Xa)	4. C(b,Xa)	15. C(c,Yb)
2. PE(a,Xa)	5. PE(b,Xab)	5. PE(b,Xab)	16. PE(c,Ybc)
3. MLE(a,a,Xa)	6. ST(b,Y)	6. ST(b,Y)	17. MLE(abc,c,Ybc)
	7. PE(b,Yb)	7. PE(b,Yb)	
Focus Set=	8. ~SH(b,Xab,Yb)	8. ~SH(b,Xab,Yb)	4. C(b,Xa)
{Xa}	{5,7,H1}	{5,7,H1}	5. PE(b,Xab)
	* 9. GTR(b,Xab,Yb)	10. MLE(ab,b,Xab)	6. ST(b,Y)
	{4,5,6,7,8,H2}	*14. MLE(ab,b,Yb)	7. PE(b,Yb)
	10. MLE(ab,b,Xab)		18. PE(b,Ybc)
			19. ~SH(b,Ybc,Xab)
	1. ST(a,X)	1. ST(a,X)	{5,18,H1}
	2. PE(a,Xa)	2. PE(a,Xa)	20. GTR(b,Ybc,Xab)
	11. PE(a,Xab)	11. PE(a,Xab)	{5,17,18,H3}
	12. GTR(a,Xab,Xa)	12. GTR(a,Xab,Xa)	21. GTR(b,Ybc,Yb)
	{2,11,10,H3}	{2,11,10,H3}	{7,17,18,H3}
	13. MLE(ab,a,Xab)	13. MLE(ab,a,Xab)	22. MLE(abc,b,Ybc)
	Focus Set=	Focus Set=	1. ST(a,X)
	{Xab}	{Xab,Yb}	2. PE(a,Xa)
			23. MLE(abc,a,Xa)
			Focus Set=
			{Xa,Ybc}

Facts and assumptions are grouped by the user action to which they refer. Only "in" [DOYL79] facts are represented and some obvious ~SH assumptions have been ignored. The justifications for assumptions made using the three heuristics formalized in the paper are given in braces ({ }) below the assumptions with the three heuristics denoted as H1-H3. Those assumptions changed during backtracking are denoted with *'s. C and ST facts are posted by the monitor as described in section 2. The remainder of the facts result from the application of various bookeeping rules.

Figure 6 - Focusing in a Sample Grammar

because, although b was assumed to continue the partial plan including a, it is possible that a is meant to be "Shared" by two X plans (i.e., Xab and Xab´ where b´ is another b action which has not yet ocurred).

When the third user action, c, is recognized, there is no way to interpret it within the existing interpretation structure (i.e., there is no fact of the form C(c,Zu) where Zu is some partial plan instantiation). This causes the system to backtrack, retract assumption 9 generated from the heuristic that continuing an interpretation is more likely than starting a new plan, and push forward the interpretation of plan Y started by action b (fact 14). User action c can now be interpreted as a continuation of plan Y (facts 15-17). Note that the interpretation of action c causes actions a and b to be reinterpreted (facts 22 and 23) because of heuristics H1 and H3.

4.0 STATUS AND FUTURE RESEARCH

The plan recognition system described is currently part of the POISE system. A special-purpose reason maintenance system was built. The additional knowledge provided by the focusing heuristics has been effective in the office automation application being studied. We are planning to explore its effectiveness in applications which are not as tightly constrained as the office.

We are currently developing a more intelligent backtracking scheme. The current scheme locates the most recent assumption whose retraction allows the current action to be explained. A more intelligent approach invloves reasoning about the "quality" of interpretation assumptions and the resulting interpretations in order to identify the "best" assumption to retract or to recognize that the user has made an error.

Another direction in which we are extending the current approach is the use of focusing heuristics which exploit the information about objects that is contained in the semantic database. These heuristics include knowledge about the use of objects

in plans, e.g., the likelihood that plans share objects or the likelihood of creating new objects.

5.0 ACKNOWLEDGEMENTS

Bruce Croft and Larry Lefkowitz have made contributions to the design and implementation of the plan recognition architecture as part of the POISE project.

6.0 REFERENCES

BATE81 P. Bates, J. Wileden, and V. Lesser.
Event Definition Language: An Aid to Monitoring and Debugging Complex Software Systems.
Technical Report 81-17, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 1981.

CROF82 W. B. Croft and L. Lefkowitz
An Office Procedure Formalism Used for an Intelligent Interface.
Technical Report 82-4, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 1982.

CROF83 W. B. Croft, L. Lefkowitz, V. Lesser, and K. Huff.
POISE: An Intelligent Interface for Profession-Based Systems.
Conference on Artificial Intelligence, Oakland, Michigan, 1983.

DOYL79 J. Doyle.
A Truth Maintenance System.
Artificial Intelligence, 12:231-272.

GISC81 J. Gischer.
Shuffle Languages, Petri Nets, and Context-Sensitive Grammars.
Communications of the ACM, 24(9):597-605.

HUFF82 K. Huff and V. Lesser.
Knowledge-Based Command Understanding: An Example for the Software Development Environment.
Technical Report 82-6, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 1982.

MCCU83 D. McCue and V. Lesser.
Focusing and Constraint Management in Intelligent Interface Design.
Technical Report 83-36, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, 1983.

MCDE80 D. McDermott and J. Doyle.
Non-monotonic Logic I.
Artificial Intelligence, 13:41-72.

VILA82 M. B. Vilain.
A System for Reasoning About Time.
Proceedings of the 1982 National Conference on Artificial Intelligence, pages 197-201, 1982.