ISSUES IN ADAPTIVE PLANNING

By

Richard Alterman

Cognitive Science Program

Institute of Cognitive Studies

University of California at Berkeley

July 1986

The production of the Berkeley Cognitive Science Report Series is supported by a grant in cognitive science to the University of California at Berkeley from the Alfred P. Sloan Foundation.

From alterman@yew.berkeley.edu Mon Jun 30 14:33:30 1986 Received: by cogsci.Berkeley.EDU (5.51/1.14) id AA09828; Mon, 30 Jun 86 14:32:31 PDT Received: by ucbvax.Berkeley.EDU (5.52/1.14) id AA11733; Mon, 30 Jun 86 13:52:03 PDT Received: by yew (1.1/5.17) id AA12576; Fri, 20 Jun 86 20:10:34 PDT Date: Fri, 20 Jun 86 20:10:34 PDT From: alterman@yew.berkeley.edu (Richard Alterman) Message-Id: <8606210310.AA12576@yew> To: chertok@cogsci.Berkeley.EDU Subject: Here's the on line version of the tech report. Status: R

(Hard copy will shortly follow)

Issues in Adaptive Planning*

Richard Alterman

Division of Computer Science
University of California, Berkeley
Berkeley, CA. 94720

This paper is about an approach to the flexible utilization of old plans called **adaptive planning.** An adaptive planner can take advantage of the details associated with specific plans, while still maintaining the flexibility of a planner that works from general plans. Key elements in the theory of adaptive planning are its treatment of background knowledge and the introduction of a notion of planning by situation matching. This paper will discuss a theory of adaptive planning as it applies to the domain of commonsense planning. The theory of adaptive planning has been embodied in a computer program called PLEXUS.

1. Introduction

Suppose a planner intends to transfer between airplanes, and the planner's normal plan is to:

- Exit first airplane via arrival gate.
- Determine departure gate.
- Walk to the departure gate.
- Board second airplane via departure gate.

---

* I would like to thank Bob Wilensky for initially getting me interested in the domain of commonsense planning and a number of provocative discussions. This work has benefitted from interactions with a number of other people. In particular, I would like to thank George Lakoff for some vigorous discussion that helped me to work on the relationship between general and specific knowledge, and James H. Martin and Michael Braverman for acting as sounding boards for many of the ideas contained in this paper. I would like to acknowledge interactions with a number of members of the Berkeley Cognitive Science Group; Jim Greeno, Peter Pirolli, John Searle, and Len Talmy, readily come to mind. I would also like to thank some past and present members of the BAIR group: David Chin, Paul Jacobs, Mark Luria, Jim Mayfield, Peter Norvig, and Lisa Rau. Michael Braverman deserves some special credit for editorial comments on the penultimate draft.

This research was sponsored in part by the Defense Advance Research Projects Agency (DOD), Arpa order No. 4031, Monitored by Naval Electronic System Command under Contract No. N00039-C-0235. This research was also supported by the National Science Foundation (IST-8514890).

In a small airport this old plan would work just fine. But in a larger airport, say Kennedy Airport where there is more than one terminal, if the arrival and departure gates were in different terminals, the plan would have to be modified (e.g. the planner would have to take a shuttle between terminals).

Adaptive planners are concerned with the problem of re-using old plans; in the case of the example above, an adaptive planner would refit the normal ‘airplane transfer’ plan to the novel circumstances at the Kennedy Airport. Previous research in this area has tended to be either much too restrictive in the utilization of old plans or has emphasized the problems of retrieval. Other research efforts were more focussed on positioning such techniques in the larger context of planning (or reasoning) than in the actual problem of utilization, or have been derailed into knowledge acquisition. But the central problem remains: It does little good to find old plans, or classify and prepare for the circumstances of utilization, if there are no detailed models of how the old plans can be re-used. Where adaptive techniques have been discussed, the role of knowledge has been largely abandoned in favor of a return to weak methods. The work described in this paper balances these views by emphasizing the problem of utilization. The refitting techniques that are described will be sufficiently robust to handle a wide range of relationships between old plans and new situations.

This paper will include some discussion of an adaptive planner called PLEXUS that models a planner refitting old plans to new situations in the commonsense planning domain. The organization of this paper is as follows: The remainder of the introduction includes a discussion of the major themes in adaptive planning and a review of the planning literature. The second section of the paper will discuss various issues concerning the representation of the background knowledge. This section begins with a discussion of the importance of background knowledge and ends with a description, and some motivation, of the four types of background knowledge used by PLEXUS. The third section of the paper gives an overview and some of the details of the PLEXUS computational model. This section includes traces of PLEXUS processing several examples. The paper concludes with a summary of the argument.

1.1. Themes

Consider a general plan for buying books at a book store in contrast to a specific plan I have for trading books at a used bookstore in Berkeley called Moe’s. The general plan for buying a book is that I go to the bookstore, browse, select a book, take the book to the cashier, exchange, and then leave. My specific plan for trading books at Moe’s add steps, such as bringing the old books to the book store, provides ordering information, such as the fact that I bring my old books to the bookstore before I try to trade books, and refines steps of the more general plan, such as ‘trading via a set policy’ instead of ‘exchanging’. The advantage of my more general book buying plan is that it applies in any number of situations, including the Moe’s situation. The disadvantage of using the general plan in a book trading situation is that the planner will need to rediscover many of the details associated with the more specific plan.

Much of the previous work in planning has tended to emphasize general plans over specific plans. Adaptive planning uses a mixture of specific and general plans, but **foregrounds specific plans.** Since much of commonsense planning occurs in typical situations, one advantage of foregrounding specific plans is that they are tailor-made for the planner’s normal circumstances. Moreover, even in the cases where the more specific plan must be re-fit, many times the cost of such changes will be much less than the cost of dealing with the subgoal and subplan interactions inherent in a process that works by instantiating more general plans.

Adaptive planning makes the **background knowledge** associated with the old plan explicit. Previous approaches to re-using old plans have dealt with an old plan in relative isolation and, therefore, the refitting task has been considerably more complicated. By making the content and organization of the background knowledge explicit, it becomes possible to re-use an old plan in a wide variety of situations.

PLEXUS' knowledge-base takes the form of a network. The background knowledge associated with an old plan is determined by the old plan's position in the network. The network includes taxonomic, partonomic, causal, and role knowledge. PLEXUS uses the taxonomic structure not only for the purposes of property inheritance, but also as a basis for performing two important reasoning functions: abstraction and specialization. The partonomic structure is used to aid in determining the piece of network which needs to be refitted in a given situation. The causal knowledge serves several functions: It identifies the abstraction which preserves the purpose of the old step, supplies appropriateness conditions, and provides dependency links between step. The role relations are used by PLEXUS for cross indexing purposes.

An important piece of the background knowledge are the more general plans. It is by having access to these more general plans that an adaptive planner gains flexibility. When a step in the old plan fails it is not totally abandoned, but rather it is treated as **representative of the category of action** which is to be accomplished. In the case of transferring between airplanes at Kennedy airport, the failing step, walking between arrival and departure gates, is representative of the correct category of action, 'travelling'. An adaptive planner uses the category knowledge, as represented by the failing step, to determine its eventual course of action, 'taking a shuttle'. PLEXUS achieves this by exploiting the background knowledge about categories in two ways. It determines what is right about the old plan step by a process of **abstraction.** During abstraction, PLEXUS removes the details from the old plan step that made it inappropriate in the current context. During **specialization,** it determines a more appropriate plan from which to interpret its course of action. In the case of the airplane transfer example, by a process of abstraction it determines that what is in common between the old plan step and the new situation is 'travelling', and by a process of specialization it identifies the alternate plan to take a shuttle.

Making explicit the background planning knowledge allows for a different kind of planner. Rather than planning by problem solving, it becomes possible to plan by **situation matching.** Rather than treating the old plan as a partial solution which is modified using weak methods, the old plan is used as a starting point from which the old and new situations are matched, and in the course of the matching a new plan is produced. The **interaction** of planning knowledge and the current situation determine a plan which fits the current context and realizes the goal. The interaction works in both directions. In the direction of planning knowledge to situation, the old plan serves as a basis for interpreting the actions of other agents and the various objects in the new situation. Moreover, it provides the planner with a course of action. In the direction of situation to planning knowledge, it is the situation which provides selection cues that aid the planner in determining an alternate course of action when complications arise.

Consider planning by situation matching in contrast to planning by problem solving. Planning by problem-solving is a constructive process: The connections between pieces of a plan are forged using various problem-solving techniques. Planning by situation matching takes an old plan and uses its position in a network of plans as a starting point for finding a match between the requirements of the new plan situation and some portion of the network of plans. Where planning by problem-solving is a constructive process, planning by situation matching works by matching what is known to what exists. Where planning by problem-solving achieves flexibility by emphasizing techniques, planning by situation matching achieves it by emphasizing the structure of knowledge.

1.2. Planning Literature

The earliest work in planning in Artificial Intelligence grew out of the work on weak method problem solving (weak methods: Newell, 1974) [. new74 .]. Given an initial state and a goal state, a means-ends problem-solver (GPS: Newell & Simon, 1972) [. new72 .] selects an operator to reduce the difference between the current state and the goal. In the case of GPS (Ernst and Newell) there existed a table that associated differences with the types of operators that reduced them. When a difference occurred GPS used the table to find an operator to reduce that kind of difference. STRIPS (Fikes & Nilsson, 1971) [. fik71.] formalized means-ends analysis in terms of predicate calculus.

In contrast to these approaches, adaptive planning reduces differences when a step in a plan fails, but not differences between operators, but instead differences between the old plan and the current situation. A second difference is the manner in which an adaptive planner reduces the differences. Where GPS uses a difference table, and STRIPS backchaining using preconditions and postconditions, Adaptive Planning reduces differences using the categorization hierarchy. By abstraction, an adaptive planner finds out what is in common between the old plan step and the new situation. By specialization, it finds an alternate detailed step from which to work. A third difference is that means-ends analysis is intended to work in knowledge poor contexts, where adaptive planning is intended to work in knowledge rich situations.

One of the problems of weak methods was that for planning problems with large search spaces the planner could rapidly run into the problem of combinatorial explosion. Some of the early work at SRI addressed this issue: first by introducing larger operators and then by introducing hierarchical planners

The first SRI approach, MACROPS (Fikes & Nilsson, 1972) [. fik72 .] attempted to deal with the problem of combinatorial explosion by working with larger chunks of knowledge. The application domain of MACROPS was robot problem solving. Solutions to old problems were generalized by substituting variables for the arguments of some of the operators. During planning, if some portion of an old plan, in its generalized form, achieved some goal or subgoal of the new plan it was re-used. The major limitation on MACROPS was that it failed to deal with the problem of flexibility. The goals and situation of the old plan, except for where variables were substituted for constants, had to identically match the new situation. For most real world problems it is rarely the case that the old and new plan situations are identical.

The second approach that came out of the early work at SRI was concerned with hierarchical planners (Sacerdoti: 1974,1977) [. sac74,sac77 .]. The key insight of hierarchical planners was to solve problems in a more abstract space and then gradually instantiate more details. For example, Sacerdoti developed two such approaches: ABSTRIPS (1974) [. sac74 .], which abstracted over preconditions, and NOAH (1977) [. sac77 .], which abstracted over operators. As hierarchical planners of this sort gradually instantiate a plan they must deal with the interactions of both goals and subplans. The interaction of parts can result in either reordering steps or backing up to resolve the problem at a higher abstraction level.

Like MACROPS, adaptive planning works by re-using old plans. Unlike MACROPS, it addresses the problem of flexibility. Like ABSTRIPS, adaptive planning has access to abstraction hierarchies. Unlike ABSTRIPS, it works from complete specific plans - foregrounding them while backgrounding the more abstract ones.

The other important themes in the planning literature has to do with cognitive theories of planning. Adaptive planning responds to several key ideas in this literature.

As opposed to the relatively straightforward conjunctive goal problems dealt with by robot planners, commonsense planners construct plans for more complicated scenarios. McDermott (1978) [. mcd78 .] characterized these more complicated scenarios as tasks: ‘‘I introduce the notion of *task* - any describable action to which the interpreter is committed’’ (p 73-74). Like ABSTRIPS, McDermott’s planner, NASL, worked by gradually instantiating more general plans. The emphasis of McDermott’s work was on techniques for interleaving planning and acting; so where ABSTRIPS compiled plans, NASL interpreted them. Like NASL, adaptive planning interleaves planning and acting, but not only because it is characteristic of commonsense planning, but also because it is the interaction which drives much of the access to knowledge.

Wilensky’s approach to planning grew out of the work at the Yale AI Laboratory on plan-based approaches to text understanding (Schank & Abelson, 1977; Wilensky 1977) [. sch77,wil77 .]. Much of the planning that characters do in stories is common-sense planning. Wilensky claimed that an important characteristic of common-sense planning is that it is knowledge intensive. He argued (1983,1984) [. wil81,wil83 .] that the planning and understanding systems shared planning knowledge - not only knowledge about plans, but also knowledge about how to plan. The emphasis of his work was on knowledge concerning goal interactions. Eventually he outlined (1983) [. wil83 .] a meta-planner which controlled the planning process via a set of higher level goals and plans called meta-goals and meta-plans.

Like Wilensky’s notion of commonsense planner, an adaptive planner is a knowledge intensive planner. Wilensky posited a much more elaborate goal structure than is used by PLEXUS. For example, he emphasizes the role of goals in plan selection. PLEXUS also uses goal information, but integrates it into a more general indexing scheme (see below for comments on Kolodner’s work). Unlike Wilensky’s proposal, which does not propose solutions for the problem of refitting, flexibility is a central issue for PLEXUS.

Carbonell’s work was the first attempt to deal with the problem of using old specific plans in commonsense situations. The techniques he proposes are much more extensive then those used in Macrops. He argued that in many cases more general examples were not available, and consequently specific plans could be used to reason by analogy. He has suggested two approaches to employing analogies. His first approach [. car81,car83a .] applied a means-ends analysis to an old problem, gradually transforming it into a solution for a new problem. His second approach [. car83b .] called derivational analogy, attempts to recreate the decision making process of relevant past problem solving situations, and apply that decision making process to the new problem situation.

An important difference between Carbonell’s work and the work on adaptive planning, concerns the proposed roles of specific plans in the planning process. For Carbonell, a specific plan is used only when more general plans are not available. In adaptive planning, specific plans are foregrounded and the general plans are in the background and are used to accommodate differences between old plans and new situations.

A second difference concerns the relation between derivational histories and background knowledge. A derivational history might include some of the background knowledge, but the two are distinguishable. Consider the following two cases. If the old plan was learned by rote there would not be any derivational history, but there would be background knowledge. Even in the cases where there exists a derivational history it is not at all clear that all of the relevant background knowledge was brought to bear in the previous problem solving situation, in which case the planner does not have access to all the potentially relevant knowledge. Moreover, the derivational history is missing an important part of the background knowledge: the abstraction hierarchy. Whereas adaptive planning can exploit the abstraction hierarchy by treating failing steps as representative of the category of action it wishes to perform, both of Carbonell's approaches are forced to use more traditional methods, such as the ones used by GPS and STRIPS, to refit failing steps. Overall the difference can be described as follows: A derivational history represents the decision making process that devised the old plan. The background knowledge represents the relationships between the old plan and the other pieces of knowledge that are related to it.

The third set of differences is largely a terminological dispute. Carbonell refers to his work as analogical reasoning. His position appears to be that if things aren't nearly identical, it must be a case of analogy. In this work I have opted for the tripartite terminology suggested by Gentner (1983) [. gen83 .]: nearly identical, similar, and analogy. For Genter a relationship is analogical if the predicates mapped over to the new situation are **only** of a higher order. Given such a classification, I take adaptive planning to be working mostly from similar examples. For example, I take the problem of transferring between planes to be a case of similarity - not analogy.

Finally adaptive planning is in the spirit of recent work in artificial intelligence on modelling human memory (Kolodner, 1983ab; Schank, 1980) [. kol83a,kol83b,sch80 .]. Kolodner's early work (CYRUS) was concerned with the problem of reconstructive memory. Her later work with Simpson [. kol84,kol85 .] has begun to deal with the problem of retrieval for case based reasoning. In both cases Kolodner uses indexing techniques for selecting specializations of categories. Other researchers have also addressed the problem of initial retrieval. Hammond's work refines the goal inter-relationships suggested by Wilensky and uses them for initial retrieval in case-based reasoning. The recent work of Hendler (1985) [. hen85 .]. suggests some spreading activation techniques.

In general, where these researchers have emphasized the problem of initial retrieval, the work on adaptive planning balances that view by investigating issues concerning flexibility and usage. There is some overlap between the specialization techniques that PLEXUS uses and the indexing techniques used in CYRUS. PLEXUS is like CYRUS in that it uses indexing techniques for specialization but differs in that the techniques it uses for subcategory selection is geared towards planning and emphasizes the interaction between the planner's knowledge and its current circumstances. One advantage of using general indexing schemes, over purely goal-based ones, is that it allows the planner to differentiate between two plans that are associated with the same goal. For example, the plan to 'buy a ticket from a machine' and the plan to 'buy a ticket from a teller' cannot be differentiated by goals but can be differentiated by other kinds of cues from the situation. Moreover, there are many situations where a step in a plan needs to be refit but no new goals have been introduced.

2. Representation of the Background Knowledge

Suppose a planner is about to ride the NYC subway for the first time, and uses its experiences on BART (Bay Area Rapid Transit) to guide its planning activity. Consider the steps involved in riding BART (see figure 1). At the BART station I buy a ticket from a machine. Next, I feed the ticket into a second machine which opens a gate to let me into the terminal and then returns my ticket. Next I take the train. At the exit station I feed my ticket to another machine that keeps the ticket and then opens a gate to allow me to leave the station. Compare that to the steps involved in riding the NYC subway: buy a token from a teller, put the token into a turnstile and then enter, ride the train, and exit by pushing through the exit turnstile. Given the BART Plan as depicted in Figure 1, there appears to be little in common between the two procedures.

- In the BART case a ticket is bought from a machine, in the NYC subway case there is no ticket machine and instead a token is bought from a teller.
- In the BART case the ticket is returned after entering the station, in the NYC subway case the token is not returned after entry.
- In the BART case the ticket is needed to exit, in the NYC subway case the token is not needed to exit.

The problem with the BART Plan, as shown in figure 1, is that it represents this plan in isolation. In isolation the old plan does not provide enough information to refit it to the NYC subway situation. There is a great deal of background knowledge associated with the BART Plan that is not explicitly represented in the figure 1, but is needed in order to re-use the old plan. Without the background knowledge the BART Plan is practically useless in the construction of a plan for riding the NYC subway.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (2.033i, 6.769i) "bart-proc"
circle  at (0.904i, 5.640i) "buy-bart-ticket"
circle  at (2.033i, 5.640i) "enter-bart"
circle  at (3.162i, 5.640i) "ride-train"
circle  at (4.066i, 5.640i) "exit-bart"
arrow from (1.426i, 6.169i) to (1.101i, 5.838i)
"step1" at (1.489i, 6.233i)
line from (1.835i, 6.572i) to (1.553i, 6.296i)
arrow from (2.033i, 6.176i) to (2.033i, 5.915i)
"step2" at (2.033i, 6.240i)
line from (2.033i, 6.494i) to (2.033i, 6.304i)
arrow from (2.633i, 6.169i) to (2.965i, 5.838i)
"step3" at (2.569i, 6.233i)
line from (2.231i, 6.572i) to (2.506i, 6.296i)
arrow from (3.120i, 6.162i) to (3.826i, 5.774i)
"step4" at (3.014i, 6.226i)
line from (2.273i, 6.635i) to (2.908i, 6.289i)
```

Figure 1: The BART Plan in isolation.

A key idea to understanding the adaptive planning approach to flexible knowing is:

- The content and structure of the background knowledge associated with the old plan must be explicit and available in order to effectively re-use the old plan in a novel context.

What is known about the BART Plan is not only the plan itself, but also that plan in relation to all the other planning knowledge that is available to the planner. Without the background knowledge most of the old plan would have to be abandoned. With the background knowledge accessible, in many cases it is possible to determine the source of the situation difference, extract out what is in common between the two situations, and use that as a basis for determining a better match for the new situation. For example, some of the background knowledge associated with the BART Plan (see figure 2) depicts that fact that the old and new situations do not match because there is no ticket machine at the NYC subway.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (2.259i, 3.607i) "buy-theatre-ticket"
circle  at (3.614i, 4.511i) "buy-ticket"
circle  at (3.388i, 5.414i) "pay-for-access"
circle  at (4.518i, 3.155i) "buy-ticket-machine"
circle  at (4.744i, 4.736i) "bart-proc"
arrow from (3.000i, 4.101i) to (3.381i, 4.355i)
"isa*" at (2.908i, 4.038i)
line from (2.492i, 3.762i) to (2.816i, 3.974i)
arrow from (3.494i, 4.991i) to (3.452i, 5.146i)
"isa" at (3.508i, 4.927i)
line from (3.551i, 4.779i) to (3.522i, 4.864i)
arrow from (4.045i, 3.868i) to (3.769i, 4.278i)
"isa" at (4.087i, 3.805i)
line from (4.362i, 3.388i) to (4.129i, 3.741i)
arrow from (4.624i, 3.918i) to (4.560i, 3.431i)
"step1" at (4.631i, 3.981i)
line from (4.701i, 4.461i) to (4.638i, 4.045i)
```

Figure 2: Some background knowledge made explicit.

Moreover, it represents that ‘buying a ticket’ is what is in common between the two situations. Furthermore, an alternate plan which better fits the situation, buying a theatre ticket, is represented as a part of the background knowledge (i.e. buying a NYC subway token is more similar to buying a theatre ticket than it is to buying a BART ticket). Adaptive planning develops techniques for exploiting the background knowledge and determining a better plan match for the new situation.

For PLEXUS the explicit background knowledge is represented in the form of a knowledge network. Since Woods (1975) [. woo75 .] much of research concerning knowledge networks has emphasized epistemic issues. This paper will describe a network at what Brachman (1979) [. bra79 .] has referred to as the conceptual level. Practically speaking, this means that I will be describing relations between planning concepts that are intuitively appealing and assume that these relations can be defined in a suitable epistemic network such as the ones described by Wilensky (KODIAK, 1985) [. wil84c .], Brachman & Schmolze (KLONE, 1985) [. bra85 .], Hendrix (partitioned networks, 1979) [. hen80 .] and Bobrow & Winograd (KRL, 1977) [. bob77 .].

The PLEXUS knowledge network currently contains four kinds of background knowledge: **taxonomic, partonomic, causal,** and **role.** In total, PLEXUS supports eight relationships amongst plans, steps, actions and their associated roles.

Categorization Knowledge

The *isa* relationship is used to indicate an inheritance relationship between two concepts. It is used to indicate both class-membership and class-subclass relationships. For example an isa relationship can be used to relate the planning concept ‘vehicular travel’ to both its class member (‘car travel’), and its subclass (‘mass transit travel’).

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (2.936i, 6.769i) "vehicular-travel"
circle  at (2.936i, 5.414i) "car-travel"
box  at (1.129i, 5.414i) "car"
circle  at (4.292i, 5.414i) "mass-transit-travel"
arrow from (2.936i, 6.120i) to (2.936i, 6.494i)
"isa*" at (2.936i, 6.056i)
line from (2.936i, 5.689i) to (2.936i, 5.993i)
arrow from (2.019i, 5.414i) to (1.588i, 5.414i)
"role" at (2.160i, 5.414i)
line from (2.661i, 5.414i) to (2.301i, 5.414i)
arrow from (3.572i, 6.134i) to (3.134i, 6.572i)
"isa" at (3.635i, 6.071i)
line from (4.094i, 5.612i) to (3.699i, 6.007i)
```

Figure 3: The isa hierarchy.

Collectively the isa relationships of the network form a taxonomy. As PLEXUS moves up the isa hierarchy it is removing properties from a concept, and as it moves down the isa hierarchy it is adding properties to a planning concept. Thus the isa hierarchy is used not only for property inheritance, but also as a basis for performing two important reasoning functions: abstraction and specialization.

If a particular subclass is salient an asterisk will be appended to the isa relationship. Saliency can occur for any number of reasons, two examples are: to indicate the normal plan (default) or to highlight that a given plan is being used frequently within the current context.

Partonomic Knowledge

The background knowledge includes a partonomic hierarchy. Each plan is represented as a sequence of steps, and, recursively, each step is a plan which, in principle, can be decomposed into a sequence of steps (or actions). For example (see figure 4), steps in the BART Plan include: buying a ticket, entering a BART station through a turnstile, riding the train, and exiting a BART station through a turnstile. Furthermore, there are substeps involved in buying a ticket: putting money into the machine and receiving a ticket in return. In this paper, steps are numbered to indicate their order. Two steps that occur in any order will be depicted by appending the same number to them. If a step must occur concurrently throughout the duration of a plan it will have no number appended to it. Collectively the steps form a partonomic structure. PLEXUS uses the partonomic structure for aid in determining the piece of network which needs to be refitted in a given situation.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (2.033i, 6.995i) "BART-Plan"
circle  at (0.678i, 6.092i) "buy-bart-ticket"
circle  at (2.033i, 6.092i) "enter-bart-station"
circle  at (3.840i, 6.092i) "exit-bart-station"
circle  at (0.226i, 5.188i) "insert-money"
circle  at (1.355i, 5.188i) "receive-ticket"
circle  at (2.936i, 6.092i) "ride-train"
arrow from (1.292i, 6.501i) to (0.911i, 6.247i)
"step1" at (1.384i, 6.565i)
line from (1.800i, 6.840i) to (1.475i, 6.628i)
arrow from (2.033i, 6.515i) to (2.033i, 6.367i)
"step2" at (2.033i, 6.579i)
line from (2.033i, 6.720i) to (2.033i, 6.642i)
arrow from (3.021i, 6.494i) to (3.593i, 6.212i)
"step4" at (2.901i, 6.558i)
line from (2.280i, 6.875i) to (2.781i, 6.621i)
arrow from (0.438i, 5.612i) to (0.346i, 5.435i)
"step1" at (0.466i, 5.675i)
line from (0.558i, 5.845i) to (0.494i, 5.739i)
arrow from (1.038i, 5.605i) to (1.193i, 5.414i)
"step2" at (0.995i, 5.668i)
line from (0.840i, 5.866i) to (0.953i, 5.732i)
arrow from (2.520i, 6.508i) to (2.739i, 6.289i)
"step3" at (2.456i, 6.572i)
line from (2.231i, 6.798i) to (2.393i, 6.635i)
```

Figure 4: Steps and substeps.

Causal Knowledge

The background knowledge includes five types of causal relations: **purpose, reason, goal, preconditions,** and **outcome.** The purpose relation is used to indicate the relationship between a plan (step or action) and the most abstract version of the plan which maintains its purpose. For example, the purpose of ‘buying a BART ticket’ is to ‘gain access to some service’ (see figure 5). The concepts between a plan-step and its purpose form a progression of abstractions which maintain the purpose of the current step. When a step in a plan does not match the current situation, PLEXUS considers as an alternate match only those plans which are specializations of plans which lie along the progression of abstractions as determined by the purpose relation (thus maintaining the purpose of a given step).

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (1.581i, 6.995i) "gain-access"
circle  at (1.581i, 6.092i) "pay-for-access"
circle  at (1.581i, 5.188i) "buy-ticket"
circle  at (1.581i, 4.285i) "buy-bart-ticket"
circle  at (2.936i, 6.092i) "break-in"
circle  at (2.711i, 5.188i) "use-money-machine"
circle  at (2.936i, 4.285i) "use-change-machine"
arrow from (1.581i, 6.572i) to (1.581i, 6.720i)
"isa" at (1.581i, 6.508i)
line from (1.581i, 6.367i) to (1.581i, 6.445i)
arrow from (1.581i, 5.464i) to (1.581i, 5.816i)
arrow from (1.581i, 4.765i) to (1.581i, 4.913i)
"isa" at (1.581i, 4.701i)
line from (1.581i, 4.560i) to (1.581i, 4.638i)
arrow from (2.195i, 6.586i) to (1.814i, 6.840i)
"isa" at (2.287i, 6.522i)
line from (2.704i, 6.247i) to (2.379i, 6.459i)
line from (0.967i, 4.899i) to (0.678i, 5.188i)
"purp" at (1.031i, 4.835i)
line from (1.384i, 4.482i) to (1.094i, 4.772i)
arrow from (0.678i, 5.188i) to (1.461i, 6.748i)
arrow from (2.195i, 4.779i) to (2.492i, 5.019i)
"isa" at (2.118i, 4.715i)
line from (1.800i, 4.454i) to (2.040i, 4.652i)
arrow from (2.816i, 4.765i) to (2.774i, 4.920i)
"isa" at (2.831i, 4.701i)
line from (2.873i, 4.553i) to (2.845i, 4.638i)
```

Figure 5: The purpose of buying a BART ticket.

The reason relation (c.f. dependencies: Doyle, 1979; McDermott & Doyle, 1980; Carbonell, 1983) connects a plan step to some latter step in the plan. The reason relation indicates that there is some causal chain between the two steps. For example, the reason for ’buying a BART ticket’ is that it is needed to ‘enter the BART station’ (see figure 6). Given that a step in a plan has failed and that PLEXUS has proposed an alternate match of the current situation, PLEXUS uses the reason relations to determine which of the latter steps in the plan have been effected.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (1.807i, 6.995i) "bart-proc"
circle  at (0.904i, 6.092i) "buy" "bart" "ticket"
circle  at (3.162i, 6.092i) "enter" "station"
box  at (2.033i, 5.188i) "have-ticket"
arrow from (1.313i, 6.508i) to (1.101i, 6.289i)
"step1" at (1.376i, 6.572i)
line from (1.609i, 6.798i) to (1.440i, 6.635i)
arrow from (2.548i, 6.501i) to (2.929i, 6.247i)
"step2" at (2.456i, 6.565i)
line from (2.040i, 6.840i) to (2.365i, 6.628i)
arrow from (1.532i, 5.584i) to (1.849i, 5.329i)
"outcome" at (1.454i, 5.647i)
line from (1.122i, 5.922i) to (1.376i, 5.711i)
arrow from (2.527i, 5.584i) to (2.216i, 5.329i)
"precondition" at (2.605i, 5.647i)
line from (2.944i, 5.922i) to (2.682i, 5.711i)
arrow from (2.195i, 6.092i) to (2.887i, 6.092i)
"reason" at (1.998i, 6.092i)
line from (1.179i, 6.092i) to (1.800i, 6.092i)
```

Figure 6: The reason for buying a BART ticket.

A goal relation connects a step to its associated goals. For example, a goal associated with ‘traveling’ is to be at the destination (see figure 7). Steps can be associated with more that one goal. Collectively that goals form a taxonomy. PLEXUS uses the goal relations for matching an old plan to a situation where the planner’s goals have changed.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
box  at (0.904i, 6.092i) "at-destination"
circle  at (3.162i, 6.092i) "travelling"
arrow from (2.019i, 6.092i) to (1.362i, 6.092i)
"goal" at (2.160i, 6.092i)
line from (2.887i, 6.092i) to (2.301i, 6.092i)
```

Figure 7: A goal of travelling.

Roughly, in PLEXUS, purpose is synonymous with ‘intent’, goal with ‘aim’, and reason with ‘justification’. The purpose of ‘buying a BART ticket’ is to ‘gain access’, the goal associated with it is to ‘have a ticket’, and the reason for doing it it that it makes it possible to ‘enter the BART station’. Functionally these three relations are used quite differently. The purpose relation points to the most abstract version of a plan (the plan in its most rarefied form) and hence is instrumental in finding out what is in common between two situations. The reason relation captures dependencies between steps and is used for determining effects of changes. Goals are used for matching prestored aims to the currently active ones.

A precondition relation (c.f. precondition formula, Fikes & Nilsson 1971) [. fik71 .] connects a step to necessary conditions that must be occurring, or have occurred, in the world as that step is about to be applied. For example, a necessary condition of ‘buying a theatre ticket’ is that a ‘ticket booth’ must exist at the point at which that plan is applied (see figure 8).

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (2.033i, 6.544i) "buy-theatre-ticket"
box  at (0.452i, 6.318i) "exist-ticket-booth"
arrow from (1.228i, 6.424i) to (0.911i, 6.381i)
"prec" at (1.369i, 6.445i)
line from (1.758i, 6.501i) to (1.511i, 6.466i)
```

Figure 8: A precondition on buying a theatre ticket.

An outcome relation connects a step to its expected outcomes. For example, an outcome associated with ‘buying a ticket’ is ‘having a ticket’ (see figure 9). PLEXUS uses the outcome relation to check to see if the expected outcome and actual outcome of an action (or plan) diverge.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (1.807i, 6.318i) "buy" "bart" "ticket"
box  at (3.388i, 6.318i) "have bart ticket"
arrow from (2.612i, 6.318i) to (2.929i, 6.318i)
"outc" at (2.471i, 6.318i)
line from (2.082i, 6.318i) to (2.329i, 6.318i)
```

Figure 9: An outcome of buying a BART ticket.

Role Knowledge

A *role* relation connects a concept to one of its slots. The current version of PLEXUS associates with each of the roles a type constraint which PLEXUS uses to control the process of interpreting a new situation in terms of an old plan. For example, one of the roles associated with ‘buying a ticket’ is ‘ticket’ which is type constrained to be a ‘permission marker’ (see figure 10). Another way to state the relationships between ‘ticket’ and ‘permission marker’ is that ‘ticket’ is the default value of the ‘permission marker’ slot of the ‘buying ticket’ plan. Future versions of PLEXUS will refine the techniques for matching roles to objects by exploiting more of the knowledge associated with the role filler.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (3.162i, 5.640i) "buy-ticket"
box  at (1.355i, 5.640i) "ticket"
box  at (1.355i, 6.769i) "permission-marker"
arrow from (1.355i, 6.233i) to (1.355i, 6.628i)
"type constraint" at (1.355i, 6.169i)
line from (1.355i, 5.781i) to (1.355i, 6.106i)
arrow from (2.245i, 5.640i) to (1.814i, 5.640i)
"role" at (2.386i, 5.640i)
line from (2.887i, 5.640i) to (2.527i, 5.640i)
```

Figure 10: Roles and constraints.

The role relations are also used by PLEXUS for cross indexing purposes. Within a plan class, subclasses are cross indexed by other concepts in the network using the role relation. These distinguishing features are depicted in the network, and in turn relate to some other part of the network. For example one of the distinguishing features of ‘car travel’ is that a ‘car must exist’ (see figure 7). Distinguishing features are used by PLEXUS to control movement up and down the isa hierarchy (c.f. lock/key memory system, Kolodner 1983 [. kol83b .]).

3. Situation Matching

3.1. PLEXUS - An adaptive planner

PLEXUS uses the old plan to interpret its course of action in its current circumstances. It considers the steps, one step at a time, in order. If a step is not an action it adapts substeps in a depth-first fashion before moving onto the next step in the plan. When a given step of the old plan has been adapted to the current circumstances, PLEXUS simulates a planner taking action on that step before moving onto the next step in the plan - thus, as did NASL (McDermott, 1978 [. mcd78 .]), PLEXUS **interleaves planning and acting.**

Associated with each step (substep) in a plan are **appropriatness conditions.** The appropriatness conditions are intended to be suggestive that a particular course of action is reasonable to pursue. Before a step is applied, PLEXUS treats the preconditions and goals of the old plan as appropriateness conditions. After a step has been applied, PLEXUS treats the expected outcomes as appropriateness conditions. Appropriateness conditions are checked by testing the type constraints associated with each of the roles attached to the appropriateness condition. The type constraints are interpreted in terms of the network. Figure 11 shows some examples of appropriateness conditions associated with ‘buying a theatre ticket’: a **precondition** of ‘buying a theatre ticket’ is ‘having money’, a **goal** of ‘buying a theatre ticket’ is to be in ‘possession of the ticket’, and an **outcome** of ‘buying a theatre ticket’ is being in ‘possession of a theatre ticket’.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (1.807i, 6.318i) "buy-theatre" "ticket"
circle  at (0.678i, 5.640i) "have" "money"
circle  at (0.452i, 6.544i) "possess" "theatre" "ticket"
circle  at (3.388i, 6.092i) "possess" "theatre" "ticket"
arrow from (1.024i, 6.445i) to (0.720i, 6.494i)
"goal" at (1.165i, 6.424i)
line from (1.539i, 6.367i) to (1.306i, 6.402i)
arrow from (1.172i, 5.936i) to (0.918i, 5.781i)
"prec" at (1.271i, 6.000i)
line from (1.567i, 6.176i) to (1.369i, 6.064i)
arrow from (2.788i, 6.176i) to (3.113i, 6.134i)
"outcome" at (2.562i, 6.212i)
line from (2.082i, 6.275i) to (2.336i, 6.247i)
```

Figure 11: Some appropriateness conditions.

A rough outline of the top-level decision procedure is shown below:

1) Are any of the before conditions associated with the old plan failing?
   a) Is this a case of step-out-of-order?
   b) Is this a case of failing precondition?
2) Has the current circumstances aroused a goal not accounted for by the current step?
   a) This is a case of differing goals.
3) Is the current step an action?
   a) If yes, perform the action.

b) If no, proceed to adapt substeps.

4) Are any of the outcomes associated with the current step failing?

a) This is a case of failing outcome?

5) Adapt next step.

If one of the before appropriatness conditions fails, or the current circumstances indicate a goal not accounted for by the old plan, one of three different types of **situation difference** is occurring: failing precondition, step-out-of-order, or differing goals. There is a fourth kind of situation difference, failing outcome, that occurs when one of the expected outcomes of a given step fails to occur. Associated with each of the types of situation difference are varying strategies. PLEXUS does not always consider the steps in order, under certain circumstances it looks ahead to the latter steps of the plan and adjusts them in anticipation of certain changes - thus PLEXUS has an element of **opportunism** (Hayes-Roth & Hayes-Roth, 1979) [. hay79 .].

The core of PLEXUS are the matching techniques it uses for finding an alternate version of a step once it determines that the step needs to be refit. To find an alternate matching action for a given situation, PLEXUS treats the failing step as **representative of the category** of action it needs to perform, and then it proceeds to exploit the background knowledge in two ways.

- By a process of **abstraction** PLEXUS uses the background knowledge to determine a category of plans in common between the two situations.
- By a process of **specialization** PLEXUS uses the background knowledge to determine an alternate course of action which is appropriate to the current circumstances.

PLEXUS accomplishes abstraction by moving up the categorization hierarchy until it finds a plan where all the before appropriatness conditions are met. PLEXUS accomplishes specialization by moving down the categorization hierarchy until it finds a plan that is sufficiently detailed to be actionable.

3.2. Core of the Matcher (Managing the Knowledge)

There are at least two important considerations concerning the control of access to knowledge. One consideration is that there is a danger of the planner becoming overwhelmed by the wealth of knowledge (c.f. **saturation,** Davis 1980 [. dav80 .]) that is available. The problem is that there are potentially too many plans that the planner might have to consider, and consequently, the planner could get bogged down in evaluating each candidate plan. Somehow the planner needs to be able to selectively consider the various alternatives available to it.

Another consideration in the control of access to knowledge comes form the cognitive science literature and is referred to as the problem of **enumeration** (e.g. Kolodner, 1983 [. kol83b .]). The problem of enumeration is that humans do not appear to be capable of listing all the instances of a category without some other kind of prompting. When asked to list the states of the union, human subjects do not accomplish this by simply listing all the members of the category of states. For the concerns of adaptive planning the problem of enumeration comes in a slightly different guise. Given an abstract plan it is not reasonable to assume that a human planner could enumerate all of the specializations of that abstract plan.

The first of these considerations dictates that PLEXUS be **selective** in its choice of planning knowledge to use. The second of these considerations acts as a sort of **termination condition:** sometimes the planner knows the right plan but circumstances are such that it cannot find it. As a result of these considerations, PLEXUS' abstraction and specialization processes must be constrained. While moving up the abstraction hierarchy PLEXUS maintains the function of the step in the overall plan. Movement down the abstraction hierarchy, towards more detailed plans, is controlled by the interaction between the planner's knowledge and the current circumstances.

Abstraction

The way to think about abstraction of a plan is that it removes details from that plan; if a particular plan fails to match the current situation, some of the details of that particular plan must be removed. Moving up the abstraction hierarchy removes the details that do not work in the current situation while maintaining much of what is in common to the two situations. Effectively, the movement of abstraction is discovering the generalization which holds between the old and new situations given that a difference has occurred.

A given plan step can have any number of abstractions associated with it. Choosing the wrong abstraction can lead to the wrong action. The planner can avoid this problem by applying the following general rule:

- Ascend the abstraction hierarchy that maintains the **purpose** of the step in the plan that is being refitted.

By moving up the abstraction hierarchy that maintains the purpose of the step, PLEXUS attempts to maintain the function of the step in the overall plan and thereby mitigate the propagated effects of changes.

In general PLEXUS uses two techniques for moving up the abstraction hierarchy.

- If a plan is failing due to the existence of a particular feature of a plan, move to the point in the abstraction hierarchy from which that feature was inherited.
- Incremently perform abstraction on a failing plan.

The first technique applies in situations where there is a specific feature in the old plan that does not exist in the current situation. The second technique of abstraction applies in situations where there is no identifiable feature which has to be removed. In such cases, PLEXUS incrementally moves up the abstraction hierarchy. In either case, for each abstraction it tries to find a specialization that will work in the current context. If it fails to find a specialization for a given abstraction, it moves to the next abstraction in the abstraction hierarchy.

Specialization

Via the process of specialization, PLEXUS moves from a more abstract plan towards more specific examples. PLEXUS navigation through the network is dependent on the planner's current circumstances. PLEXUS descends down the classification hierarchy one step at a time. PLEXUS tests the applicability of a specialization by checking the before appropriateness conditions; if one of these conditions fails, the movement is rejected. At each point in the hierarchy PLEXUS is faced with one of five options:

1) Is the plan sufficiently detailed to act on?
2) Is there a feature suggested by the type of situation difference which cross indexes some subcategory of the current category of plan?
3) Is there an observable feature which cross indexes some subcategory of the current category of plan?
4) Is there an observable feature with an abstraction that cross indexes a subcategory of the current category?

5) Is there a salient subcategory?

PLEXUS stops descending the categorization hierarchy when it gets to a leaf node (option 1). If the node is not a leaf, it continues to descend (options 2-5). Sometimes the type of situation difference suggests cues for subcategory selections (option 2). Sometimes 'observable features' act as cues for subcategory selection (options 3-4). These 'observable features' can either directly cross index some subcategory of plan (option 3), or have an abstraction which cross indexes a subcategory of plan (option 4). Certain subcategories are salient regardless of context and can always be selected (option 5).

Many of these techniques are employed in the following example: Suppose a planner wants to transfer between planes at the Kennedy Airport in NYC. The planner's normal plan for transferring between planes is to walk from the arrival to the departure gate. But when the planner arrives at Kennedy Airport the arrival and departure gates turn out to be in different terminals. Suppose the planner decides that the walk between terminals is too strenuous, and thus a new goal is aroused: preserve energy. The detection of this goal has no correspondent in the old plan and it is determined that the plan must be adjusted to account for this goal; this is a case of the **differing goals** type of situation difference. By a process of abstraction, PLEXUS moves up the categorization hierarchy from the plan to 'walk' to the more general plan of 'travelling'. Next PLEXUS must determine an alternate plan, within the category of 'travelling', from which to act (see figure 12). The newly aroused goal acts as a cue for selecting 'vehicular travel' as a potential subcategory of plan from which to act (option 2). Suppose the planner has never used a shuttle before at an airport, but it sees (observable feature) a sign concerning 'airport shuttles'. An abstraction of 'shuttle' acts as a cue for selecting 'mass transit travel' as a subcategory of 'vehicular travel' (option 4). Moreover, 'shuttle' is a cue for selecting 'shuttle travel' as a subcategory of 'mass transit travel' (option 3). 'Shuttle travel' is sufficiently detailed for PLEXUS to attempt to adapt (option 1).

```
circlerad = 0.265i
boxwid = 0.862i
boxht = 0.265i
circle  at (2.546i, 6.358i) "vehicular-travel"
circle  at (1.273i, 5.085i) "mass-transit"
circle  at (2.546i, 5.085i) "boat"
circle  at (3.394i, 5.085i) "car"
circle  at (4.243i, 5.085i) "cab"
circle  at (0.424i, 4.024i) "shuttle-" "mass-transit"
circle  at (1.273i, 4.024i) "train-" "mass-transit"
circle  at (2.122i, 4.024i) "plane-" "mass-transit"
box  at (-1.061i, 5.722i) "bus"
box  at (-0.212i, 5.722i) "train"
box  at (0.636i, 5.722i) "shuttle"
box  at (-0.212i, 7.207i) "mass-transit-vehicle"
box  at (1.697i, 7.207i) "preserve energy"
box  at (2.122i, 7.843i) "at destination"
circle  at (3.394i, 7.207i) "travel"
arrow from (1.943i, 5.761i) to (2.360i, 6.172i)
"isa" at (1.883i, 5.702i)
line from (1.459i, 5.271i) to (1.823i, 5.642i)
arrow from (0.875i, 4.588i) to (1.114i, 4.880i)
"isa" at (0.829i, 4.528i)
line from (0.583i, 4.230i) to (0.782i, 4.469i)
arrow from (1.273i, 4.581i) to (1.273i, 4.827i)
"isa" at (1.273i, 4.522i)
line from (1.273i, 4.283i) to (1.273i, 4.462i)
arrow from (1.671i, 4.588i) to (1.432i, 4.880i)
"isa" at (1.717i, 4.528i)
line from (1.962i, 4.230i) to (1.764i, 4.469i)
arrow from (2.546i, 5.748i) to (2.546i, 6.099i)
"isa" at (2.546i, 5.688i)
line from (2.546i, 5.344i) to (2.546i, 5.629i)
arrow from (2.950i, 5.755i) to (2.692i, 6.139i)
"isa" at (2.990i, 5.695i)
line from (3.249i, 5.304i) to (3.030i, 5.635i)
arrow from (3.348i, 5.761i) to (2.758i, 6.206i)
"isa" at (3.421i, 5.702i)
line from (4.031i, 5.238i) to (3.494i, 5.642i)
arrow from (-0.988i, 5.854i) to (-0.285i, 7.074i)
arrow from (-0.212i, 5.854i) to (-0.212i, 7.074i)
arrow from (0.564i, 5.854i) to (-0.139i, 7.074i)
line from (1.313i, 5.344i) to (1.485i, 6.570i)
arrow from (0.709i, 6.862i) to (0.146i, 7.074i)
"role" at (0.842i, 6.809i)
line from (1.485i, 6.570i) to (0.975i, 6.756i)
arrow from (0.537i, 4.966i) to (0.617i, 5.589i)
"role" at (0.530i, 4.906i)
line from (0.457i, 4.283i) to (0.524i, 4.846i)
arrow from (2.692i, 7.558i) to (2.393i, 7.710i)
"goal" at (2.804i, 7.498i)
line from (3.162i, 7.319i) to (2.917i, 7.439i)
arrow from (2.055i, 6.842i) to (1.830i, 7.074i)
"goal" at (2.115i, 6.782i)
line from (2.360i, 6.544i) to (2.175i, 6.723i)
arrow from (3.003i, 6.822i) to (3.209i, 7.021i)
"isa" at (2.944i, 6.762i)
line from (2.731i, 6.544i) to (2.884i, 6.703i)
```

Figure 12: Vehicular Travel

3.3. Four types of situation differences

PLEXUS recognizes a number of types of situation difference. Associated with each type of situation difference are various strategies for determining a better match to the current situation. In this section of the paper I will describe four such situation difference types and the adaptive methods for handling each type of situation. These techniques involve identifying the step that needs to be re-fit, determining features which need to be abstracted out, and providing cues for specialization. The techniques are cast in terms of the network relations that were described in the section on background knowledge.

Failing Preconditions

If PLEXUS can not find a correspondent for one of the preconditions of a plan (step) then it is not possible to apply that step and PLEXUS must refit it to the current circumstances. Consider an example:

> The planner is travelling in San Francisco and intends to go to a bookstore to trade some books he has read for some other books he has not read. He intends to adapt an old plan for trading books at Moe's to the new circumstances at the bookstore in San Francisco. As is the case at Moe's, he takes the books that he wants to trade with him to the bookstore. When he arrives at the bookstore he discovers one of the differences between the two situations. At Moe's trading occurs under a fixed policy (for example, paperbacks are traded for a fixed percentage of the cover price). A precondition of the trading step of the Moe's book trade plan is that there exists a fixed policy under which to trade books. When he arrives at the San Francisco bookstore he discovers that at this bookstore there is no fixed policy at work: the type of difference between the two situation is failing precondition.

The type of difference between the two situations is failing precondition. A precondition of the 'trade-via-policy' step in the Moe's Plan is that the bookstore must have some sort of trading policy in force. At the San Francisco bookstore no such trading policy exists.

Given the background knowledge, to circumvent a failing precondition PLEXUS employs the basic mechanisms of abstraction and specialization. For failing preconditions abstraction works as follows:

- Move up the abstraction hierarchy, according to the purpose of the step, to a point at which the *failing condition* has been abstracted out.

For the trading books example, PLEXUS uses the **purpose** relation to select the correct abstraction hierarchy for the failing step 'trade via policy'. It abstracts out the condition that a trade policy must exist by moving up the hierarchy to the more abstract notion of 'trade'. In the event it is unable to find a specialization for trade it would continue move up the abstraction hierarchy, eventually allowing it to consider the notion of 'purchasing' the book. In this case, 'bartering' is a salient subcategory of 'trading', and it is adaptable to the planner's current situation.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (0.452i, 6.995i) "Moe's" "Book-trade"
circle  at (-0.226i, 5.640i) "take-old-books" "-to-Moe's"
circle  at (1.129i, 5.640i) "trade-via" "trade-policy"
circle  at (2.485i, 6.544i) "trade"
circle  at (2.485i, 5.414i) "barter"
circle  at (2.711i, 7.673i) "exchange"
circle  at (3.840i, 6.544i) "purchase"
arrow from (0.099i, 6.289i) to (-0.106i, 5.887i)
"step1" at (0.127i, 6.353i)
line from (0.332i, 6.748i) to (0.155i, 6.416i)
arrow from (0.805i, 6.289i) to (1.009i, 5.887i)
"stepN" at (0.776i, 6.353i)
line from (0.572i, 6.748i) to (0.748i, 6.416i)
arrow from (1.871i, 6.134i) to (2.252i, 6.388i)
"isa" at (1.779i, 6.071i)
line from (1.362i, 5.795i) to (1.687i, 6.007i)
arrow from (2.485i, 6.007i) to (2.485i, 6.268i)
"isa*" at (2.485i, 5.944i)
line from (2.485i, 5.689i) to (2.485i, 5.880i)
arrow from (2.605i, 7.136i) to (2.654i, 7.405i)
"isa" at (2.591i, 7.073i)
line from (2.541i, 6.812i) to (2.576i, 7.009i)
arrow from (3.233i, 7.151i) to (2.908i, 7.475i)
"isa" at (3.296i, 7.087i)
line from (3.642i, 6.741i) to (3.360i, 7.024i)
line from (1.405i, 6.628i) to (1.581i, 7.221i)
"purpose" at (1.391i, 6.565i)
line from (1.200i, 5.908i) to (1.376i, 6.501i)
arrow from (1.581i, 7.221i) to (2.449i, 7.574i)
```

Figure 13a: Trading Books.

```
% prolog
CProlog version 1.2
                .
                .
                .

Working on step trade_via_policy
    Precondition exist policy is failing.
        Can neither delete trade_via_policy nor re-order steps
        Will try abstraction and specialization
            (Must abstract out feature policy )
          Found trade as an abstraction for trade_via_policy
            ( barter is a salient subcategory of barter )
          Found barter as a specialization of trade
Action barter has been adapted and performed.
    (No failing outcome)
                   .
                   .
                   .
```

Figure 13b: Trace of trading books.

Failing outcome

After applying a step in a plan, if PLEXUS cannot find a correspondent for one of the outcomes of the steps, it tries to find a better match for the situation. An example of a situation where a planner encounters a failed postcondition is:

> The planner rents a car for the weekend. When it comes time to exit the car he attempts to use his normal plan. After parking the car, he puts it in neutral, sets the parking brake, and begins to perform a series of steering wheel actions. Two of the effects of the steering actions is to switch the engine off and remove the key. The plan he uses on his own car is to switch the engine off by turning the ignition key, and then remove the key by pulling it out. He tries this plan but when he pulls the key it fails to come out of the ignition lock.

The above is a case of a failed outcome because an expected outcome of the plan is that the planner retains his key. There are two ways in which PLEXUS can adjust to a failed outcomes:

- Find an alternate interpretation of the situation where that outcome is no longer necessary.
- Find an alternate plan which will achieve that outcome.

The basic idea behind the first method is that there are occasions where a planner does not need to change its course of action, but instead must re-interpret the events that are occurring. PLEXUS uses the **reason** relation to check to see if the failing outcome can be circumvented by re-interpreting other steps in a manner consistent with the failed outcome. In the case of removing the car key (see figure 14ab), PLEXUS uses the **reason** relation to check to see if there is an alternate interpretations of entering a car that are acceptable. In this case, the first policy fails, because the planner is unwilling to accept the alternate interpretations, and it must apply the second policy.

For failing outcomes, there is no specific feature to abstract out. Consequently, the only abstraction technique that is applicable to this situation is:

- PLEXUS incrementally moves up the abstraction hierarchy indicated by the purpose relation.

In this case, PLEXUS moves up to the concept of removing keys and attempts to find an alternate specialization that will work in the current context. For failed outcomes the situation type does not provide a specific feature to cross index on, consequently PLEXUS can use all of its specialization strategies. For the example of removing the car key, the plan that failed is the normal plan, so the first specialization strategy fails. When the planner attends to the steering wheel column it observes a button which acts as a cross index to the correct plan: while pushing the button, turn the key and then pull it out.

```
circlerad = 0.225i
boxwid = 0.732i
boxht = 0.225i
circle  at (2.163i, 5.763i) "exit-car"
circle  at (0.541i, 4.862i) "put-car-in-neutral"
circle  at (1.442i, 4.862i) "set-parking-brake"
circle  at (2.344i, 4.862i) "steering-wheel-actions"
circle  at (3.065i, 4.682i) "exit"
circle  at (3.606i, 5.042i) "lock-door"
circle  at (1.803i, 4.141i) "turn-off-engine"
circle  at (3.606i, 3.780i) "pull-out-key"
circle  at (1.262i, 3.600i) "turn-key"
circle  at (4.507i, 4.501i) "remove-car-key"
circle  at (5.228i, 3.600i) "remove-key-" "button"
circle  at (3.245i, 2.879i) "push-button"
circle  at (3.966i, 2.879i) "turn-key"
circle  at (4.507i, 2.879i) "pull-key-out"
circle  at (2.344i, 3.600i) "use-button"
circle  at (4.327i, 2.158i) "engine-off"
circle  at (5.228i, 2.338i) "have-key"
box  at (5.949i, 4.321i) "button"
circle  at (4.507i, 5.583i) "retain-object"
circle  at (5.408i, 4.682i) "have-key"
circle  at (5.769i, 5.763i) "enter-locked" "car"
arrow from (1.290i, 5.279i) to (0.732i, 4.969i)
"step1" at (1.375i, 5.330i)
line from (1.972i, 5.656i) to (1.459i, 5.380i)
arrow from (1.780i, 5.285i) to (1.577i, 5.037i)
"step2" at (1.820i, 5.335i)
line from (2.028i, 5.589i) to (1.859i, 5.386i)
arrow from (2.259i, 5.290i) to (2.299i, 5.076i)
"step3" at (2.248i, 5.341i)
line from (2.208i, 5.549i) to (2.237i, 5.392i)
arrow from (2.637i, 5.194i) to (2.924i, 4.851i)
"step4" at (2.597i, 5.245i)
line from (2.304i, 5.594i) to (2.558i, 5.296i)
arrow from (2.952i, 5.363i) to (3.408i, 5.138i)
"step5" at (2.856i, 5.414i)
line from (2.361i, 5.668i) to (2.761i, 5.465i)
arrow from (2.056i, 4.473i) to (1.932i, 4.321i)
"step1" at (2.090i, 4.524i)
line from (2.214i, 4.682i) to (2.124i, 4.575i)
arrow from (3.008i, 4.293i) to (3.437i, 3.921i)
"step2" at (2.952i, 4.344i)
line from (2.513i, 4.721i) to (2.896i, 4.394i)
arrow from (1.499i, 3.842i) to (1.420i, 3.758i)
"step1" at (1.549i, 3.893i)
line from (1.645i, 3.983i) to (1.600i, 3.944i)
arrow from (4.096i, 4.175i) to (4.332i, 4.366i)
"isa*" at (4.034i, 4.124i)
line from (3.780i, 3.915i) to (3.972i, 4.073i)
arrow from (4.845i, 4.079i) to (4.642i, 4.327i)
"isa" at (4.885i, 4.028i)
line from (5.093i, 3.775i) to (4.924i, 3.977i)
arrow from (4.130i, 3.200i) to (3.454i, 2.958i)
"coord" at (4.259i, 3.251i)
line from (5.020i, 3.521i) to (4.389i, 3.301i)
arrow from (4.535i, 3.206i) to (4.158i, 2.992i)
"step1" at (4.620i, 3.256i)
line from (5.037i, 3.487i) to (4.704i, 3.307i)
arrow from (4.834i, 3.211i) to (4.665i, 3.037i)
"step2" at (4.885i, 3.262i)
line from (5.070i, 3.442i) to (4.935i, 3.313i)
arrow from (2.518i, 3.465i) to (3.070i, 3.014i)
arrow from (4.158i, 2.496i) to (4.231i, 2.355i)
"r" at (4.135i, 2.546i)
line from (4.062i, 2.682i) to (4.113i, 2.597i)
arrow from (4.890i, 2.592i) to (5.048i, 2.468i)
"r" at (4.845i, 2.625i)
line from (4.687i, 2.749i) to (4.800i, 2.659i)
line from (4.507i, 4.721i) to (4.507i, 5.042i)
line from (4.237i, 5.042i) to (4.146i, 5.042i)
"isa" at (4.327i, 5.042i)
line from (4.507i, 5.042i) to (4.417i, 5.042i)
line from (4.146i, 5.042i) to (4.507i, 5.223i)
arrow from (4.507i, 5.223i) to (4.507i, 5.363i)
line from (3.820i, 4.749i) to (3.966i, 5.403i)
"purpose" at (3.808i, 4.699i)
line from (3.656i, 3.994i) to (3.797i, 4.648i)
arrow from (3.966i, 5.403i) to (4.299i, 5.515i)
arrow from (5.634i, 4.011i) to (5.837i, 4.208i)
"r" at (5.589i, 3.966i)
line from (5.386i, 3.758i) to (5.544i, 3.921i)
arrow from (5.042i, 4.608i) to (5.194i, 4.637i)
"outc" at (4.930i, 4.586i)
line from (4.721i, 4.546i) to (4.817i, 4.563i)
```

```
arrow from (5.577i, 5.200i) to (5.476i, 4.890i)
"prec" at (5.594i, 5.251i)
line from (5.701i, 5.555i) to (5.611i, 5.301i)
arrow from (5.166i, 5.166i) to (5.611i, 5.606i)
"reason" at (5.115i, 5.115i)
line from (4.665i, 4.659i) to (5.065i, 5.065i)
```

Figure 14a: Removing the ignition key.

```
% prolog
CProlog version 1.2
                    .
                    .
                    .
Working on step pull_out_key
Action pull_out_key has been adapted and performed.
    Outcome [have, key] is failing.
The reasons for doing pull_out_key are [enter_locked_car]

Projecting forward to step enter_locked_car
++++++++++++++++++
Entering default mode
Assuming following conditions are failing: [[have, key]]
            (Doing incremental abstraction)
    Cannot adapt enter_locked_car
Ending projection on step enter_locked_car
Ending default mode
++++++++++++++++++

Cannot find reasonable alternative for step enter_locked_car
        (Will try to find another plan for pull_out_key )
            (Doing incremental abstraction)
          Found remove_car_key as an abstraction for pull_out_key
            ( remove_key_button is cross indexed by button )
               (Used observable feature button )
          Found remove_key_button as a specialization of remove_car_key
Action remove_key_button has been adapted and performed.
                    .
                    .
                    .
```

Figure 14b: Trace of removing the ignition key.

Differing Goals

There are a large number of cases where a situation difference is caused by the planner determining a goal in the current context that does not have a correspondent in the old plan. A third type of situation difference are the cases where a new goal has arisen which causes the planner to refit its old plan. Again consider the transfer-between-airplanes planning problem example:

> A planner wants to transfer planes at the Kennedy Airport in NYC. His normal plan for transferring planes is to walk from the arrival gate to the departure gate. But at the Kennedy airport the arrival and departure gates are in different terminals. The assumption is that the planner knows about shuttle, but has never taken one at an airport.

The planner decides that the walk between terminals is too strenuous, and thus the above situation arouses a new goal: preserve energy. The detection of this goal has no correspondent in the old plan and it is determined that the plan must be adjusted to account for this goal. The only abstraction technique that is applicable to this situation is:

PLEXUS incremently moves up the abstraction hierarchy indicated by the purpose relation.

In this case, PLEXUS moves up to the general concept of ‘travel’. (see figure 15ab).

Next PLEXUS must determine an alternate way of traveling within an environment other than walking. For situation differences concerning goals PLEXUS has the following constraint on specialization:

- The new plan must be indexed under both the old and new goals.

In some cases this constraint is sufficient to determine the specialization. In this case the preservation goal cross indexes the category of plans concerning vehicular travel, but not a course of action. To determine a course of action PLEXUS must continue to specialize. For the situation of differing goals PLEXUS can avail itself of all of its specialization rules: In this case none of the normal specializations are applicable. It might be possible to take a cab. PLEXUS observes a sign concerning shuttle service which triggers the recognition that mass transit travel, in particular shuttle-riding, might be applicable.

```
circlerad = 0.182i
boxwid = 0.590i
boxht = 0.182i
circle  at (0.145i, 5.083i) "travel"
circle  at (2.326i, 4.501i) "airplane-xtransfer"
circle  at (1.599i, 3.774i) "walk"
circle  at (2.907i, 3.774i) "departure"
circle  at (0.436i, 4.356i) "vehicular-travel"
circle  at (1.163i, 5.083i) "bipodal-" "motion"
circle  at (1.744i, 4.646i) "bicycling"
circle  at (0.145i, 3.774i) "boat"
circle  at (0.581i, 3.774i) "car"
circle  at (1.017i, 3.774i) "cab"
circle  at (-0.727i, 3.048i) "shuttle"
circle  at (-0.291i, 3.048i) "train"
circle  at (0.145i, 3.048i) "plane"
circle  at (-0.436i, 3.774i) "mass-transit"
box  at (-0.581i, 5.664i) "to-destination"
box  at (-0.436i, 4.937i) "preserve-energy"
box  at (-1.453i, 4.792i) "mass-transit-vehicle"
box  at (-2.616i, 3.920i) "bus"
box  at (-1.744i, 3.920i) "shuttle"
box  at (-1.017i, 3.920i) "train"
arrow from (1.935i, 4.115i) to (1.726i, 3.902i)
"step2" at (1.976i, 4.156i)
line from (2.198i, 4.374i) to (2.017i, 4.197i)
arrow from (2.634i, 4.115i) to (2.798i, 3.915i)
"step3" at (2.603i, 4.156i)
line from (2.435i, 4.360i) to (2.571i, 4.197i)
arrow from (0.845i, 4.456i) to (0.282i, 4.964i)
"isa*" at (0.886i, 4.415i)
line from (1.463i, 3.893i) to (0.927i, 4.374i)
arrow from (0.286i, 4.737i) to (0.209i, 4.914i)
"isa" at (0.300i, 4.696i)
line from (0.372i, 4.524i) to (0.313i, 4.656i)
arrow from (1.372i, 4.451i) to (1.217i, 4.914i)
"isa" at (1.385i, 4.410i)
line from (1.544i, 3.942i) to (1.399i, 4.369i)
line from (1.508i, 3.929i) to (1.163i, 4.501i)
arrow from (0.681i, 4.774i) to (0.300i, 4.992i)
"purpose" at (0.749i, 4.733i)
line from (1.163i, 4.501i) to (0.818i, 4.692i)
arrow from (1.422i, 4.892i) to (1.308i, 4.978i)
"isa" at (1.472i, 4.851i)
line from (1.599i, 4.751i) to (1.522i, 4.810i)
arrow from (0.300i, 4.088i) to (0.359i, 4.197i)
"isa" at (0.282i, 4.047i)
line from (0.223i, 3.933i) to (0.263i, 4.006i)
arrow from (0.504i, 4.083i) to (0.477i, 4.183i)
"isa" at (0.513i, 4.042i)
line from (0.540i, 3.947i) to (0.522i, 4.002i)
arrow from (0.699i, 4.092i) to (0.563i, 4.229i)
"isa" at (0.740i, 4.051i)
line from (0.890i, 3.902i) to (0.781i, 4.011i)
arrow from (-0.577i, 3.434i) to (-0.500i, 3.606i)
"isa" at (-0.590i, 3.393i)
line from (-0.663i, 3.216i) to (-0.604i, 3.352i)
arrow from (-0.368i, 3.429i) to (-0.400i, 3.602i)
"isa" at (-0.359i, 3.388i)
line from (-0.327i, 3.220i) to (-0.350i, 3.347i)
arrow from (-0.164i, 3.434i) to (-0.327i, 3.634i)
"isa" at (-0.132i, 3.393i)
line from (0.036i, 3.188i) to (-0.100i, 3.352i)
arrow from (0.041i, 4.092i) to (0.286i, 4.256i)
"isa" at (-0.018i, 4.051i)
line from (-0.286i, 3.874i) to (-0.077i, 4.011i)
arrow from (-0.981i, 4.324i) to (-1.363i, 4.701i)
"r" at (-0.945i, 4.288i)
line from (-0.563i, 3.902i) to (-0.908i, 4.251i)
arrow from (-2.003i, 4.383i) to (-1.576i, 4.701i)
"isa" at (-2.053i, 4.342i)
line from (-2.494i, 4.011i) to (-2.103i, 4.301i)
arrow from (-1.590i, 4.379i) to (-1.485i, 4.701i)
"isa" at (-1.603i, 4.338i)
line from (-1.712i, 4.011i) to (-1.617i, 4.297i)
arrow from (-1.245i, 4.379i) to (-1.408i, 4.701i)
"isa" at (-1.226i, 4.338i)
line from (-1.063i, 4.011i) to (-1.208i, 4.297i)
arrow from (-0.045i, 4.678i) to (-0.295i, 4.846i)
"goal" at (0.014i, 4.637i)
line from (0.286i, 4.456i) to (0.073i, 4.597i)
arrow from (-0.259i, 5.410i) to (-0.463i, 5.573i)
"goal" at (-0.209i, 5.369i)
line from (0.005i, 5.192i) to (-0.159i, 5.328i)
```

Figure 15a: Transferring Planes.

```
% prolog
CProlog version 1.2
                    .
                    .
                    .
Working on step walk
        preserve_energy is a new goal.
            (Doing incremental abstraction)
          Found travel as an abstraction for walk
            (Goal preserve_energy is a cross index for vehicular_travel )
            ( shuttle is a mass_transit_vehicle )
               (and mass_transit_vehicle cross indexes mass_transit )
               (Used observable feature shuttle )
            ( shuttle_mass_transit is cross indexed by shuttle )
               (Used observable feature shuttle )
          Found shuttle_mass_transit as a specialization of mass_transit
Action shuttle_mass_transit has been adapted and performed.
    (No failing outcome)
                    .
                    .
                    .
```

Figure 15b: Trace of transferring planes.

Step out of order

Sometimes PLEXUS is trying to apply an old plan to new circumstances and the situation is such that PLEXUS must apply a step out of order. An example of a situation where PLEXUS encounters a situation difference of the type ‘step out of order’ is:

> A student planner is travelling in France. While in Paris he intends to eat at a student cafeteria because the meals are inexpensive. He has never been to a french student cafeteria before, so the plan he recalls is his normal cafeteria eating plan (see figure 16ab). His normal plan for eating at a cafeteria is that first he selects food and then he pays for it. But when he gets to the french student cafeteria, it turns out that he must pay first.

There are two kinds of adjustments that are possible when a planner perceives a step out of order.

- The intermediate steps should be deleted.
- The steps of the old plan need to be re-ordered.

In order to determine which kind of adjustment is applicable, PLEXUS treats the outcomes of the intermediate steps as cases of failed outcomes and tests to see if it is possible to find alternate interpretations of the situation where the failed outcome is no longer necessary. In the event that it succeeds in this task, PLEXUS assumes that the intermediate steps can be deleted and moves on to the next step in the plan. In the next section of the paper we will see an example of this case, but for the french student cafeteria example PLEXUS determines that the intermediate steps are necessary (i.e. if the planner is going to eat it needs to select food), and consequently attempts to re-order the steps. The method for testing if the steps can be re-ordered is:

- PLEXUS sequences through succeeding steps testing their preconditions.
- If one of the other steps can be applied, PLEXUS removes it from the sequence of steps, applies it, and proceeds with trying to apply the failing step.
- If none of the other steps can be applied, PLEXUS treats this as a case of failing precondition.

In the case of the french student cafeteria, PLEXUS can apply the step to ‘pay the cashier’. After simulating action on that plan, PLEXUS proceeds with trying to ‘select food’.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (1.807i, 6.995i) "restaurant" "eating"
circle  at (1.807i, 6.092i) "cafeteria" "eating"
circle  at (0.904i, 4.962i) "select" "food"
circle  at (2.259i, 4.962i) "pay" "cashier"
arrow from (1.807i, 6.572i) to (1.807i, 6.720i)
"isa" at (1.807i, 6.508i)
line from (1.807i, 6.367i) to (1.807i, 6.445i)
arrow from (1.327i, 5.492i) to (1.073i, 5.181i)
"step1" at (1.376i, 5.555i)
line from (1.638i, 5.873i) to (1.426i, 5.619i)
arrow from (2.040i, 5.499i) to (2.160i, 5.224i)
"step2" at (2.019i, 5.562i)
line from (1.906i, 5.831i) to (1.998i, 5.626i)
```

Figure 16a: Eating at a French Student Cafeteria

```
% prolog
CProlog version 1.2
                    .
                    .
                    .
Working on step select_food
    Precondition access food is failing.
        Can re-order steps.  Will perform step pay_cashier
Action pay_cashier has been adapted and performed.
    (No failing outcome)

Working on step select_food
Action select_food has been adapted and performed.
    (No failing outcome)
                    .
                    .
                    .
```

Figure 16b: Trace of eating at a french student cafeteria.

3.4. An Extended Example

In this section of the paper we will take an extensive look at the BART-NYC Subway example. In the process of adapting the old BART plan to the novel circumstances of the NYC subway, PLEXUS will bring to bear situation matching techniques associated with three of the situation types previously mentioned.

In addition to the technical aspects of this discussion, there will be three recurrent themes which correspond to the three underpinnings of adaptive planning. That is to say, the background knowledge has been made explicit. Moreover, knowledge flexibility is being achieved by exploiting the structure of the background knowledge. Finally the ways in which PLEXUS detects and responds to differences results from a process of situation matching.

Assume that an adaptive planner intends to ride the NYC subway for the first time and that the planner is reminded of the BART Plan. The planner arrives at the NYC subway, descends the stairs, and begins to look for a ticket machine. None of the observable objects are ticket machines. It rejects the notion that the candy machine is somehow relevant. Consequently the precondition that there 'exist a ticket machine' is failing and PLEXUS must adapt the first step of the old plan (see figure 17ab).

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (2.259i, 3.607i) "buy" "theatre" "ticket"
circle  at (3.614i, 4.511i) "buy-ticket"
circle  at (3.388i, 5.414i) "pay" "for" "access"
circle  at (4.518i, 3.155i) "buy" "ticket" "machine"
circle  at (4.744i, 4.736i) "bart" "procedure"
circle  at (5.647i, 4.285i) "use" "money" "machine"
box  at (2.033i, 4.962i) "ticket-office"
box  at (5.873i, 5.414i) "machine"
arrow from (3.000i, 4.101i) to (3.381i, 4.355i)
"isa*" at (2.908i, 4.038i)
line from (2.492i, 3.762i) to (2.816i, 3.974i)
arrow from (3.494i, 4.991i) to (3.452i, 5.146i)
"isa" at (3.508i, 4.927i)
line from (3.551i, 4.779i) to (3.522i, 4.864i)
arrow from (4.045i, 3.868i) to (3.769i, 4.278i)
"isa" at (4.087i, 3.805i)
line from (4.362i, 3.388i) to (4.129i, 3.741i)
arrow from (4.624i, 3.918i) to (4.560i, 3.431i)
"step1" at (4.631i, 3.981i)
line from (4.701i, 4.461i) to (4.638i, 4.045i)
arrow from (5.118i, 3.762i) to (5.449i, 4.087i)
"isa" at (5.054i, 3.699i)
line from (4.715i, 3.353i) to (4.991i, 3.635i)
arrow from (2.132i, 4.376i) to (2.054i, 4.821i)
"o" at (2.139i, 4.313i)
line from (2.209i, 3.875i) to (2.146i, 4.249i)
arrow from (5.781i, 4.941i) to (5.845i, 5.273i)
"o" at (5.767i, 4.878i)
line from (5.704i, 4.553i) to (5.753i, 4.814i)
line from (4.384i, 4.489i) to (4.292i, 5.414i)
"purpose" at (4.391i, 4.426i)
line from (4.489i, 3.431i) to (4.398i, 4.362i)
arrow from (4.292i, 5.414i) to (3.664i, 5.414i)
```

Figure 17a: No Ticket Machine

```
% prolog
CProlog version 1.2
                .
                .
                .
Working on step buy_ticket_machine
   Precondition exist ticket_machine is failing.
      Can neither delete buy_ticket_machine nor re-order steps
      Will try abstraction and specialization
         (Must abstract out feature ticket_machine )
       Found buy_ticket as an abstraction for buy_ticket_machine
         ( buy_theatre_ticket is a salient subcategory of buy_theatre_ticket )
       Found buy_theatre_ticket as a specialization of buy_ticket
Action buy_theatre_ticket has been adapted and performed.
   (No failing outcome)
                .
                .
                .
```

Figure 17b: Trace of the no ticket machine situation.

In this case the type of situation failure is failing precondition. PLEXUS finds an abstraction, i.e. ‘buy ticket, which does not incorporate the failing precondition, by moving up the ‘isa’ hierarchy in the direction indicated by the **purpose** relation. PLEXUS specializes that plan by cross indexing the abstract plan with ‘ticket office’, which is an observable object in the planner’s immediate environment. The values of roles in the BART Plan are propagated to the plan to ‘buy a theatre ticket’, e.g. ‘NYC subway ticket plays the role of ‘theatre ticket’.

PLEXUS successfully applies this version of the step. During course of purchasing the ‘ticket’, PLEXUS is forced to again modify the plan under construction, this time substituting ‘NYC subway token’ for ‘NYC subway ticket’.

Having accomplished a version of the first step, the adaptive planner proceeds to the second step. In this case it must use its procedure for entering BART as a basis for matching the NYC subway entrance situation (see figure 18ab). The first substep of this plan is to insert the token into the entrance machine. The planner successfully accomplishes this step.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (3.388i, 4.285i) "bart-enter"
circle  at (2.485i, 3.155i) "insert-ticket"
circle  at (3.614i, 3.155i) "ticket-returned"
circle  at (4.518i, 3.155i) "enter"
box  at (2.936i, 5.414i) "bart-procedure"
arrow from (2.908i, 3.685i) to (2.654i, 3.374i)
"step1" at (2.958i, 3.748i)
line from (3.219i, 4.066i) to (3.007i, 3.812i)
arrow from (3.508i, 3.692i) to (3.558i, 3.424i)
"step2" at (3.494i, 3.755i)
line from (3.445i, 4.016i) to (3.480i, 3.819i)
arrow from (3.988i, 3.685i) to (4.320i, 3.353i)
"step3" at (3.925i, 3.748i)
line from (3.586i, 4.087i) to (3.861i, 3.812i)
arrow from (3.148i, 4.878i) to (3.289i, 4.546i)
"step2" at (3.127i, 4.941i)
line from (2.993i, 5.273i) to (3.106i, 5.005i)
```

Figure 18a: Entering NYC Subway

```
% prolog
CProlog version 1.2
            .
            .
            .
Working on step bart_enter
To perform bart_enter the following steps must be adapted:
          [insert_ticket,ticket_return,enter_no_barrier]

Working on step insert_ticket
Action insert_ticket has been adapted and performed.
     (No failing outcome)

Working on step ticket_return
     Precondition accessible permission_marker is failing.
          Can re-order steps.  Will perform step enter_no_barrier
Action enter_no_barrier has been adapted and performed.
     (No failing outcome)

Working on step ticket_return
     Precondition accessible permission_marker is failing.
     Will try to treat outcomes of ticket_return as failing outcomes.

Projecting forward to step bart_exit
++++++++++++++++++
Entering default mode
Assuming following conditions are failing: [have,ticket]
               (Doing incremental abstraction)
            Found exit_institution as an abstraction for bart_exit
               (The cross index locking_door of exit_building )
                   (is type constrained by barrier )
                   (and exit_turnstile is of that type)
                   (Used observable feature exit_turnstile )
            Found exit_building as a specialization of exit_institution
Found exit_building as possible alternative for bart_exit
Ending projection on step bart_exit
Ending default mode
++++++++++++++++++

project returns exit_building as a substitution for bart_exit
Step ticket_return can be deleted

Finished adapting steps of bart_enter
Action bart_enter has been adapted and performed.
     (No failing outcome)
            .
            .
            .
```

Figure 18b: Trace of entering NYC subway situation.

The next substep of ‘BART enter’ is that the ticket is returned by the machine, after which the turnstile opens and the planner enters. In this case the token is not returned but it is possible to push through the turnstile. Hence the type of situation difference is ‘step out of order’. Recall that here are two sets of strategies associated with ‘step out of order’:

- Delete intermediate steps.
- Find an alternate plan which reorders steps.

PLEXUS begins by trying the first set of strategies. In order to delete intermediate steps PLEXUS must treat the outcomes of each intermediate step as a case of a ‘failing outcome’ and test to see if the latter steps in the plan affected by the failing outcome can be adapted.

In this case there is only one intermediate step, ‘ticket returned’. The outcome associated with this intermediate step is that the planner ‘has the ticket’ (or in this case ‘token’) (see figure 19ab). PLEXUS applies the first strategy associated with the situation difference type ‘failed outcome’:

Find an alternate interpretation of the situation where that outcome is no longer necessary.

PLEXUS uses the reason relation to associated with ‘ticket return’ to determine which of the latter steps are affected by the failing outcome. In this case, the reason that the ticket is returned is so it can be used when exiting the station (see figure 19ab).

PLEXUS must try to re-interpret exit in such a manner that it can exit without a ticket. This leads to a situation of failed precondition for the step ‘BART-EXIT’. Via abstraction PLEXUS extracts that ‘exiting an institution’ is what is in common between the old plan and the new situation. PLEXUS ‘observes’ the exit turnstile and uses it as a cross index for determining ‘exit_building’ as an alternate plan for ‘exiting the station’, where ‘exit turnstile’ plays the role of ‘locking door’. Since it can find an alternate interpretation to ‘exiting the station’, which does not involve using a ticket, PLEXUS treats the step-out-of-order situation, that occurs during execution of the plan ‘BART enter’, as a case of deletion.

```
circlerad = 0.282i
boxwid = 0.918i
boxht = 0.282i
circle  at (2.033i, 5.640i) "ticket-returned"
circle  at (4.744i, 5.866i) "bart-exit"
circle  at (4.518i, 4.736i) "insert-ticket"
circle  at (3.162i, 4.962i) "have-ticket"
circle  at (5.195i, 6.769i) "exit-institution"
circle  at (5.195i, 7.899i) "exit"
circle  at (1.581i, 6.544i) "bart-enter"
circle  at (2.936i, 7.673i) "bart-procedure"
circle  at (6.776i, 5.640i) "exit" "building"
arrow from (2.661i, 5.259i) to (2.922i, 5.104i)
"outc" at (2.562i, 5.322i)
line from (2.273i, 5.499i) to (2.464i, 5.386i)
arrow from (4.624i, 5.273i) to (4.574i, 5.005i)
"step1" at (4.638i, 5.336i)
line from (4.687i, 5.598i) to (4.652i, 5.400i)
arrow from (3.734i, 4.864i) to (3.431i, 4.913i)
"prec" at (3.875i, 4.842i)
line from (4.249i, 4.786i) to (4.016i, 4.821i)
arrow from (3.551i, 5.767i) to (4.468i, 5.845i)
"reason" at (3.353i, 5.753i)
line from (2.308i, 5.661i) to (3.155i, 5.739i)
line from (5.195i, 7.045i) to (5.195i, 7.221i)
line from (4.744i, 7.221i) to (4.518i, 7.221i)
"isa" at (4.856i, 7.221i)
line from (5.195i, 7.221i) to (4.969i, 7.221i)
line from (4.518i, 7.221i) to (5.195i, 7.447i)
arrow from (5.195i, 7.447i) to (5.195i, 7.624i)
arrow from (4.984i, 6.353i) to (5.075i, 6.522i)
"isa" at (4.955i, 6.289i)
line from (4.864i, 6.113i) to (4.927i, 6.226i)
arrow from (1.821i, 6.064i) to (1.913i, 5.887i)
"step2" at (1.793i, 6.127i)
line from (1.701i, 6.296i) to (1.765i, 6.191i)
arrow from (2.216i, 7.066i) to (1.793i, 6.720i)
"step2" at (2.287i, 7.129i)
line from (2.725i, 7.496i) to (2.358i, 7.193i)
arrow from (3.875i, 6.734i) to (4.546i, 6.064i)
"step4" at (3.812i, 6.798i)
line from (3.134i, 7.475i) to (3.748i, 6.861i)
arrow from (5.929i, 6.247i) to (5.421i, 6.607i)
"isa" at (6.014i, 6.184i)
line from (6.551i, 5.802i) to (6.099i, 6.120i)
```

Figure 19a: Exiting NYC Subway

```
% prolog
CProlog version 1.2
                  .
                  .
                  .
Projecting forward to step bart_exit
++++++++++++++++++
Entering default mode
Assuming following conditions are failing: [have,ticket]
           (Doing incremental abstraction)
         Found exit_institution as an abstraction for bart_exit
           (The cross index locking_door of exit_building )
              (is type constrained by barrier )
              (and exit_turnstile is of that type)
              (Used observable feature exit_turnstile )
         Found exit_building as a specialization of exit_institution
Found exit_building as possible alternative for bart_exit
Ending projection on step bart_exit
Ending default mode
++++++++++++++++++

project returns exit_building as a substitution for bart_exit
                  .
                  .
                  .
Working on step exit_building
Action exit_building has been adapted and performed.
    (No failing outcome)

Finished adapting bart_proc
```

Figure 19b: Trace of exiting NYC subway.

4. Summary

A planner that has access to general plans (alternately abstract or high-level plans) is flexible because such plans will apply to a large number of situations. A problem for a planner working exclusively with general plans is that many of the details associated with more specific plans (e.g. sequencing information and causal relationships) must be recomputed. For a planner that works from more specific plans the situation is reversed: There is a wealth of detail, but there are problems with flexibility. Adaptive planners foreground specific plans, but gain flexibility, in situations where the old plan and the planner's current circumstances diverge, by having access to more general plans.

One application domain for adaptive planners is commonsense planning. Commonsense planning situations have been characterized as being knowledge intensive, but there are other characteristics of these types of situations that make adaptive planning an appropriate model for describing human activity in this domain. A second characteristic is that human planners in such situations appear to work from large chunks of knowledge. That is, rather than taking small atomic operations and using robust techniques to combine them into plans, human planners largely work from previously constructed plans, varying them to meet the demands of the new situation. Varying old plans, as opposed to reconstructing them from scratch, works in commonsense situations because human activity seems to be highly habitualized. A third characteristic of commonsense planning situations has to do with levels of specificity. Due to the habitual nature of commonsense planning situations, and for reasons of parsimony in processing, specific plans tend to be foregrounded

Adaptive planning makes the **background knowledge** associated with an old specific plan explicit. By making the content and organization of the background knowledge explicit, it becomes possible to re-use an old plan in a wider variety of situations.

The PLEXUS knowledge network allows for several kinds of background knowledge: **taxonomic, partonomic, causal,** and **role.** PLEXUS uses the taxonomic structure not only for the purposes of property inheritance, but also as a basis for performing two important reasoning functions: abstraction and specialization. The partonomic structure is used to aid in determining the piece of network which needs to be refitted in a given situation. The causal knowledge serves several functions: It identifies the abstraction which preserves the purpose of the old step, supplies appropriateness conditions, and provides dependency links between steps. The role relations are used by PLEXUS for cross indexing purposes.

The adaptive process can be decomposed into two important components: strategic control and situation matching. Strategic control works as a function of the type of difference that is occurring between the old plan and the new situation. Situation matching works by using the position of the old plan in a planning network as a starting point for finding a match to the planner's current circumstances.

Strategic control performs several functions: It determines the portion of the plan which needs to be refit, provides features which need to be abstracted out of a failing step, and suggests cues for specialization. Each of these strategic characteristics is determined by the type of situation difference.

PLEXUS recognizes four types of situation difference. One type of situation difference occurs when a precondition of a step fails. A second type occurs when the outcome of a performed step does not match the expected outcome from the old plan. A third type of situation difference is caused by the planner determining a goal in the new context that has no correspondent in the old plan. A fourth type of situation difference occurs when the planner is forced to perform a step out of order. Each type of situation difference manifests slight variations on the basic process of accommodating differences.

PLEXUS treats the failing steps of the old plan as representative of the category of action it needs to eventually perform. By a process of situation matching, PLEXUS first determines what is in common between the two situations (abstraction), and then determines an alternate interpretation which is a better match for the new situation (specialization). During abstraction PLEXUS removes the details from the old plan step that made it inappropriate in the current context. During specialization PLEXUS determines a more appropriate example from which to interpret its course of action.

The situation matching process is driven by the **interaction** of planning knowledge and the current situation. The interaction works in both directions. In the direction of planning knowledge to situation, the old plan serves as a basis for interpreting the actions of other agents and the various objects in the new situation. Moreover, it provides the planner with a course of action. In the direction of situation to planning knowledge, it is the situation which provides selection cues that aid the planner in determining an alternate course of action when complications arise (and thereby acts to control access to knowledge).

5. References