Everyday Believability

Garrett A. Pelton Jill Fain Lehman

March 1995
CMU-CS-95-133

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This paper also appears in the IJCAI-95 workshop on Entertainment and AI/ALIFE.

Abstract

Believable Agents must be able to handle a large number of goals, opportunistically acting on them while maintaining a focus of purpose. They also must be able to operate over long life times without significant performance degradations. We describe in this paper a method of managing an agent's goals so that the agent can make progress on as many goals as possible, without spending undue time considering goals that it can't make progress on. The key part of this method is a way of specializing and generalizing the set of perceptual cues that are used to bring a goal into consideration by the agent. The method is described via a simple example.

This research was supported in part by a grant from Martin Marietta and in part by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and ARPA under grant F33615-93-1-1330.

Keywords: Artificial Intelligence, Learning, Plan Execution, Program Transformation, Soar

1 Introduction

Would you be willing to use a travel agent who could schedule trips for at most ten people at a time? Would you use a stock broker who failed to make a small, simple transaction for you, because he was waiting for somebody else's large, complex transaction to finish? How would you feel about a secretary who constantly scheduled other meetings for you during your weekly meeting with the boss? Behaviors like these from humans would be unexpected and jarring. The ability to intend to act, delay action, and remember to act when appropriate for large numbers of goals is an essential characteristic of people performing everyday tasks over long lifetimes. For that very reason, we contend it is also an essential characteristic of believable software agents. No matter what their domain of expertise, if they are to live long and useful lives, they must be able to juggle large numbers of goals of various durations effectively. Without this characteristic, software agents will be believable as absent-minded, inattentive, or forgetful children, at best.

We have created an agent, Laureli, and an environment, Laureli's World, to address this fundamental issue: what structures and mechanisms must an agent have such that she can effectively juggle large numbers of potentially interacting goals of different durations over a long lifetime? By effective, we mean that Laureli must continue to act at a reasonable rate, managing her goals so that she accomplishes most of them in time with the world. Our approach to the complexity of managing many goals is to pay a computational cost only for those goals Laureli can make progress on at any particular time. To do this, Laureli suspends goals when a lack of progress is noticed, at the same time creating a method to reactivate the goal when further progress can be achieved.

A balance must be struck between over-generalization and over-specialization when creating the conditions for activating a goal. The former causes the goal to entail computational cost when no progress can be made, the latter can prevent reactivation when progress is expected to be possible. In this paper we introduce Laureli and her world, and focus on the domain-independent mechanisms for creating, generalizing and specializing reactivation conditions.

2 Architecture

We chose the mechanisms used to create Laureli so that Laureli, and other agents, might be able to survive for long periods of time without significant slowdown. An additional criteria was that the mechanisms be domain independent, so that the essential characteristics outlined in the introduction should be available across domains as part of the basic architecture of the agent. In this section we describe the assumptions and architectural constructs our solution requires.

Figure 1 shows a high-level picture of the relevant portions of the agent architecture. We require six basic components in addition to an external world simulation:[1]

- A perceptual system
- A motor system
- A working memory connected to the perceptual and motor systems
- A long-term rule-based memory

[1] Although Laureli is implemented in Soar [Laird and Rosenbloom, 1987], our list abstracts away from Soar. As shown, the mechanisms can be specified in a Soar-independent fashion, although it must be admitted that these mechanisms work very efficiently in Soar, and taken as a whole, Soar is the only architecture that currently provides all of them.

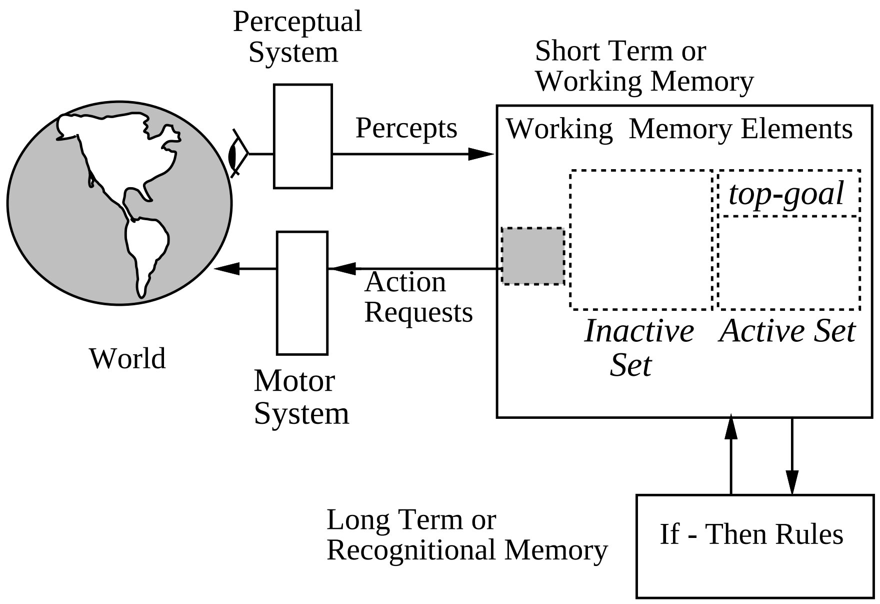


Figure 1: Architecture supporting Laureli

- An efficient match algorithm that triggers rules by instantiating them with working memory elements
- A learning mechanism that creates new rules

The perceptual system places a world description in working memory as a data structure of working memory elements (WMEs). These WMEs describe objects in the world and relationships between objects. The *if* part of the if-then rules in the long-term memory are matched against this description of the world, and any rule that matches can modify the description via its *then* portion. Modification can include the addition of goals that the world description evokes. If, in the service of a goal, certain portions of the working memory are changed (indicated by the grey box in Figure 1), then these changes are interpreted as an action-request by the motor system, which passes them on to the external world simulation.

As the figure shows, a goal may reside in any of three areas: the active set in short-term memory, the inactive set in short-term memory, or in long-term memory. Laureli's behavior at any moment depends on her *top-goal* (dotted box in the figure). This is the goal she has deliberately chosen to pursue during the current processing cycle. The top goal is chosen from among the agent's *active set* of goals in short-term memory. Picking a top-goal involves analyzing each active set goal deciding the goal's importance in relation to other active goals, and deciding whether working on that goal will interact with other goals in such a way that it should not be picked. We call this way of managing the set of appropriate goals *top-level control* of the agent.

The existence of two functionally different memories is key to keeping the agent's top-level control tractable over a long lifetime. Since the computational cost of making importance and conflict decisions is exponential in the number of goals in the active set, we want to keep that part of short-term memory as small as possible. On the other hand, by using the learning mechanism to move goals into long-term memory, the cost of the match algorithm grows at most linearly in the number of rules [Doorenbos, 1994]. Keeping a goal in short-term memory in the inactive set is also more expensive than keeping it in long-term memory because it can increase the combinatorics

Top-level Control	Operating System
1 Only one goal is actively pursued at a time.	1 Only one task can execute on the processor at a time.
2 Goals are only considered when either progress can be made or a conflict arises. Goals that do not contribute to a decision are not included in the decision process.	2 Operating systems maintain two lists of tasks, one with suspended tasks and one with tasks that are available to become the actively executing task.
3 A single task-independent method is used to move goals into and out of active set.	3 A single task-independent method is used to move tasks into and out of active set.
3a *How* to move a goal into and out of the active set is independent of task features.	3a When an event "wakes-up" a suspended task, the OS simply changes what list it is on.
3b *When* to move a goal into and out of the active set is determined by task features and the execution situation.	3b Specific, hard-coded types of events cause a task to be suspended or woken up.

Figure 2: Analogy between top-level control and an operating system

of the match process. Thus, the optimal scheme would be a mechanism that keeps all goals in long-term memory except (1) those that can make progress in the current situation and, (2) those whose projected continuations create a potential resource conflict with a goal satisfying (1). This is the essence of our approach.[2] The scheme is conceptually similar to Birnbaum's "elaborate and index" mechanism for recalling goals [Birnbaum, 1986]. The actual movement mechanisms are described in Sections 3.4 and 3.5.

2.1 Top-level control is like an operating system

The way that moving goals into and out of short-term memory is managed in Laureli is similar to the way multi-tasking is handled in an operating system like Unix. Figure 2 shows the important analogies between operating systems and top-level control. The first item shows that both assume a serial bottleneck in a critical resource. Just as only one task can execute on a serial processor at a time, only one goal can be deliberately pursued by Laureli at a time. Any parallelism in the processing of tasks occurs only in the input and output portions of the architecture. In Laureli, the motor system and perceptual system can operate independently of the top-level control. Likewise, in a multi-tasking operating system some parallelism can exist in the disk controllers and other hardware. The important point is that there is one resource in each system where the processing of tasks is serialized.

Limiting the processing to one goal or task at a time means each system must have a method for switching its executing goal or task. Item 2 of Figure 2 shows that when switching, both systems consider only the goals or tasks that can make progress. In the case of an operating system, for example, you can have many windows on your screen, but the operating system only looks at the one in which you are typing. The other windows receive no processing; they are tasks that are

[2] The current system is an approximation of this optimal scheme in that it uses the inactive set in short-term memory as a temporary redundant place-holder for goals in long-term memory under some conditions. Because the use of short-term memory is redundant, the architecture's efficient match algorithm can still be used to move goals into the active set. There may be some small additional cost to the redundant use of short-term memory, although we have not yet measured this.

suspended waiting for an external event (typing) to make them active. In a similar way, top-level control considers only those goals the agent can make progress on when deciding which goal to pursue. The goals in short-term memory's active set are the goals the agent thinks it can make progress on. The goals not considered, either in short-term memory's inactive set or in long-term memory, are called *suspended*.

Item 2 also refers to the issue of interactions among goals in top-level control and among tasks in an operating system. Interactions can occur between any goals; the status (active or suspended) of the goals is not important. Similarly, the status of the tasks is not important in resolving operating system conflicts such as whether or not another task has entered a mutually exclusive area. Although we recognize this issue as key, we do not address it at this time.

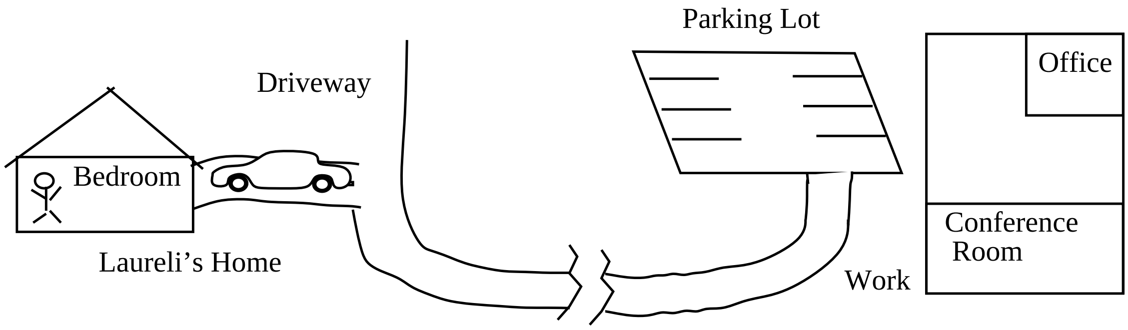
Item 3 in Figure 2 is where Laureli's top-level control starts to differ from an operating system. Both have a task-independent way to move goals (or tasks) from active to suspended status and back. Both define an inability to make progress, or *slack time*, as one reason to move from active to suspended status. However, the operating system has a pre-defined set of conditions (like a new keystroke in a window) that describes when the change from the suspended list to the active list should occur. In our agent, features of the goal being attempted and the plan chosen to achieve the goal determine when the goal should be moved back from long-term to short-term memory. In general, the right set of features has to be determined by analyzing the goal, the plan, and the situation, and cannot be listed a priori. As a by-product of the analysis process, Laureli learns a new rule that effectively transfers the goal to long-term memory. The *if* portion of the rule encapsulates the analysis of when to reactivate the goal. When the match process can bind the conditions in the *if* portion to working memory elements provided by the current world description, the *then* portion of the rule is used to place the goal back into short-term memory.

3 Description of Laureli's Behavior

To ground our discussion of how goals change status, we explore two examples of Laureli's behavior. Laureli lives in the world shown in Figure 3. Day after day, she goes to a meeting at work and then, later, comes home. To get to the meeting, Laureli walks from the bedroom of her home to her car in the driveway, gets into the car, drives to the parking lot, and then walks to the conference room. She follows the inverse path to get home.

So far in her very everyday existence, Laureli has gone to work and come home for more than 300 simulated days. Over the course of that experience, she develops long-term, episodic memories of all her trips, with each trip identified by a unique constant. Besides distinguishing a trip from all the other trips of the same type, this constant provides the thread linking the representations of the goal together as it moves back and forth from short-term memory to long-term memory. We call a goal with an assigned constant a *token* goal, as opposed to a *type* goal (without a constant), because it represents a particular instance of the type.

The bottom of Figure 3 gives a more detailed description of the goal types as we will use them in the rest of this paper. Each goal type has an activation stimulus which is the set of features that have to be present in the perceived world for a goal to be active. As an example, the Goto-work goal arises whenever Laureli looks at her watch and recognizes it is morning and it is a weekday. The goals (eg: Drive-close-to-location) that have negated conditions in their stimulus are subgoals of some other goal. The negated conditions come from a means-ends-analysis difference analysis [Newell and Simon, 1972]. Each goal type also has an expected completion situation that indicates that the goal has been completed. The activating situations and the completion situations



Goal Type	Activation Stimulus	Final Situation
Go-to-meeting	morning & weekday	at conference-room
Go-home	evening & at-work	at Laureli's bedroom
Drive-close-to-location	not(at desired-location) & parkable-location close-to desired-location & desired-location far from current-location & car is near to current-location	close-to desired location
walk-to-location	not(at desired-location) & desired-location near to current-location	at desired location
start-car	not(car-running) & car-running is desired	car-running

Figure 3: Laureli's world and a selection of Laureli's goals

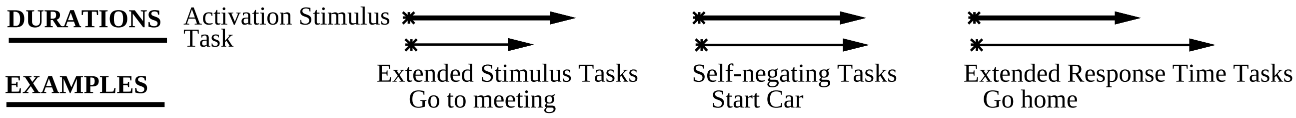


Figure 4: Temporal extent of goals compared to temporal extent of activation stimuli

are independent of each other in general, and the consequences of this independence is discussed next.

3.1 Temporal extent of tasks

In the remainder of this paper, we use the word *task* to mean the combination of a goal and a plan chosen to achieve the goal. Thus, each goal in Figure 3 entails a task and this task takes time to accomplish, i.e. has a temporal extent. In this section, we examine the simplest situations Laureli might find herself in, where each of Laureli's tasks is pursued to completion without interruption. By looking at this simple case we can classify the tasks by properties of their temporal extents.

Figure 4 shows all the possible comparisons of a single task's temporal extent to the temporal extent of the activation stimuli that gave rise to the task's goal. The first column of Figure 4 represents the case where the duration of the activation stimulus exceeds the time it takes to do the task. An example of this situation in Laureli's world occurs for the go-to-meeting goal whose completion-situation (at conference-room) is satisfied while its activation-stimulus (weekday & morning) is also true. Thus, unless a mechanism is added to the architecture shown in Figure 1 to remove this satisfied goal, the goal of going to the meeting will be active even while Laureli is at the meeting. The mechanism that we use is to mark tasks completed when their corresponding goal's completion-situation is satisfied. This, in turn, removes the goal from the active set, although they remain in short-term memory for as long as their activation conditions are true. Of course, since they are no longer in the active goal set, completed goals do not add to the cost of selecting the goal for the top-goal slot.[3]

In the case represented by the second column of Figure 4 (Self-negating tasks), the duration of the stimulus conditions coincides with the duration of the task. This happens whenever the outcome of a task negates some stimulus condition. In Laureli's world, for example, part of the activation stimulus for the goal of starting the car is that the engine is not running. When the engine is running we simultaneously complete the start car task and negate the engine not running stimulus. Thus Laureli recognizes the completion of the start engine task, and she deliberately selects another task to work on. Also, since the completed task's activation conditions are no longer true, it is automatically removed from short-term memory.

In the final case (Extended Response Tasks), the activation stimulus duration is shorter than the time it takes to do the task. An example from Laureli's world is the go-home task.[4] The activation conditions include Laureli being at work. Somewhere on the way home Laureli is no longer considered *at-work*, it doesn't matter where. The loss of the activation stimulus causes the go-home goal to be removed from the active goal set. However, even though the activation stimulus is gone, Laureli doesn't lose her focus on getting home, because the goal of going home is the currently selected top-level goal (i.e. is in the top-goal slot in Figure 1). Since Laureli hasn't deliberately changed what she was doing, she continues with the go-home goal. Of course, if Laureli were to decide to pursue some other goal in the time it takes her to go home, the go-home goal would need some new activation stimuli to remain active. This is the situation we turn to in the next section.

3.2 Deliberate Suspension

Most of the goals Laureli (or any reasonable agent) pursues are not pursued myopically to completion. The first two goals at the bottom of Figure 3, for example, exist for extended periods of time since, in Laureli's world, it takes 20 minutes to drive to work and back. The driving is modeled at a very large grain and takes no effort on Laureli's part once she has set the car in motion (it is more like a taxi ride). It would seem absurd if she sat there simply repeating "go to work, go to work" until she arrived at the parking lot. Similarly, it would be absurd if she attended to other tasks while in motion (like planning what will happen in the upcoming meeting) but failed to note that she had achieved her goal when she arrived at the parking lot.

The generalization of this type of capability is to allow less important tasks some processing power when a more important task is suspended waiting for some external event to occur. However, some care must be taken so that the less important task interferes only minimally with the more important, suspended, task. We don't want Laureli to start driving to the grocery store, or worse, open the door to get out of the car, while driving to the meeting.

In the architecture of Figure 2, switching from the go-to-meeting goal to some other goal means

[3]As mentioned in Section 2, goals on the inactive list do carry some computational cost. However, having the recently completed goals in short-term memory does not seem to be a large issue because the incremental match cost is small and because the completed goals only persist in short-term memory for a limited time.

[4]Obviously, the go-to-work and go-home tasks are symmetric and could both be considered as either extended stimulus or extended response tasks. We have given them asymmetric activation conditions in order to have a simple example of each class in this section, and motivate the different solutions to the problem of forming reactivation conditions in Sections 3.4 and 3.5.

1. Notice Slack time
2. Anticipate expected response, via lookahead in visualized world. This builds the continuation rule.
3. Remove goal from top-level slot, and prevent re-selection.
4. When expected response occurs, the continuation rule matches making the goal available again by moving it to the active set.

Figure 5: Slack time suspension procedure

that Laureli has to deliberately remove the go-to-meeting goal from the top-goal slot and inhibit its reselection as the new top-goal. At the same time, a method needs to be built for bringing the goal back into the active goal set so that it can be re-selected in the future. We call this entire removal and re-activation building process *suspending* the goal. Goal suspension moves the goal into long-term memory until it is reactivated. The issues in suspending the current top-goal are:

When out - We will use slack time. Slack time is the time between requesting an action and getting the expected response. In our example, the only instance of significant slack time is the drive to and from work.

Progress - When the goal gets re-chosen as the top-goal, progress-to-date on that goal has to be represented. We do not describe the progress mechanisms in this paper.

When in - When the slack time is over. In our example, when the car reaches the destination. This is not when the suspended goal is chosen again as the top-goal, but instead when the goal again becomes part of the active set and is, thus, available for selection as the top-goal.

Figure 5 describes at a high level how the suspension of the current top-goal is done. Slack time is noticed by having an action-request sent through the motor system without recognizing its expected response in the perceived world. In step 2, the long-term memory rule is built that, when triggered by the expected response, will make the goal active again. This rule is called the *continuation* of the goal. The conditions of the *if* portion of the continuation are built by an EBL[5]-like process (Soar's chunking mechanism). The learner works with a copy of the world description that has been modified to look like a future world description via a visualization process. By using a one-step lookahead in the plan being used to achieve the goal, the EBL process extracts from all the possible information in the visualized world only those portions pertinent to re-activating the goal. We expand on the visualization and lookahead processes in Section 3.3. Once the continuation has been built, step (3) in the suspension process removes the goal from the top-goal slot and prevents its re-selection. Once step 3 has been done, other goals can contend for the top-goal slot. Of course, any of those goals may also lead to actions with slack time, continuations, etc. However, when the world matches the continuation built in step 2, the goal suspended in step 3 will become active again and can contend for selection as the top-goal of the agent.

3.3 Visualization and Lookahead

The purpose of step 2 in Figure 5 is to build the continuation rule so that, at some time in the future, the suspended goal can become active again. Lookahead and visualization are the methods

[5] Explanation Based Learning [Mitchell *et al.*, 1986]

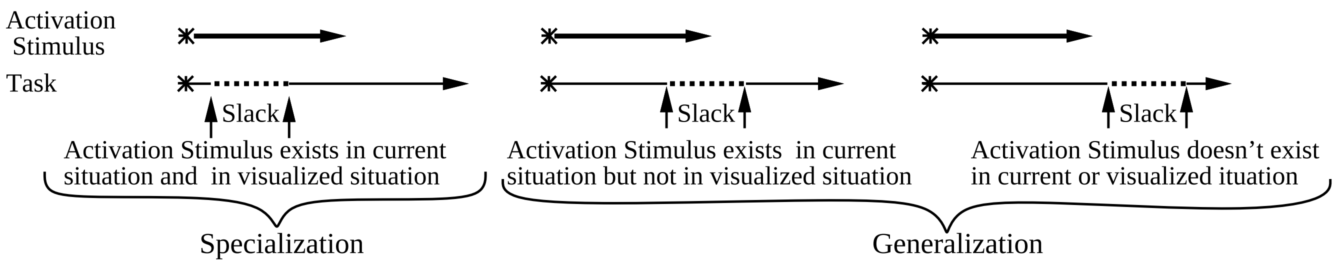


Figure 6: Relationships between slack time and activation stimulus conditions

of including domain-specific knowledge in the continuation in a domain-independent way.

Since the learning mechanism must construct rule conditions that will match some future short-term memory description of the world, we first need to construct a possible version of that future world description. We call this construction process *visualization*. Visualization occurs by augmenting a copy of the current world description with the expected response of the action-request that caused the suspension. The expected response is assumed to be available from some memory of requesting this, or a similar, action in the past.

In general, the visualized world could contain at least as much detail as the world description currently supported by perception. In addition, visualizing the outcome of the action-request adds information that is important to continuing the current task. However, with regard to continuing the task, the visualized world as a whole contains other, unnecessary information. If we included all of the visualized detail in the conditions for continuation, the resulting rule would be hopelessly overspecific. Instead, we want to anticipate a world description in which progress on this task can be made. To focus the EBL process on the correct level of detail, we use the visualized world to execute a single-step lookahead on the plan for achieving the goal. The learning mechanism then automatically produces a continuation rule testing only those elements that the system's current knowledge indicate are critical for making progress.

The lookahead is done in two steps; setting up the goal context, and looking for the next step in the plan. The visualized world is a complete copy of the short-term memory portion of the architecture (see Figure 1), including its own top-goal slot. Setting up the goal context means placing a copy of the real top-goal in the visualized world's top-goal slot. Having the same goal in the visualized world's top-goal as in the real world's top-goal provides us an index into the current plan being followed in the real world.

Once the visualized world's top-goal is selected, we can use the situation found in the visualized world to find the next step in the plan. The search for the next step is encapsulated by the EBL process into the conditions of the long-term memory continuation rule. Having the same short-term memory structure in the visualized world as is used in representing the real world means that the long-term memory rule learned in the visualized world will apply in similar real world situations.

Another way to think about what conditions should be used to reactivate a goal is to focus on the relationship between the activation stimulus and the state of the world at the end of the slack time. In Figure 6 we show the possible relationships between the temporal extent of the activation stimulus and the slack time. If the activation stimuli will exist at the end of the slack time (first column of the figure), then it will exist in the visualized world and we will be able to specialize these activation conditions to build the continuation. We will specialize the original activation conditions by adding some condition that is true in the visualized world (i.e. at the end of the slack time) that indicates further progress can be made on this goal, and that is not true during the slack time.

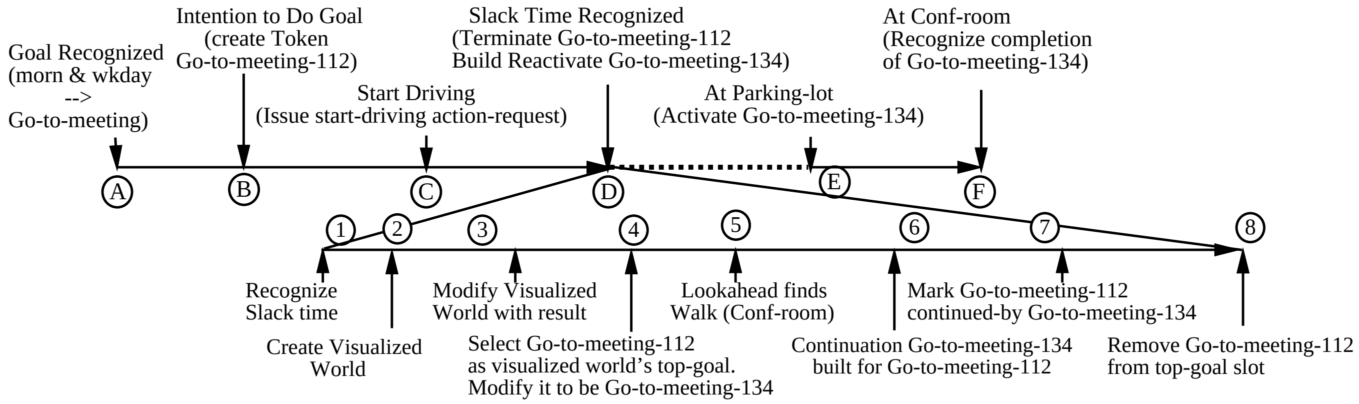


Figure 7: Timeline for the go-to-meeting goal

The other two columns of Figure 6 show cases when the activation stimuli are not true at the end of the slack time, and therefore are not available in the visualized world. The activation conditions for the continuation cannot contain the initial activation conditions, for we know they won't hold, but instead must be formed from some other domain knowledge. We call this case *generalization* because it creates an entirely new set of activation conditions for the continuation of the goal. This new set will match in cases where the original activation conditions would not.

In general, *specialization* is preferred over *generalization* as it provides more restriction on when the goal is in the active set. The *generalization* procedure produces relatively general activation conditions for the continuation because it does not use any specialized domain knowledge to pick the correct set. It just uses the one-step lookahead. Thus continuations generated by the *generalization* process might have to go through the *specialization* process before a reasonably restrictive set of activation conditions are found. In the next two sections, each of *specialization* and *generalization* is explained via examples in turn.

3.4 Specialization

In the *specialization* case the activation stimulus exists throughout the slack time. Thus, to work on some task other than the one associated with the current top goal, we must both suspend the current top goal and create a continuation which adds conditions to the current activation stimulus. The extra conditions restrict the goal from being considered during the temporal interval corresponding to the slack time. As an example, consider the go-to-meeting goal. Its initial activation stimuli are the descriptors *morning* and *weekday* in short-term memory. To take advantage of the time Laureli is en route to the meeting (the slack time), we want to build a continuation that will re-activate the go-to-meeting goal when further work on this goal can make progress. It turns out the descriptor *at-work-parking-lot* is used in conjunction with the initial activation descriptors. Figure 7 shows how creation, suspension, and the specialized continuation for this goal occurs.

As expected, morning and weekday lead Laureli to recognize the goal of going to the meeting Ⓐ. As a result, the token goal go-to-meeting-112 is added to the active set, and is subsequently selected as Laureli's top-goal Ⓑ. Sometime later, Laureli issues the start-driving action request Ⓒ, and soon after, recognizes that she is in slack time Ⓓ. Laureli now begins the visualization and lookahead process described above and expanded in the lower timeline of Figure 7. After recognizing slack time ①, Laureli creates a copy of the current world description ②. She then anticipates the successful conclusion of the action request, i.e. she imagines herself in the parking lot at work ③. Since the

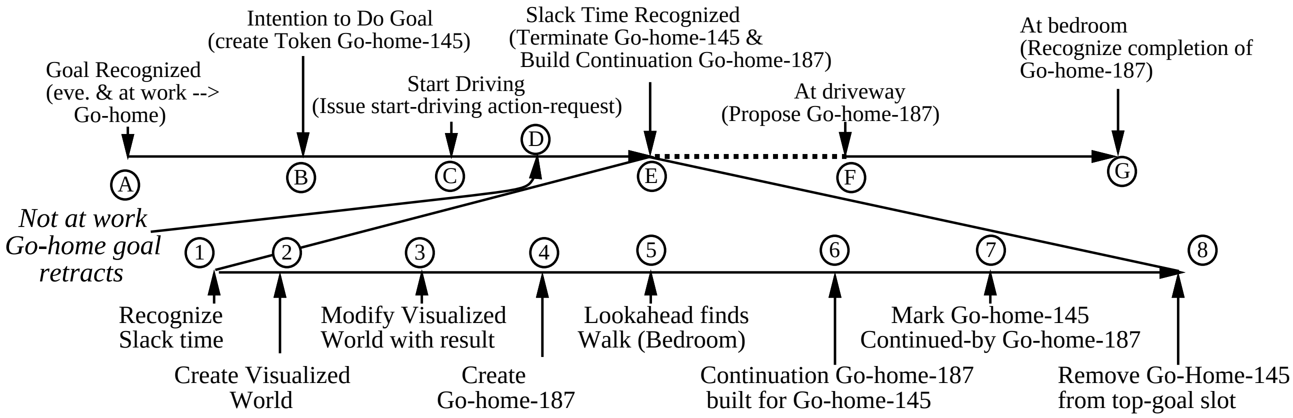


Figure 8: Timeline of go-home goal

initial activation stimuli are still true in the visualized world ④, they will form the context for the lookahead and the basis for the continuation (go-to-meeting-134). Since being at the parking lot is sufficient to enable the next step in Laureli's plan (walking to the conference room) ⑤, lookahead will result in at-work-parking-lot being added to the continuation stimuli for go-to-meeting-134. The conditions and actions form a complete continuation rule that is added to long-term memory ⑥. The go-to-meeting-112 is now marked that it is continued by goal go-to-meeting-134 ⑦. This results in go-to-meeting-112's removal from the active set and from the top-goal slot ⑧, making pursuit of other goals possible in the remaining slack time. When the car arrives at the parking lot, the continuation rule matches allowing the go-to-meeting-134 goal to be selected as the top-goal again Ⓔ. Finally, Laureli gets to the conference room and the go-to-meeting-134 goal is recognized as completed and removed from the active set with no further continuations built Ⓕ.

3.5 Generalization

In the case of *generalization*, the current activation stimuli is not true in the visualized world, and therefore not expected to be true when slack time is over. Thus, we will have to find some descriptors in the visualized world that can act as stimuli for the continuation. Since generalization doesn't rely upon any of the current activation stimuli, it can create a whole new set of activation conditions for continuation. This is similar to creating a disjunctive set of activation conditions for the goal. As an example, consider the go-home goal. Its initial activation stimuli are the descriptors *evening* and *at-work* in short-term memory. To take advantage of the time Laureli is en route to home (the slack time), we want to build a continuation that will re-activate the go-home goal when further work on this goal can make progress. The descriptor *at-driveway-at-home* is such a condition (and will turn out to be the only test on the continuation). Figure 8 shows how the suspension and generalized continuation of the goal occurs.

When she is at work in the evening, Laureli recognizes that she has a goal to go home Ⓐ. As above, the token goal go-home-145 is added to the active set, and is subsequently selected as Laureli's top-goal Ⓑ. Sometime later Laureli issues the start-driving action request Ⓒ. Almost as soon as the driving starts, Laureli is no longer considered *at-work*, and so the initial activation stimuli for the go-home goal are no longer true Ⓓ. Laureli continues to go home because go-home-145 is the currently selected top-goal. However, soon she recognizes that she is in slack time Ⓔ. Laureli now begins the visualization and lookahead process described above and expanded in the lower timeline of Figure 8. After recognizing slack time ①, Laureli creates a copy of the current world description ②. She modifies this copy with the expected result of the action request ③. Since the activation stimuli are not true in the visualized world, the continuation goal (go-home-187) does not automatically include any of the activation conditions of go-home-145 ④. The lookahead can now proceed and finds that when Laureli is at the driveway at home, she should walk to her bedroom ⑤. Since the activation conditions are not true in the visualized state, the conditions for the continuation are just those of the lookahead step. The action component of the continuation rule is to propose go-home-187 as a new goal. This rule now gets added to long-term memory ⑥. Now the go-home-145 goal is marked "continued-by go-home-187" ⑦. This results in the go-home-145 goal's removal from the active set and from the top-goal slot ⑧, making pursuit of other goals possible in the remaining slack time. When the car arrives at the driveway the continuation rule applies creating the go-home-187 goal which is selected to be the new top-goal Ⓕ. Finally Laureli gets to her bedroom, and go-home-187 is marked completed and removed from the active set Ⓖ.

4 Discussion

We are examining the issues that arise when agents live a long time in a changing environment. We believe such an agent requires special mechanisms to allow it to focus on only a small set of its goals at any time. Such mechanisms allow the agent to pursue goals with extended durations by interleaving their substeps with those of other goal-directed activities, as time permits. The work described here has much in common with the approach outlined in [Bratman *et al.*, 1988] and found in systems like HAP [Loyall and Bates, 1993], Runner [Hammond, 1989] and Pareto [Pryor, 1994].

Creation of a method that will dynamically compose the re-activation conditions for a suspended goal distinguishes this work from HAP, where the programmer designates the reactivation conditions at design time. Both Runner and Pareto decide how to continue the goal at run-time by analyzing the goal and choosing from a pre-determined set of re-activation conditions. Neither system can extend the analysis if the initial analysis produces re-activation conditions that are either too specific or too general (in fact, they have no mechanism for determining overgenerality or overspecificity). Laureli can extend its analysis at each instance of suspension by adding or removing domain-specific conditions in a domain-independent way. This allows the system a limited ability to tailor reactivation conditions on the fly as the goal moves through different stages across its duration.

Our methodology seems most similar to the way Hammond's Runner suspends goals and recognizes when they should become active again [Hammond, 1989]. However, Hammond seems to treat all continuations as generalizations. Once a goal has started in Runner, it seems to continue independent of its activation stimulus. As a result, Runner needs a domain model that explicitly indicates which pre-conditions for a goal are persistent, etc., in order to decide which conditions make an appropriate index for his continuation goals. What Hammond is trying to avoid is constantly being reminded of the goal (an over-general index). In Laureli's agent architecture, the specialization mechanism provides a way to add constraint to when a goal is applicable. So if Laureli gets an over-general continuation (usually built by the generalization procedure) the specialization procedure will simply add more conditions to it from other future steps in the plan.

Suspending and reactivating goals per se is only one aspect of this work. Of equal concern is whether we can manage goals in a timely fashion over a long lifetime. At this time, we still have a non-linear slowdown over a life of 300 simulated days. The non-linear slowdown seems to be due to some quadratic algorithms in the matching procedure and we are currently investigating their source. Once we have found the reasons for the quadratic effect, we would expect to see the simple linear slowdown predicted by Doorenbos' [Doorenbos, 1994] theoretical analysis. Since even this linear slowdown will prove untenable in the long run, we are currently looking at ways to achieve a sub-linear bound, at the cost of having to actively reconstruct memories for events in the distant past.

References

[Birnbaum, 1986] L. Birnbaum. Integrated processing in planning and understanding. Technical Report YALEU/CSD/RR 480, Yale University, 1986.

[Bratman *et al.*, 1988] M.E. Bratman, D.J. Isreal, and M.E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4, 1988.

[Doorenbos, 1994] R.B. Doorenbos. Combining left and right unlinking for matching a large number of learned rules. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 451–458, Seattle, WA, August 1994. Soar #94.17.

[Hammond, 1989] K.J. Hammond. Opportunistic memory. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 504–510, Los Altos, CA, 1989. Morgan Kaufmann.

[Laird and Rosenbloom, 1987] A. Laird, J.E. Newell and P.S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[Loyall and Bates, 1993] A.B. Loyall and J. Bates. Real-time control of animated broad agents. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, June 1993. Lawrence Erlbaum.

[Mitchell *et al.*, 1986] Tom M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1), January 1986.

[Newell and Simon, 1972] A. Newell and H. Simon. *Human Problem Solving.* Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[Pryor, 1994] L. M. Pryor. *Opportunities and Planning in an Unpredictable World.* PhD thesis, Northwestern University, June 1994. TR-53-94.