A Plan-Based Personalized Cognitive Orthotic

Colleen E. McCarthy
Department of Computer Science
University of Pittsburgh
colleen@cs.pitt.edu

Martha E. Pollack
Computer Science and Engineering
University of Michigan
pollackm@umich.ed

Abstract

The majority of reminder systems are inflexible; reminders are issued at static, prespecified times. To be effective, cognitive orthotics should reason about what reminders should be issued and when. This paper describes the personalized cognitive orthotic (PCO), a system that uses plan-based reasoning to attain flexibility. PCO relies on local search techniques to generate high-quality reminder plans based on knowledge of the user's plans and her typical behavior. PCO is being developed in concert with other technologies aimed at improved plan management, including systems that update a user's plans and track action execution. We describe the PCO as it is implemented in the Nursebot application: where it provides timely and relevant reminders to elderly people who have cognitive decline that necessitates assistance in managing their daily activities.

Introduction

It has become common practice to use personal organizers to manage our daily activities. Most systems are equipped with reminder capabilities, but these tools are generally inflexible; reminders are issued at static, prespecified times. The goal of our current work is to develop better personal cognitive orthotics, or more generally, better plan management tools. Cognitive orthotics should reason about what reminders should be issued and when, so as to balance the need to (i) ensure that the user is aware of planned activities; (ii) avoid introducing inefficiency into the user's activities (iii) avoid annoyance; and (in some cases) (iv) avoid making the user overly reliant on the reminders.

This paper describes the Personalized Cognitive Orthotic (PCO), a system that uses plan-based reasoning to achieve these goals. PCO relies on local search techniques to generate high-quality reminder plans based on knowledge of the user's daily plan and her typical behavior. PCO is being developed in concert with other technologies aimed at improved plan management, including systems that update a user's plans and track action execution. Currently, PCO is designed to interact with these systems in an overall plan-management system called Autominder (Pollack *et al.* 2001). We describe the PCO in a particular application, the Nursebot project: where it provides timely and relevant reminders to elderly people who have cognitive decline that necessitates assistance in managing their daily activities.

Motivation

The proportion of elderly people in the United States is growing at a phenomenal rate. Currently, 12.5% of the population is age 65 or older; by 2030, this fraction is projected to surpass 20% (Census 1997). In the same time frame, the number of persons residing in nursing homes will double or triple (Rivlin & Wiener 1988). It is generally accepted that quality of life is usually better for people who are living in their own homes, provided they are capable of doing so. However, due to the decline of cognitive function often associated with aging, this option is not always available. The Initiative on Personal Robotic Assistants for the Elderly (Nursebot 2000) is a multi-university research effort[1] aimed at investigations of robotic technology for the elderly. Its initial focus is on the design of an autonomous mobile robot, currently called Pearl, that will "live" in the home of an elderly person and assist in the management of her daily plan.

A central software component of Pearl is Autominder, an automated agent designed to serve as a "cognitive orthotic," assisting an elderly client in carrying out the required activities of daily life (ADLs) by providing her with timely and appropriate reminders. Autominder stores and updates plans representing a user's ADLs, tracks their execution, learns the typical behavior of the client with regard to the execution of these plans, and provides carefully chosen and timed reminders of the activities to be performed.

Autominder differs from most of the technology-based reminder systems in the medical domain. First, most of these systems are geared towards aiding the doctors and caregivers (Cannon & Allen 2000), but even those systems that do address the needs of the patient generally use simple, scripted reminders to prompt the user at prespecified times. They do not adapt to changes such as the introduction of a new plan, or the execution of activities. In general, the current suite of cognitive orthotics lack the power to reason about the relevance, timing, and/or interaction of reminders. Most existing systems lack one of more of the following key ca-

[1] The initiative includes researchers from the University of Pittsburgh, Carnegie Mellon University, and the University of Michigan.

pabilities:

Automatic scheduling
A cognitive orthotic should decide what to reminders to issue and when. Some activities are central to the safety and well-being of the user so there must be some guarantee that the orthotic will issue a reminder for them. However, less critical activities do not always require a reminder, especially if the user typically remembers the activity on her own. In addition, the addition and deletion of reminders should coincide with new user goals and user activity.

Utilization of multiple information sources
In developing a plan of action, the cognitive orthotic should be able to combine information from many sources, including both hard constraints on activity execution and interaction, or the soft constraints of user habits and preferences.

Plan quality assessment
There are many aspects to a high quality plan. Just some of the reminder plan qualities that a cognitive orthotic should address are:

- The spacing between reminders - Reminders are often more effective when they are not issued in quick succession (IPAT 1999).
- Shared properties inherent in the activities, such as location or timing - Reminders for proximal activities should be merged into a single reminder when appropriate.
- Potential overlap between activities - The cognitive orthotic should identify and avoid issuing reminders during the execution of other activities.

The focus of this paper is a new system, the Personalized Cognitive Orthotic (PCO). The PCO was developed to fulfill the requirements above, and to do so efficiently in a dynamic environment. The PCO: identifies those activities that require reminders based on importance and likelihood of being forgotten; determines effective times to issue the reminders; and adapts to environmental changes. The PCO is an integral part of the Autominder system so we use the next section to provide an overview of the Autominder architecture and explain how the PCO interacts with the other components of the system. We then proceed with a detailed description of the PCO algorithm and its implementation

Autominder

As already noted, Autominder is responsible for Pearl's plan management and reminder capabilities. Autominder relies on a number of AI techniques, including interleaved planning and execution, sophisticated temporal reasoning, and reasoning under uncertainty. Figure 1 shows the Autominder architecture. Note that there are three main modules: a Plan Manager (PM), a Client Modeler (CM), and the PCO.

The example we will use throughout this paper involves a day in the life of a typical user who we will call Rose. Rose usually wakes up by 7:00am. She eats breakfast, lunch, and

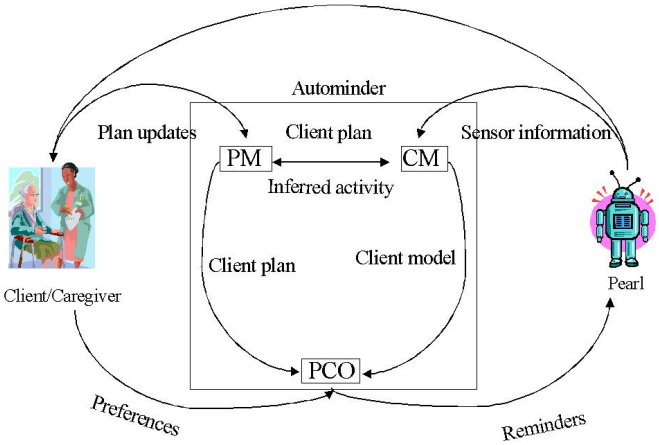

Figure 1: The Autominder Architecture

dinner, and takes medication twice a day[2]. Her caregiver has recommended that she drink water at least 8 times a day and perform her rehabilitation exercises 3 times a day. Rose has a favorite television show at 11:00am, and attends the daily Bingo game at 3:00pm. Updates to Rose's plan (e.g., a new doctor's appointment) can be made by her caregiver. Figure 2 shows a simplified version of these activities, omitting causal and temporal constraints for presentational clarity.

The information provided by the caregiver is used by the Autominder Plan Manager (again, see Figure 1). The PM is an extension of the Plan Management Agent, a prototype intelligent calendar tool (Pollack & Horty 1999). The PM stores Rose's plans, updating them as activity constraints are added, deleted, or modified, and/or as Rose executes activities. A central task for the PM is to ensure that there are no conflicts amongst the user's plans, suggesting alternative ways to resolve any potential conflicts it detects (e.g., suggesting the use of the toilet before leaving for the doctor's office).

Although Figure 2 only shows a simple list of activities, in fact, within Autominder, the daily activities are represented as Disjunctive Temporal Problems (DTP) (Oddi & Cesta 2000; Stergiou & Koubarakis 1998; Tsamardinos 2001). A DTP is an expressive framework for temporal reasoning problems that extends the well-known Simple Temporal Problem (STP) (Dechter, Meiri, & Pearl 1991) and the Temporal Constraint Satisfaction Problem (TCSP) [*ibid.*] by allowing arbitrary disjunctions; e.g read the newspaper before noon or watch the news at 5 p.m. The user plan is also able to support a rich set of temporal constraints between the activities, including relative start time, duration, and minimum and maximum time between activities. The PM uses efficient, constraint-based reasoning algorithms for DTPs to update the users's plans by recording the time of execution

[2]In the early versions of the Autominder, we are not directly issuing reminders about medicine-taking, due to safety concerns: we want to ensure the correctness of Autominder before seeking FDA approval to include medicine reminders. We do however include a range of other modeled activities.

Activity	Earliest Start	Latest Start	Duration
Breakfast	7:00am	11:00am	[30-60]
Lunch	11:00am	4:00pm	[30-60]
Dinner	4:00pm	8:00pm	[30-60]
Med1	7:00am	1:00pm	[5-10]
Med2	11:00am	6:00pm	[5-10]
Hydrate1	7:00am	9:00pm	[2-5]
Hydrate2	7:00am	9:00pm	[2-5]
⋮	⋮	⋮	⋮
Hydrate8	7:00am	9:00pm	[2-5]
Exercise1	7:00am	9:00pm	[15-15]
Exercise2	7:00am	9:00pm	[15-15]
Exercise3	7:00am	9:00pm	[30-30]
Watch TV	11:00am	11:00am	[30-30]
Bingo	3:00pm	3:00pm	[45-75]

Figure 2: Library of Daily Activities (Causal & Temporal Links Omitted)

Activity	Earliest Start	Latest Start
Breakfast	7:00am	9:00am
Med1	7:00am	11:00am
Hydrate1	7:00am	8:20pm
Exercise1	7:00am	7:30pm
Hydrate2	7:05am	8:25pm
⋮	⋮	⋮
Exercise2	7:30am	8:00pm
Hydrate8	7:35am	8:55pm
Exercise3	8:00am	8:30pm
Lunch	11:00am	2:00pm
Watch TV	11:00am	11:00am
Med2	12:00pm	4:00pm
Bingo	3:00pm	3:00pm
Dinner	4:00pm	7:00pm

Figure 3: Extracted User Plan

and propagating any affected constraints to other activities (Tsamardinos 2001). For instance, if Rose is supposed to take medicine no less than two hours after eating, the time for medicine-taking can be made more precise upon learning that the user has begun lunch. Once the constraints have been propagated and the DTP has been shown to be consistent, a single STP is chosen as the user's daily plan. Figure 3 shows a user plan extracted from the DTP constructed for Figure 2. The start and end times of some activities have been constrained to meet the temporal requirements of the activities (e.g. Rose can not do all three sets of exercises at the same time).

An effective cognitive orthotic should be aware of changes in the environment. In some domains information about actions may be direct, such as input to a Palm Pilot. In other domains such as Nursebot, information is obtained via less direct sensors, such as cameras. While these sensors can detect information such as the location of the elderly client, they cannot directly recognize when an activity such as "eat dinner" has been performed. Autominder's next component, the Client Modeler (CM), uses the user plan and sensor information to infer the probability that a planned activity has been initiated or has ended (e.g., going to the kitchen around the normal dinner time may indicate that the user is beginning dinner). This inference is performed using a novel, bi-level dynamic Bayesian network framework (Colbry, Peitner, & Pollack 2001). Over time, the CM should also construct a model of the user's expected behavior (e.g., Rose usually remembers to take medicine in the morning, but frequently forgets in the afternoon); however, this last capability (learning) has not been implemented.

The CM and PM provide input to the third component of Autominder, the Personalized Cognitive Orthotic (PCO). The PCO is designed to balance user compliance and efficiency against annoyance and over-reliance. Reminders should ensure that the user is aware of upcoming activities, but not at the expense of distracting or annoying the user. In addition, adherence to a reminder should not lead to inefficient execution of the user plan, and if possible, the reminder plan should be structured to prevent overreliance on the system. The PCO achieves these goals by using planning techniques and domain knowledge to build and maintain a plan of reminders. This plan is continously updated to react to user actions and changes in user preferences and caregiver recommendations.

PCO

The PCO uses an approach based on the Planning by Rewriting paradigm (PbR) (Ambite & Knoblock 2001). PbR is an anytime planning system that uses local search and rewrite rules to transform suboptimal plans into high quality plans. PbR consists of four stages: (i) generation of an initial solution, (ii) application of rewrite rules to produce new candidate solutions, (iii) evaluation of the candidates, and (iv) selection of the next solution for further expansion. The PbR approach is well-suited to the development of reminder plans since it is easy to generate an initial plan, but harder to generate a high-quality plan. While the number of different activities in a daily plan may be small, the flexibility in the timing of the reminders would create a search space that is prohibitively large for a global search. Using local search allows the planner to develop high-quality plans (important for user satisfaction in our domain) but still provides an anytime property.

The PCO adapts the PbR framework to the requirements of a cognitive orthotic and extends it to support continued maintenance of those plans as the environment changes. Pseudo-code algorithm for the PCO algorithm is given in Figure 4, and we now consider each stage in turn.

PCO

1. Generate initial reminder plan P

Loop:

While no interrupt:

2. a. Use all applicable rewrite rules to create a neighborhood of reminder plans $P_1' \ldots P_n'$

b. For each plan P_i

i. Repair reminder plan as needed

ii. Check for consistency

c. Evaluate reminder plan quality

d. Choose P') with max value

3. If reminder interrupt:

a. Issue reminder;

b. Update reminder plan

4. If other interrupt:

a. Update reminder plan

End Loop

Figure 4: The PCO algorithm

To create an initial reminder plan, the PCO needs only to set a reminder for the earliest start time of every activity. This information is available from the user plan. Reminder plans are represented as a graph structure where each node represents a reminder step and each edge denotes a temporal or causal constraint between the steps. Each reminder step has fields representing the associated user plan step(s) and time of issuance. Causal and temporal information in the reminder plan are inherited from the user plan because the reminders must respect the ordering of the activities in the user plan. As we will discuss later, the reason for maintaining causal structure (in the form of causal links) is to enable appropriate reminder plan update when there are environmental changes. The goal state in the initial reminder plan is a reminder for every activity. Figure 5 (next page) shows an initial reminder plan for our running example, constructed with reminders at the earliest start time.

Generating an initial plan

The rewrite rules

It is easy to see that the initial reminder plan may be far from optimal. In this example, the majority of the reminders are clustered early in the morning, even though some of the activities could (and possibly should) be deferred to the afternoon. Therefore, the next step in the PCO algorithm is to improve the quality of the reminder plan. For this, the PCO uses rewrite rules to generate a neighborhood of candidate plans.

For the current Nursebot application, the rules are created from information about the plan activities and structure, the user's habits and preferences, and any caregiver recommendations. The rewrite rules in the current version of the PCO are designed to:

Rule Type:	Time to Issue:
Preferred PR(A)	At time recommended by caregiver
Expected ExR(A)	After the expected time
Spreadout SOR(A)	At even distribution over time
Conflict CR(A)	Before or after conflicting activity
Merged MR(A)	In combination with other reminders
Earliest ER(A)	At earliest start time
Latest LR(A)	At latest start time

Figure 6: Types of rewrite rules

- use times recommended by the caregiver,
- use times similar or after the expected time of activity execution,
- use earliest and latest start times,
- space out activities of similar types,
- delete reminders for activities that are seldom forgotten by the user, and/or
- merge reminders[3].

Rewrite rules consist of an antecedent and an instantiation. The antecedent matches operators, links, and goals against objects in the reminder plan P. The instantiation of a rule involves deleting a goal or removing some subplan P_i from P and inserting a new (possibly empty) subplan P_j. In Autominder, rewrite rules remove a node designating a reminder for activity A and replace it with another node designating one of the reminder types in Figure 6. Sample rules are shown in Figure 7.

For example, 7(a) shows the useExpected rule, which can replace a reminder to do some activity A at an arbitrary time (perhaps earliest or latest), with a reminder $ExR(A)$ for its expected time. Similarly, the removeReminder rule in 7(b) also uses information from the user model to transform the reminder plan. If there exists a very high probability that the user will remember to perform A on her own, any reminders for A are removed. However, recall that the goal state of the initial reminder plan is to remind for *all* activities. Thus, the removeReminder rule also deletes this top level goal.

The rules shown above affect only a single reminder step or goal, but rewrite rules can also alter larger subplans. For example, one rule in the PCO spaces out reminders for activities of the same type. In Figure 8 we show the reminder plan that results from applying this "SpreadOut" rule to the hydration reminders in the initial reminder plan of Figure 5. As you can see, the EarliestReminder steps for Hydration1 through Hydration8 have been replaced by reminders that spread out the reminder times.

Note that none of the rewrite rules we have described provide information about how to embed the new subplans into

[3]This rule has not been implemented in the current version of PCO.

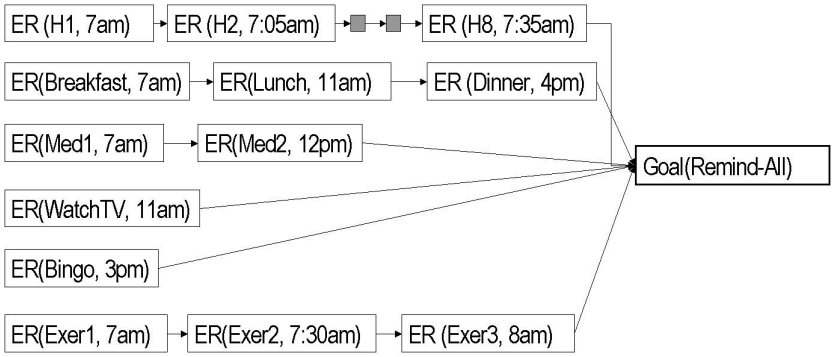

Figure 5: Initial Reminder Plan

```
useExpected(Plan P, Activity A)
{
    // Take a subplan from P
    ReminderStep X(A) = P.opsList(x);

    //Find the step in user plan associated with A
    ClientPlanStep S = A.associatedStep;

    // Find operator Y that uses expected time
    ReminderStep Y = createNewReminder(A,"Expected");

    // If Y exists, remove X and replace with Y
    replace(X(A), Y(A), P);

    return P;
}
```

(a)

```
removeReminder(Plan P, Activity A)
{
    // Take a subplan from P
    ReminderStep X(A) = P.opsList.get(x);

    //Find the step in user plan associated with A
    ClientPlanStep S = A.associatedStep;

    // Find the probability of S being executed
    long Y = getprobability(A);

    // If Y exceeds threshold, remove reminder
    if (Y > THRESHOLD)
        removeFromGoal(S,P);
        replace(X(A), null, P);

    return P;
}
```

(b)

Figure 7: Sample Rewrite Rules

the graph structure of the plan, ie., how to add and delete arcs. However, the PbR approach allows for partially specified rewrite rules. In such cases, rule application is followed by invocation of a planner to complete the plan. The current version of the PCO uses a handcrafted partial-order planner to correct flaws in the candidate reminder plan, therefore, the nodes and edges in the plan in Figure 8 are completely specified.

The rewrite rules do not always produce (valid) reminder plans. That is, when a node is removed or replaced, the resulting plan may be inconsistent with respect to the chaining temporal or causal constraints. For instance, in Figure 9 the useLatest rewrite rule has been applied to the reminder plan in Figure 8. By changing the time for Hydrate1 to its latest possible start time, we have violated the temporal ordering between Hydrate1 and Hydrate2. All invalid candidate plans are detected by the planner and discarded by the PCO.

Evaluation

Once a neighborhood of reminder plans is generated and any invalid plans are discarded, the local search algorithm must evaluate the remaining candidate solutions. Since there is no way to directly measure the effectiveness of each possible individual reminder, we have operationalized evaluation criterion that "stand in" for our actual goals[4].

The first criterion in a reminder plan is that it must include a reminder for all critical activities or else the value is set to $-\infty$. After correctness, the quality of the plan is most greatly affected by the number of reminders, their timing, and their relative spacing.

Number: In general plans with fewer reminders are assessed more highly. Since the PCO guarantees that critical activities are accounted for, having fewer reminders means that unnecessary reminders have been removed and/or reminders have been merged.

Timing: The timing of a reminder can affect plan quality in several ways:

- Reminders issued close to the preferred and/or recommended times should increase client and caregiver sat-

[4]We will later conduct field tests with users and caregivers to validate this correlation.

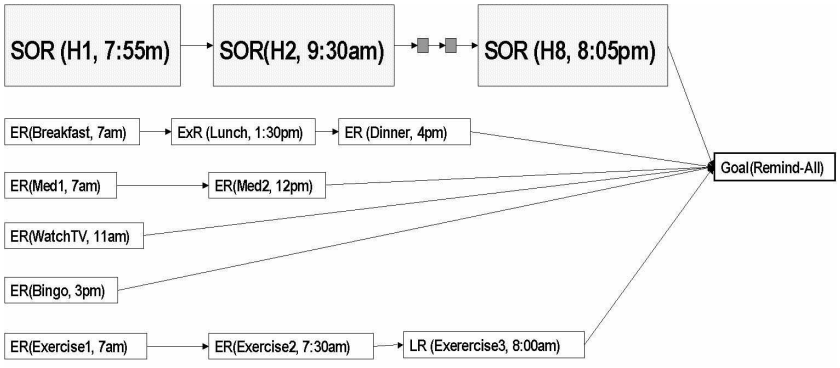

Figure 8: Reminder plan after the application of a SpaceOut rewrite rule

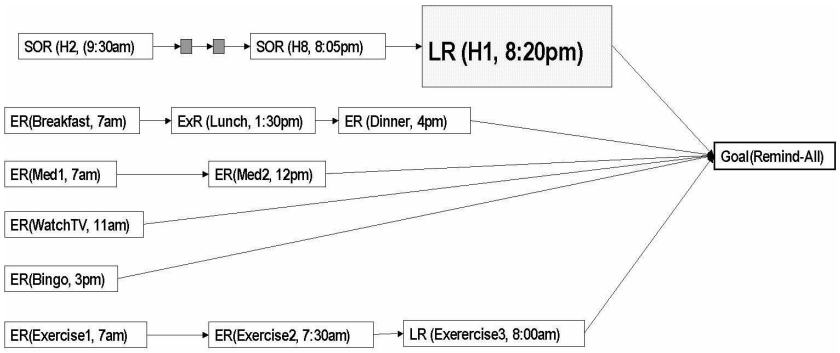

Figure 9: Application of "useLatest"

isfaction.

- Reminders issued *after* the expected time of activity execution should increase client autonomy.

Spacing: The elapsed time between reminders is important in two ways. First, to increase efficiency in plan execution the overall distribution of reminders should avoid any clustering and/or overlap. Second, it is often the case that a single activity should be executed multiple times in a day, such as the exercises and hydration in our current example. In such cases it is especially helpful if these particular activities are performed throughout the day rather than in rapid succession. In both of these cases there is no specific time that is preferred for these activities, rather there exists a preference on the spacing between them.

The relative importance of these different attributes is reflected in the evaluation metrics.

We evaluate plans by looking at the individual reminders and then the plan as a whole. The time associated with each reminder is compared to any specialized input (expected, preferred, etc.). The distance between scheduled reminder times and preferred/expected time is then multiplied by a weighting factor, w_{pref} or w_{esp} respectively. For repeated activities, we calculate the difference between the actual space between each pair of subsequent activities and the optimal space. We then sum these and normalize by dividing by the optimal spacing. We apply similar calculations over all reminders to check that the reminder plan as a whole is evenly distributed.

Assume we weigh the value of the preferred, recommended, and expected times and the value given to evenly spaced activities and plans, $w_{pref}, w_{rec}, w_{exp}, w_{distrib}, and w_{space}$ respectively. In addition, we know the number of reminders (#reminders), number of activities in the client plan (#activities), and the optimal spacing between the reminders (S). Plan evaluation is done as follows:

1) Loop through reminders r
   If $|r_{time} - pref_{time}| < \epsilon$
      Plan += $w_{pref} * |r_{time} - pref_{time}|$
   If $|r_{time} - rec_{time}| < \epsilon$
      Plan += $w_{rec} * |r_{time} - exp_{time}|$
   If $r_{time} > exp_{time}$
      Plan += w_{exp}
2) Plan += Deviation from preferred-spacing * $w_{distrib}$
3) Plan += Deviation from $S * w_{space}$
4) Plan += $(\#activities - \#reminders) \div (\#activities)$

As an illustration, consider the case where the useLatest and useExpected rules are applied to the plan in Figure 8. There are now two new plans to evaluate, shown in Figures 10 & 11. The reminder plan in Figure 10 was created by applying the rule useLatest; by moving the time for Exercise3

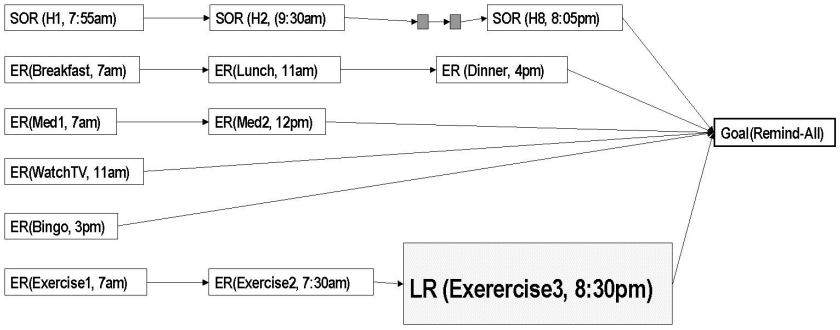

Figure 10: Another application of "useLatest"

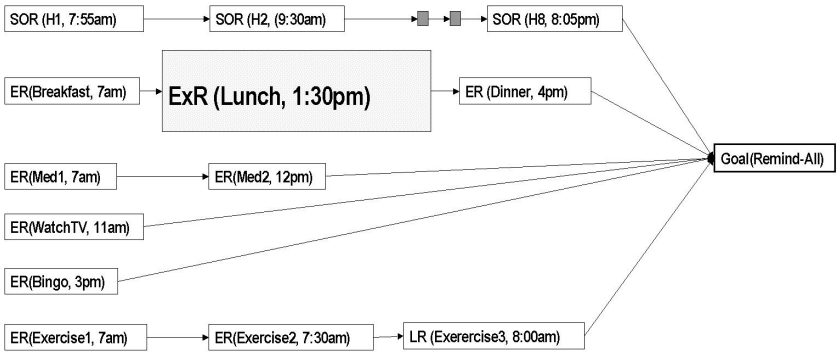

Figure 11: Application of "useExpected"

to 8:30pm. In this scenario the value of the plan increases because average spacing of the plan has improved. In Figure 11 the application of useExpected rewrite rule incorporates user information into the reminder plan by moving the reminder for Lunch to 1:30. This increases Note that the application of a single rewrite rule can affect the plan value in multiple ways. For instance, moving a reminder to its latest time might fortuitously satisfy some criteria not fulfilled in an earlier plan (i.e. the preferred time is also 8:30pm), and vice versa (the preferred time to eat lunch is 12:00pm).

Selection

The final step in the local search of PCO is to choose the reminder plan with the highest value. If the plan quality has exceeded a given threshold, the PCO will pause, otherwise the loop is continued. However, two types of events will cause the algorithm to break the pause or loop: the issuance of a reminder, or an interrupt from the Autominder indicating that there has been a change in the environment. The PCO must react to these changes in order to ensure the validity of the reminder plan and increase user satisfaction.

Reacting to change

As reminders are issued and activity constraints are added, deleted, or modified, and/or as the user executes activities, the reminder plan may need to be updated. At the start of the day, many temporal constraints on the daily activities are extremely flexible, but as time progresses, a number of these constraints will tighten. For example, eating lunch can affect the time bounds for taking medication. If the PCO does not react appropriately to change, reminders may be issued at incorrect times.

Hard constraints in a plan are all the constraints that, when changed, require re-evaluation of, and possible modifications to, the plan. In Autominder, the client plan may change when a new activity is added or time bounds are modified. These changes introduce in hard constraints in the reminder plan, which, after all, must reflect the client's scheduled activities. All changes in hard constraints must be addressed: and the PCO must check whether the reminder plan is still valid, and if necessary, replace any invalidated reminders in the current reminder plan with EarliestReminders (see Figure 12 for algorithm details).

The PCO also handles changes in soft constraints (preferences, recommendations, client model information, etc.). Soft constraints affect the quality of the reminder plan, but not the validity. Consider the case in which the user indicates that she would like to change her preferred reminder for lunch from 11:30am to 12:30pm. Because the time bounds on lunch have not changed, the PCO has the option of ignoring the preference change in time critical situations. The current reminder plan may not reflect the new soft constraints, but it is still valid. However, as shown in Figure 13, if the PCO updates the ExpectedReminder to reflect this new preference, the reminder plan must be checked to ensure that the new reminder time is within the valid time bounds of the

Change in hard constraint

1. If change is plan addition A
   a. add new goal A to reminder plan
   b. add EarliestReminder for A to plan;
2. If time bound change for A
   a. change reminder step to reflect newest start or execution time
   b. check that plan is still valid with newly changed reminder step and bounds,
      i. if not, replace with EarliestReminder

Figure 12: Reacting to changes in hard constraints

Change in soft constraint

1. If Change in preference P
   a. If time-critical
      i. ignore
   else
   b. update operators affected by P
   c. check that plan is still valid with newly changed operators,
      i. if not, replace operator with useEarliest

Figure 13: Reacting to changes in soft constraints

user plan. That is, the validity of the soft constraint must be ensured.

Related Work

As described earlier, the Planning by Rewriting (PbR) paradigm (Ambite & Knoblock 2001) is a planning system that uses local search to generate plans. It works by quickly generating an initial plan and then performing incremental improvement to increase the quality of the plans. PbR offers a number of advantages: it is domain-independent, accepts complex quality metrics (as opposed to quality based only on plan length), and is an anytime algorithm. In addition PbR the uses declarative plan-rewriting rules. PbR balances efficiency and quality and has been shown to generate high-quality plans quickly; PbR was one of only three planners able to solve some of the problems at the 2000 AIPS planning competition (Bacchus Fall 2001). To date, PbR has been used primarily in the domains of query processing and logistics. The PCO extends the PbR algorithm by interleaving planning and execution. As noted above, changes in the environment are handled by the application of rewrite rules or a quick repair to the current reminder plan. The rewrite rules in the PCO can also adjust the goal state as opposed to only the intermediate plan steps.

Execution Systems

Another relevant class of systems oversees plan execution (Pell *et al.* 1996; Bonasso *et al.* 1997). These systems typically include a plan deliberation component to produce plans that are then dispatched to an execution component, or executive, which is responsible for the performance of the actions in the plan. Similar to reminder systems, dispatch systems are responsible for ensuring the timely completion of activities. Efficient algorithms for dispatch have been studied for plans represented as Simple Temporal Problems (STPs) and Temporal Constraint Satisfaction Problems (TCSPs) (Muscettola, Morris, & Tsamardinos 1998). However, these systems don't reason about the necessity of the reminders or their timing. Instead, all activities are executed at the earliest possible time.

Medical reminder systems

There has also been important work done that directly addresses issuing reminders. Kirsch and Levine (Kirsch *et al.* 1987) developed an automated cuing system intended to achieve goals similar to our own. In their system, caregivers input a detailed sequence of steps for target activities and the system provides cues to an individual user about these structured tasks. Their system also monitored time in order to refocus the user in the case of interruption (Kirsch & Levine June 1988). Parente (Parente 1991) also addressed the need for software to assist persons with memory disorders by using expert system techniques to provide cues for complex activities. Like the activities used by Kirsch and Levine, the tasks modeled by Parente are hierarchical. Note that in both of these systems, the reminders are prescriptive – there is no reasoning about the selection or timing of reminders.

The PEAT system (Levinson 1997) goes beyond most other automated medical reminder systems. PEAT uses AI planning technology on a PDA to assist users with traumatic brain injury in planning and executing daily activities. PEAT uses an AI planning system called PROPEL, the PROgram Planning and Execution Language (Levinson 1995). In PROPEL, user-defined scripts are used to guide the planner and execution monitor. Planning involves simulating the script before it is executed. The planner evaluates each simulation with respect to the goals, and it searches for program variations that maximize goal achievement. Finally, the planner generates advice rules that are used during execution to prompt the user through each plan step. PEAT is able to propagate temporal information, but the reminder system is neither selective nor adaptive. Reminders are issued for all activities and the relative time of issuance does not change.

Conclusions

The PCO as described in this paper is fully implemented and integrated with the current Autominder platform. Unlike traditional reminder systems, the PCO uses planning

techniques to generate high quality reminder plans that integrate information about plan structure, caregiver recommendations, and user preferences. The PCO generates timely, relevant reminders. When it is time for a reminder to be issued, a text string is sent to Pearl (the robot platform on which Autominder is deployed). Pearl then does text-to-speech translation and relays the reminder to the user. The PCO is designed to enable the generation of justifications for reminders, in hopes that user adherence to plans may be improved when the reasoning behind the existence and timing of a reminder is provided. For example, a reminder of the form "If you take your medicine now, you will not have to do it in the middle of your TV show," may be more compelling than the generic message "Time for medicine." In generating a justification for a reminder, the PCO will make use of the underlying user plan, the preferences of the caregiver and the user, and the particular rewrite rules used in creating the current reminder plan.

The PCO has been tested on a library of plans created with the help of healthcare professionals from the Nursebot team. The PCO runs on a Pentium III 933 MH processor with 262 MB of RAM and the time to generate and maintain small plans has been insignificant. Due to a limited plan library, testing on the speed of the PCO for large domains has not yet been done. A near-term goal is to conduct extensive interview with residents and caregivers at a retirement community. This will not only aid in evaluating our quality metrics, it will also provide additional knowledge acquisition.

Future work on the PCO will include additions to the current corpus of rewrite rules. In particular, the inclusion of rules that will combine activities that share common resources must be completed. Activity priorities need to be implemented in the system as well. Finally, the PCO does not currently exploit the full flexibility of the plans encoded in the PM. Where the PM handles disjunctive constraints, the PCO bases the reminders only on the STP chosen as the current daily plan. Allowing disjunctions would require a new initial plan generation scheme. We are investigating the user of work on flexible dispatch of DTPs (Tsamardinos, Pollack, & Ganchev 2001) amongst other solutions to address this problem.

Acknowledgments

This research was supported by the Air Force Office of Scientific Research (F49620-01-1-0066), the National Science Foundation (115-0085796), and a fellowship from the National Physical Sciences Consortium. The views and conclusions herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR or the U.S. Government.

References

Ambite, J. L., and Knoblock, C. 2001. Planning by rewriting. *Journal of Artificial Intelligence Research (JAIR)* 15:207–261.

Bacchus, F. Fall, 2001. AIPS'00 Planning Competition: The Fifth International Conference on Artificial Intelligence Planning and Scheduling Systems. *AI Magazine* 22(3).

Bonasso, R. P.; Kortenkamp, D.; Miller, D.; and Slack, M. 1997. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence* 9(1).

Cannon, D. S., and Allen, S. N. 2000. A comparisons of the effects of computer and manual reminders on compliance with a mental health clinical practice guideline. *JAMIA* 7:196–203.

Census. 1997. Aging in the United States: Past, present, and future. Technical report, National Institute on Aging and United States Bureau of the Census. Available at http://www.census.gov/ipc/prod/97agewc.pdf.

Colbry, D.; Peitner, B.; and Pollack, M. 2001. Execution monitoring using quantitative temporal dynamic bayes nets. Submitted.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

IPAT. 1999. Solutions: Assistive technology for people with hidden disabilities. Available at www.ndipat.org.

Kirsch, N., and Levine, S. P. June 1988. Improving functional performance with computerized task guidance systems. In *Proceedings Compe Rendu Internatinoal Conference of the Association for the Advancement of Rehabilitation Technology ICAART 88*, 564–56.

Kirsch, N.; Levine, S. P.; Fallon-Krueger, M.; and Jaros, L. A. 1987. The micorcomputer as an 'orthotic' device for pateints with cognitive deficits. *Journal of Head Trauma Rehabilitation* 2(4):77–86.

Levinson, R. 1995. A general programming language for unified planning and control. *Artificial Intelligence.* 76.

Levinson, R. 1997. PEAT – The Planning and Execution Assistant and Trainer. *Journal of Head Trauma Rehabilitation*.

Muscettola, N.; Morris, P.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Proceedings of the 6th Conference on Principles of Knowledge Representation and Reasoning*.

Nursebot. 2000. Nursebot project: Robotic assistants for the elderly. Available at http://www.cs.cmu.edu/ nursebot/.

Oddi, A., and Cesta, A. 2000. Incremental forward checking for the disjunctive temporal problem. In *European Conference on Artificial Intelligence*.

Parente, R. 1991. Personal communication.

Pell, B.; Gat, E.; Keesing, R.; Muscettola, N.; and Smith, B. 1996. Plan execution for autonomous spacecraft. In *AAAI Fall Symposium Series: Plan Execution: Problems and Issues*, 109–116.

Pollack, M. E., and Horty, J. F. 1999. There's more to life than making plans: Plan management in dynamic environments. *AI Magazine* 20(4):71–84.

Pollack, M. E.; McCarthy, C. E.; Tsamardinos, I.; and *et al.* 2001. Autominder: A planning, monitoring, and reminding assistive agent. Submitted for publication.

Rivlin, A. M., and Wiener, J. M. 1988. Caring for the disabled elderly: Who will pay?

Stergiou, K., and Koubarakis, M. 1998. Backtracking algorithms for disjunctions of temporal constraints. In *15th National Conference on Artificial Intelligence (AAAI)*.

Tsamardinos, I.; Pollack, M. E.; and Ganchev, P. 2001. Flexible dispatch of disjunctive plans. In *To appear in the 6th European Conference on Planning*.

Tsamardinos, I. 2001. *Constraint-Based Temporal Reasoning Algorithms, with Applications to Planning*. Ph.D. Dissertation, University of Pittsburgh, Pittsburgh, PA.