

Execution-Time Plan Management for a Cognitive Orthotic System

Martha E. Pollack¹, Colleen E. McCarthy², Sailesh Ramakrishnan¹, and Ioannis Tsamardinos³

¹ Artificial Intelligence Laboratory, Dept. of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, U.S.A

`pollackm,sailesh@umich.edu`,

`http://www.eecs.umich.edu/pollackm/`

² Department of Computer Science, University of Pittsburgh, U.S.A

`colleen@cs.pitt.edu`

³ Department of Biomedical Informatics, Vanderbilt University, U.S.A

`ioannis.tsamardinos@vanderbilt.edu`

Abstract. In this paper we discuss our work on plan management in the Autominder cognitive orthotic system. Autominder is being designed as part of an initiative on the development of robotic assistants for the elderly. Autominder stores and updates user plans, tracks their execution via input from robot sensors, and provides carefully chosen and timed reminders of the activities to be performed. It will eventually also learn the typical behavior of the user with regard to the execution of these plans. A central component of Autominder is its Plan Manager (PM), which is responsible for the temporal reasoning involved in updating plans and tracking their execution. The PM models plan update problems as disjunctive temporal problems (DTPs) and uses the Epilitis DTP-solving system to handle them. We describe the plan representations and algorithms used by the Plan Manager, and briefly discuss its connections with the rest of the system.

1 Introduction

In this paper we discuss our work on plan management in the Autominder cognitive orthotic system. Autominder is being designed as part of the Initiative on Personal Robotic Assistants for the Elderly (Nursebot)[15], a multi-university collaborative project aimed at investigations of robotic technology for the elderly. The initial focus of this initiative is the design of an autonomous robot, currently called Pearl, that will “live” in the home of an elderly person. Autominder stores and updates plans representing the activities that the elderly client is expected to perform, tracks their execution via sensor input from the robot, learns the typical behavior of the client with regard to the execution of these plans, and provides carefully chosen and timed reminders for the activities to be performed.⁴

⁴ The learning component is not yet implemented.

A central component of Autominder, is its Plan Manager (PM), which is responsible for the temporal reasoning involved in updating plans and tracking their execution. The PM must be able to reason about complex temporal constraints. For example, the client’s plan may specify constraints on when she should eat meals, e.g., that they should be spaced four hours apart. Thus, when the client wakes up and eats her first meal, Autominder must propagate this information to update the times for lunch and dinner. Adjustments may also need to be made to the timing constraints on other activities, such as taking medicine. Notice that the adjustments in the plan may be rather complex: for instance, if the new scheduled time for lunch conflicts with a recreational activity, such as a visit to a neighbor, it may be necessary to reschedule the visit.

Autominder’s PM extends our earlier work in which we developed a prototype plan manager for an intelligent personal calendaring tool, called PMA (the Plan Management Agent) [20]. The goal of PMA is to allow a user to manage a complex schedule of tasks by checking whether new tasks conflict with existing ones, suggesting ways of resolving conflicts that are detected, and providing reminders to the user at the time at which an activity must be executed. The PM in Autominder extends PMA in two ways:

- It allows more expressive plan representations, notably supporting arbitrary disjunctive temporal constraints; and
- It provides efficient algorithms for handling plans with this level of expressiveness. In fact, the algorithms have been shown to be two orders of magnitude faster on a range of benchmark problems [24].

Additionally, we have added an execution monitor to Autominder that was not present in PMA, and we have separated out the reminding task into a distinct module that performs more detailed reasoning about the appropriateness of alternative reminders.

This paper focuses on Autominder’s plan manager. In the next section, we describe our plan representation, showing how it supports richer set of temporal constraints between activities than most alternative representations. Section 3 describes the algorithms used by the plan manager to perform plan update at execution time, while Section 4 discusses the underlying reasoning system, Epilitis. Section 5 relates the plan manager to the larger Autominder system and Section 6 briefly describes the robot platform (Nursebot) that Autominder is designed to run on. Finally, the remainder of the paper discusses related and future work.

2 Plan Representation

The PM uses a library of plans that represent the structure of activities that the client typically performs. In our current Autominder domain these activities include taking medicine correctly, eating, drinking water, toileting, taking care of personal hygiene, performing physical exercises (e.g., “Kegel” bladder exercises),

performing self-examinations (e.g., foot exams by diabetics), engaging in recreational activities (e.g., watching television, attending a Bingo game), and going to doctors’ and dentists’ appointments. Many of these activities are modeled as being decomposable into more primitive activities. For instance, a doctor’s appointment consists of making the appointment beforehand, arranging transportation, confirming the visit before leaving, going to the doctor’s office, and scheduling a follow-up visit. Essentially, each high-level action corresponds to a complete partial-order plan with steps, causal links, and temporal constraints; however, as we describe below, the set of constraints we allow is much richer than that of most partial order planners.

At initialization the caregiver supplies Autominder with a typical daily plan of activities. Information need only be provided about “top-level” activities, e.g., a doctor’s appointment; the entire plan for that activity, including all the steps (e.g., arranging transportation for the appointment) and constraints, is then loaded from the plan library and inserted into the client’s overall daily plan. Two points should be made about plan initialization. First, while the plan library may provide default temporal constraints, the user has complete control over these, and can specify start times and durations for all activities. Second, additional activities can later be added, either as a one-time event (e.g., a visit from a relative) or a recurrent activity (e.g., attending a weekly Bingo game). The plan developed at initialization acts as a template for daily activities, not an absolute schedule.

The plan manager supports a rich set of temporal constraints: for example, we can express that the client should take a medication within 15 minutes of waking, and then eat breakfast between 1 and 2 hours later. Importantly, as indicated above, the time constraints can be flexible: fixed, rigid times do *not* need to be assigned to each activity. Instead, the plan may specify that an activity must be performed without specifying the exact time at which it should occur, or that a particular goal must be achieved without specifying what plan the client should use to achieve that goal. Figure 1 shows the types of temporal constraints that can be specified.

- | | |
|---|-------------------------------------------------|
| 1 | Earliest Start Time |
| 2 | Latest Start Time |
| 3 | Earliest End Time |
| 4 | Latest End Time |
| 5 | Minimum Duration |
| 6 | Maximum Duration |
| 7 | Minimum Period of Separation between Activities |
| 8 | Maximum Period of Separation between Activities |

Fig. 1. Temporal Constraints for an Activity

To achieve this flexibility, we model client plans as Disjunctive Temporal Problems (DTP) [17, 22, 24]. A DTP is an expressive framework for temporal rea-

soning problems that extends the well-known Simple Temporal Problem (STP) [6] by allowing disjunctions, and the Temporal Constraint Satisfaction Problem (TCSP) [*ibid.*] by removing restrictions on the allowable disjunctions. DTPs are represented as a set of variables and a set of disjunctive constraints between members of this set. Formally, a DTP is defined to be a pair $\langle V, C \rangle$, where

- V is a set of variables (or nodes) whose domains are the real numbers, and
- C is a set of disjunctive constraints of the form:
 $C_i : x_1 - y_1 \leq b_1 \vee \dots \vee x_n - y_n \leq b_n$, such that x_i and y_i are both members of V , and b_i is a real number.⁵

In the PM, we assign a pair of DTP variables to each activity in the client’s plan: one variable represents the start time of the activity, while the other represents its end time. We can easily encode a variety of typical planning constraints, including absolute times of events, relative times of events, and event durations, and can also express ranges for each of these. Figure 2 shows some of these relations between two activities, s_i and s_j . A simple example is shown in Figure 3 where we represent information about two activities, toileting and watching TV. Note that to express a clock-time constraint, e.g., TV watching beginning at 8:00am, we use a *temporal reference point* (TR), a distinguished value representing some fixed clock time. In Autominder the TR corresponds to midnight.

To express:	Use:
s_i precedes s_j	$\text{end}(s_i) - \text{start}(s_j) \leq 0$
s_i begins at 9am, Monday	$\text{ref} - \text{start}(s_i) \leq -9 \wedge \text{start}(s_i) - \text{ref} \leq 9$ (assuming ref is 12am on Monday)
s_i lasts between 2 and 3 hours	$\text{end}(s_i) - \text{start}(s_i) \leq 3 \wedge \text{start}(s_i) - \text{end}(s_i) \leq -2$
s_i occurs more than 48 hours before s_j	$\text{end}(s_i) - \text{start}(s_j) \leq -48$

Fig. 2. Temporal constraint representations

3 Plan Updates

The primary job of the Plan Manager is to maintain an up-to-date model of the plan activities that the user should execute. Updates occur in response to four types of events:

The addition of a new activity to the plan. During the course of the day, the client and/or his or her caregivers may want to make additions to the plan: for instance, to attend a Bingo game or a newly scheduled doctor’s

⁵ As is customary in the literature, in this paper we will assume without loss of generality that b_i is an integer.

“Toileting should begin between 11:00 and 11:15.”
$660 \leq Toileting_S - TR \leq 675$
“Toileting takes between 1 and 3 minutes.”
$1 \leq Toileting_E - Toileting_S \leq 3$
“Watching the TV news can begin at 8:00 or 11:00.”
$480 \leq WatchNews_S - TR \leq 482 \vee$
$660 \leq WatchNews_S - TR \leq 662$
“The news takes exactly 30 minutes.”
$30 \leq WatchNews_E - WatchNews_S \leq 30$
“Toileting and watching the news cannot overlap.”
$0 \leq WatchNews_S - Toileting_E \leq \infty \vee$
$0 \leq Toileting_S - WatchNews_E \leq \infty$

Fig. 3. Examples of the use of DTP Constraints

appointment. At this point, plan merging must be performed to ensure that the constraints on the new activity are consistent with the other constraints in the existing plan.

The modification or deletion of an activity in the plan. Plan merging must also occur in the case of activity modification or deletion. Note that the PM will add or tighten constraints if needed, but will not “roll back” (i.e., weaken) any constraints. For instance, if the bounds on eating lunch are constrained to allow for an appointment, but the appointment is later retracted, the bounds on lunch will not be changed to allow for more time. More sophisticated plan retraction is an area of future research.

The execution of an activity in the plan. It is important for the PM to respond to the execution of activities in the plan. Information about activity execution is provided by another component of Autominder, the Client Modeler (CM). The CM is tasked with monitoring plan execution. It receives reports of the robot’s sensor readings, for instance when the client moves from one room to another, and uses that to infer the probability that particular steps in the client plan have been executed; it can also issue questions to the client for confirmation about whether a step has been executed. When the CM believes with probability exceeding some threshold that a given step has begun or ended, it passes this information on to the PM. The PM can then update the client plan accordingly.

The passage of a time boundary in the plan. Finally, just as the execution of a plan step may necessitate plan update, so may the non-execution of a plan step. Consider our example in Figure 3, where the client wants to watch the news on television each day, either from 8:00-8:30 a.m. or from 11:00-11:30 p.m. At 8:00am (or a few minutes after), if the client has not begun watching the news, then the PM should update the plan to ensure that the 11:00-11:30 slot is reserved for that purpose.

To perform plan update in each of these cases, the PM formulates and solves a disjunctive temporal problem (DTP). Detailed explanations of the updates

performed in each of these four cases can be found in [19]. Here we just sketch the approach taken for the first case, when a new activity is added to the plan. In that case, the PM performs *plan merging*, a process that can be decomposed into the following steps:

1. creating DTP encodings of the existing plan and the new activity;
2. identifying potential conflicts introduced by the plan change;
3. creating a DTP that includes all the possible alternative resolutions of the conflicts; and finally
4. attempting to solve a DTP that combines the encodings in (1) and in (3).

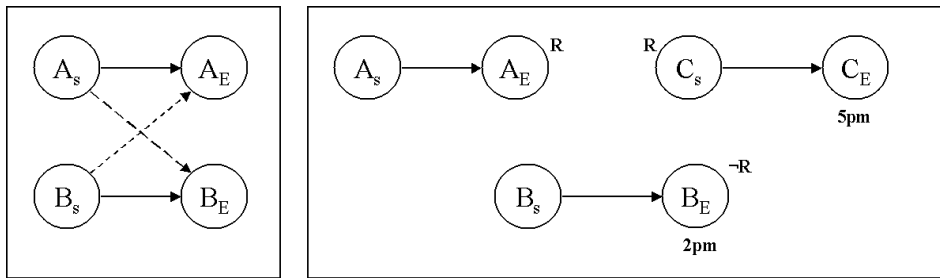


Fig. 4. Resolving temporal conflicts

Because we are using the DTP representation, we can model the traditional conflict resolution techniques (promotion and demotion) with a single constraint, as illustrated in Figure 4a: either A must be before B or B must be before A . But the use of DTPs also allows us to handle quantitative temporal constraints. In Figure 4b we illustrate an action A that achieves condition R for action C ; we also include another action B that threatens the causal link from A to C . The DTP constraint that captures the alternative resolutions of this threat is $B_E - A_S \leq 0 \vee C_E - B_S \leq 0$. However, suppose further that action B must end by 2pm and action C by 5pm. It is then clear that only the first disjunct of the threat resolution constraint is satisfiable, and this is precisely what a DTP-solver would find. While this is a very simple example, in general the PM handles large sets of complex disjunctive constraints.

Clearly, solving DTPs is a central part of what the PM does. In the PM, we use the using the Epilitis DTP solving system developed in our group [24, 25]. In the next section, we briefly describe DTP solving in general, and Epilitis in particular.

4 Solving DTPs

As stated earlier, a DTP is an extension of a Simple Temporal Problem (STP) [6]. Essentially, an STP is a DTP in which constraints must be simple inequalities

without any disjunctions; i.e., each constraint in C takes the form $x - y \leq b_{xy}$. This constraint represents the fact that y must occur no more than b_{xy} time units after x . Because an STP contains only binary constraints, it can be represented with a weighted graph called a Simple Temporal Network (STN), in which an edge (x, y) with weight b exists between two nodes iff there is a constraint $(y - x \leq b_{xy}) \in C$. Polynomial time algorithms can be used to compute the all-pairs shortest path matrix, or *distance array* of the STN. A path p from node x to node y imposes the following (induced) constraint: $y - x \leq \sum_{i=0}^n b_{p_i p_{i+1}}$ where $x = p_0$, $y = p_{n+1}$ and the rest p_i are the nodes on the path p . The tightest (induced or explicit) constraint between any nodes x and y is therefore given by the shortest path connecting them, denoted by d_{xy} , and called the *distance* between x and y . The *distance array* for a particular STN is its all-pairs shortest-path matrix. An STN is consistent if and only if for every node x , $d_{xx} \geq 0$, which means that there are no negative cycles, something that can be computed in polynomial time. Finding a solution to an STN is also a polynomial time computation.

A DTP can be viewed as encoding a collection of alternative STPs. To see this, recall that each constraint in a DTP is a disjunction of one or more STP-style inequalities. Let C_{ij} be the j^{th} disjunct of the i^{th} constraint of the DTP. If we select one disjunct C_{ij} from each constraint C_i , the set of selected disjuncts forms an STP, which we will call a *component STP* of a given DTP. It is easy to see that a DTP D is consistent if and only if it contains at least one consistent component STP. Moreover, any solution to a consistent component STP of D is also a solution to D itself. Because only polynomial time is required both to check the consistency of an STP, and, if consistent, extract a solution to it, in the remainder of this paper we will say that the solution of a given DTP is any consistent component STP of it.

The computational complexity in DTP solving derives from the fact that there are exponentially many sets of selected disjuncts that may need to be considered; the challenge is to find ways to efficiently explore the space of disjunct combinations. This has been done by casting the disjunct selection problem as a constraint satisfaction processing (CSP) problem [22, 17] or a satisfiability (SAT) problem [1]. Because the original DTP is itself a CSP problem, we will refer to the component-STP extraction problem as the *meta-CSP* problem. The meta-CSP contains one variable for each constraint C_i in the DTP. The domain of C_i is the set of disjuncts in the original constraint C_i . The constraints in the meta-CSP are not given explicitly, but must be inferred: an assignment satisfies the meta-CSP constraints iff the assignment corresponds to a consistent component STP. For instance, if the variable C_i is assigned the value $x - y \leq 5$, it would be inconsistent to extend that assignment so that some other variable C_j is assigned the value $y - x \leq -6$.

In Autominder, we use the Epilitis DTP solver developed in our research group [24, 25]. Like prior DTP solvers [17, 21, 1], Epilitis does not attempt to solve the DTP directly by searching for an assignment of integers to the time points. Instead, it solves a meta-CSP problem: it attempts to find one disjunct from each disjunctive constraint such that the set of all selected disjuncts forms a

consistent STP. Epilitis can return an entire STP, which provides interval rather than exact constraints on the time points in the plan. Consider an example of a plan that involves taking medicine between 14:00 and 15:00, which is amended with a plan to leave for a bridge game at 14:30. Epilitis will return a DTP that constrains the medicine to be taken sometime between 14:00 and 14:30; it does not have to assign a specific time (e.g., 14:10) to that action.

In general, there may be multiple solutions to a DTP, i.e., multiple consistent STPs that can be extracted from the DTP. In the current version of Autominder, the PM arbitrarily selects one of these (the first one it finds). If subsequent execution is not consistent with the STP selected, then the DTP will attempt to find an alternative consistent solution. A more principled approach would select solutions in an order that provides the greatest execution flexibility. For example, the solution that involves watching the 8:00 a.m. news leaves open the possibility of instead watching the news at 11:00 a.m. However, if the first solution found instead involved watching the later news show, then after an execution failure there would be no way to recover, as it would be too late to watch the 8:00 a.m. news. Unfortunately selecting DTP solutions to maximize flexibility is a difficult problem [26].

Epilitis combines and extends previous approaches in DTP solving, in particular by adding no-good learning. As a result, it achieves a speed-up of two orders of magnitude on a range of benchmark problems[24]. For our current Autominder scenarios, which typically involve about 30 actions, Epilitis nearly always produces solutions in less than one second, a time that is well within the bounds we require. Details of the Epilitis system can be found in [24, 25]).

5 Autominder

In the Autominder architecture (depicted in Figure 5), other important components rely on as well as support the PM, notably, a Client Modeler (CM) and a Personal Cognitive Orthotic (PCO). We briefly describe each of these in turn.

The Client Modeler is responsible for two tasks: (i) inferring what planned activities the client has performed, given sensor data; and (ii) learning a model of the client's expected behavior. These tasks are synergistic, in that the client model developed is used in the inference task, while the results of the inference are used to update the model.

The client's expected behavior is represented with a new formalism for temporal reasoning under uncertainty, Quantitative Temporal Dynamic Bayes Nets (QTDBNs) [4]. QTDBNs combine a simplified form of time nets with an augmented form of Dynamic Bayes nets (DBN), thereby supporting reasoning about changing conditions (fluents) in settings where there are quantitative constraints amongst them. For the purposes of the current paper, it is sufficient to consider the Dynamic Bayes net component, which represents all the client's activities for one day. The nodes in each time slice of the DBN are random variables representing (i) the incoming sensor data (e.g., client has moved to kitchen); (ii) the execution of planned activities (e.g., client has started breakfast); and (iii)

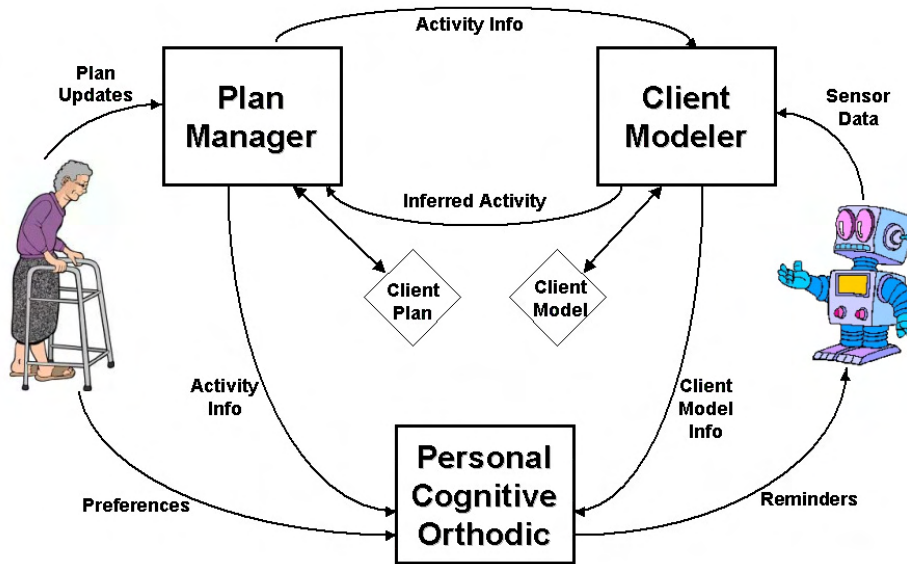


Fig. 5. The Autominder Architecture

whether a reminder for each activity has already been issued. Initially, the model is derived from the client plan by making two assumptions: first, that all activities in the plan will be executed by the client, without reminders, within the time range specified in the plan, and second, that the actual time of an activity can be described by a uniform probability density function over the range associated with that activity. The CM uses sensor data and the current time to update the model. Each time sensor data arrives, the CM performs Bayesian update. If a node representing the execution of some activity execution node's probability rises above a threshold, the activity is believed to have occurred, and the CM notifies the PM of the executed activity.

The Personalized Cognitive Orthotic (PCO) [11] uses information from the Plan Manager and Client Modeler to decide what reminders to issue and when. The PCO identifies those activities that may require reminders based on their importance and their likelihood of being executed on time as determined by the CM. It also determines the most effective times to issue each required reminder, taking account of the expected client behavior and any preferences explicitly provided by the client and the caregiver. Finally, the PCO is also designed to enable the generation of justifications for reminders. In generating a justification for a reminder, the PCO can make use of the underlying client plan, the current reminder plan, and the preferences of the caregiver and the client.

The client and caregiver communicate with Autominder via a graphical user interface (see Figure 6). On the right of the screen the client can view her daily

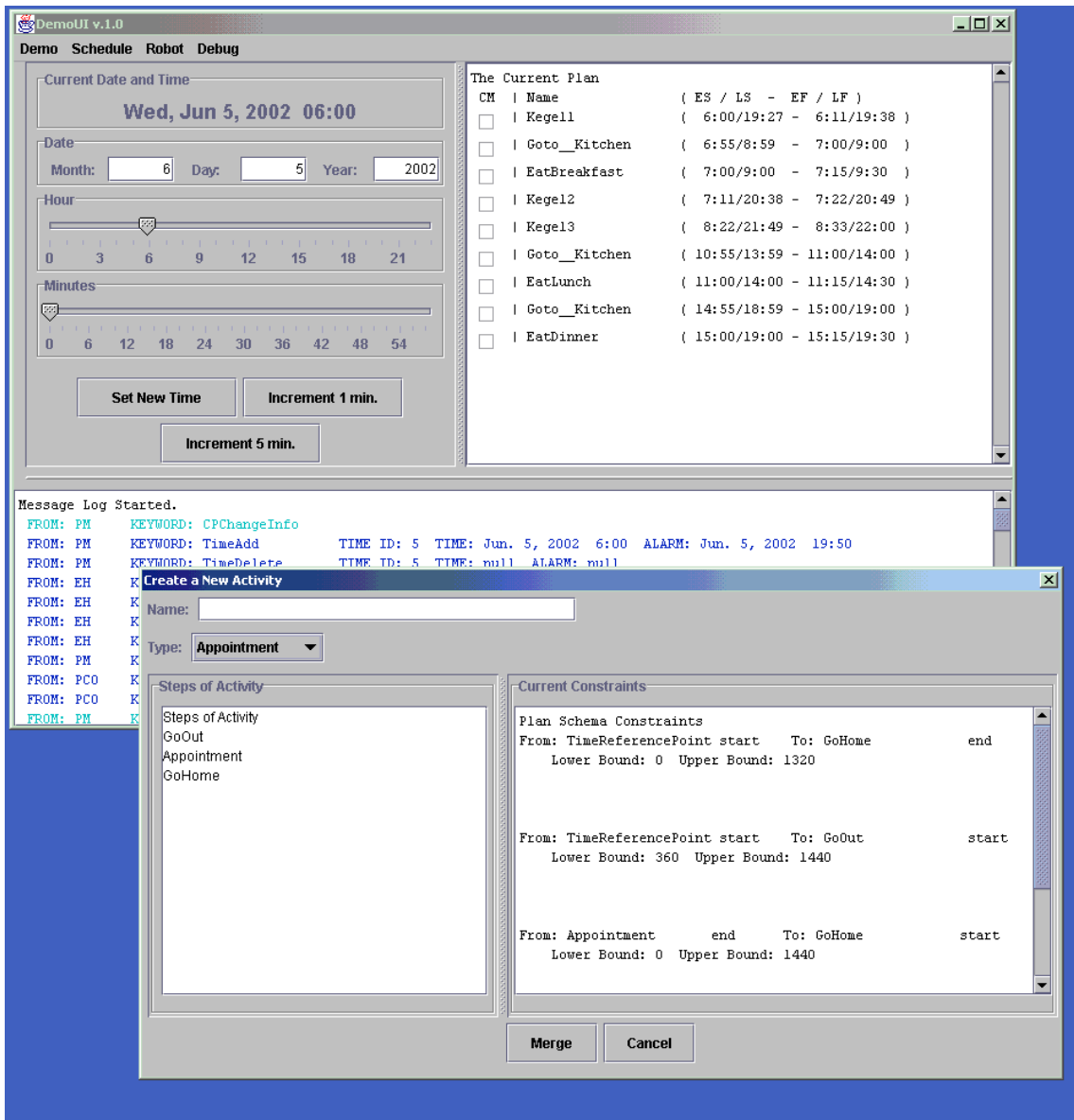


Fig. 6. Autominder: Adding new activities to the daily plan

activities, including the earliest and latest start times of each activity. Check-boxes allow the client to directly communicate to the system that he or she has completed an activity, rather than depending on sensor data to identify and confirm this information. To add new activities and constraints at run time, the client can use a drop-down menus. In this scenario, the caregiver is adding a doctor's appointment for 8 a.m. (see lower screen).

A prototype version of Autominder has been fully implemented in Java and Lisp. It includes all of the components and most of the features described above; however, the client model does not yet learn client behavior over time, and the PCO does not yet handle preferences nor issue justifications for reminders. An earlier version of the system was installed on the robot platform (Pearl), and underwent field testing in June, 2001; we are currently in the process of integrating our newer version on Pearl.

6 Nursebot

The Autominder cognitive orthotic is being developed as part of the Initiative on Personal Robotic Assistants for the Elderly, a multi-university, multi-disciplinary research effort conceived in 1998.⁶ The initial focus of the Initiative is to design an autonomous mobile robot that can "live" in the home of an older individual, and provide him or her with reminders about daily plans. To date, two prototype robots have been designed and built by members of the initiative at Carnegie Mellon. Pearl, the more recent of these robots, is depicted in Figure 7. Pearl is built on a Nomadic Technologies Scout II robot, with a custom-designed and manufactured "head", and includes a differential drive system, two on-board Pentium PCs, wireless Ethernet, SICK laser range finders, sonar sensors, microphones for speech recognition, speakers for speech synthesis, touch-sensitive graphical displays, and stereo camera systems [2, 23, 18]. Members of the Initiative also have interests both in other ways in which mobile robots can assist older people (e.g., telemedicine, data collection and surveillance, and physically guiding people through their environments).

There are numerous reasons for embedding the Autominder system on a robotic platform in the Nursebot project. First of all, although handheld systems are popular, they are more likely to be lost or forgotten. Second, the cost of developing a robotic assistant is less expensive than retrofitting a home. At the same time, mobility is not lost since Pearl can navigate through the home. Finally, the sensor capabilities of the robot allow for the monitoring of the client's action, thus providing integral information to the action inference engine.

7 Related Work

The large literature on workflow systems [9, 10, 16] is relevant to Autominder since both systems are designed to guarantee that structured tasks are per-

⁶ In addition to the University of Michigan, the initiative includes researchers at the University of Pittsburgh and Carnegie Mellon University.



Fig. 7. Pearl: A Mobile Robot Platform for the Autominder Cognitive Orthotic. Photo courtesy of Carnegie Mellon University.

formed by humans in a timely manner. The emphasis in workflow systems, however, tends to be quite different from that in Autominder. Much of the research on workflow systems focuses on the transfer of information between people in an organization who are jointly responsible for carrying out the tasks, and on the transformation of that information to enable its processing by different computer systems (interoperability). Workflow systems do not typically provide capabilities for action inference, learning of behavior patterns, or sophisticated reasoning about reminders. However, recent efforts to integrate AI planning technology with workflow tasks [7, 3] may lead to results that can be integrated into Autominder.

The literature on cognitive orthotics for persons with cognitive deficits is relevant to Autominder, but not described in detail here; see [5] for an overview. Other work in plan management in the medical domain is being done by Miksch et al. [12, 8]. In their work they are focusing on using planning techniques to respond to the practical demands of planning to achieve clinical guidelines. They use the Asbru representation language to check the temporal, clinical, and hierarchical consistency of plans. While our application domain that of clients with mild memory impairment, their work is geared towards assisting the caregivers.

Finally, we have already pointed to previous literature on DTPs [17, 22, 24]. The Autominder system also builds on prior work on the dispatch of disjunctive plans [14, 27, 26]. This work has primarily been focused on plans that are represented as STPs.

8 Conclusion

Execution-time plan management is essential to many systems in dynamic environments. We have described our work on plan management in the context of Autominder, a planning and reminding system designed to aid elderly persons with memory impairment. Autominder's PM builds on our earlier work on plan

management, providing the tools to store and update plans representing the activities that the client is expected to perform. However, Autominder extends the earlier work by providing significantly increased representational power, along with highly efficient algorithms for handling expressive plans.

There are a number of areas of future work that we are pursuing. One of the most important is extending the PM to handle uncontrollable events[13], activities that are not under the client's direct control, such as the time at which deliveries are made. Another extension would support optional, prioritized activities in plans. We are also developing better capabilities for plan revision. In dynamic environments, plans may need to be revised in several circumstances: (1) when a new goal to be adopted conflicts with existing plans; (2) when the environment changes in ways that invalidate the existing plans; and (3) when the client's desires and/or preferences change in ways that affect the existing plans. We are developing plan revision algorithms for plans that include expressive temporal constraints of the kind handled by our PM; our algorithms aim to make the minimal modifications required to ensure plan correctness in the face of the kinds of changes just listed.

References

1. E. Giunchiglia A. Armando, C. Castellini. Sat-based procedures for temporal reasoning. In *5th European Conference on Planning*, 1999.
2. G. Baltus, D. Fox, F. Gemperle, J. Goetz, T. Hirsch, D. Margaritis, M. Montermelo, J. Pineau, N. Roy, J. Schulte, and S. Thrun. Towards personal service robots for the elderly. In *Workshop on Interactive Robots and Entertainment*, 2000.
3. Pauline M. Berry and Karen L. Myers. Adaptive process management: An AI perspective. In *Proceedings of the Workshop Towards Adaptive Workflow System*, 1998. Available at <http://www.ai.sri.com/berry/publications/cscw98-sri.html>.
4. Dirk Colbry, Barth Peintner, and Martha E. Pollack. Execution monitoring with quantitative temporal dynamic bayesian networks. In *Proceedings of the Sixth International Conference on AI Planning Systems (AIPS)*, Toulouse, France, 2002.
5. Elliot Cole. Cognitive prosthetics: an overview to a method of treatment. *NeuroRehabilitation*, 12:39–51, 1999.
6. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
7. B. Drabble, T. Lydiard, and A. Tate. Workflow support in the air campaign planning process. In *Proceedings of the Workshop on Interactive and Collaborative Planning, AIPS98*, Pittsburgh, PA, 1998.
8. G. Duftschmid, S. Miksch, and W. Gall. Verification of temporal scheduling constraints in clinical practice guidelines. In *To appear in Artificial Intelligence in Medicine, 2002*, 2002.
9. Dimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.
10. Dirk Mahling, Noel Craven, and W. Bruce Croft. From office automation to intelligent workflow systems. *IEEE Expert*, 10(3), 1995.
11. Colleen E. McCarthy and Martha E. Pollack. A plan-based personalized cognitive orthotic. In *Proceedings of the Sixth International Conference on AI Planning Systems (AIPS)*, Toulouse, France, 2002.

12. Silvia Miksch. Plan management in the medical domain. *AI Communications*, 4, 1999.
13. Paul Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In *International Joint Conference on Artificial Intelligence-2001*, pages 494 – 502, 2001.
14. Nicola Muscettola, Paul Morris, and Ioannis Tsamardinos. Reformulating temporal plans for efficient execution. In *Proceedings of the 6th Conference on Principles of Knowledge Representation and Reasoning*, 1998.
15. Nursebot. Nursebot project: Robotic assistants for the elderly, 2000. Available at <http://www.cs.cmu.edu/nursebot/>.
16. Gary J. Nutt. The evolution towards flexible workflow systems. *Distributed Systems Engineering*, pages 276–294, 1996.
17. Angelo Oddi and Amedeo Cesta. Incremental forward checking for the disjunctive temporal problem. In *European Conference on Artificial Intelligence*, 2000.
18. J. Pineau and S. Thrun. High-level robot behavior control using POMDPs. In *AAAI-02 Workshop on Cognitive Robotics*, 2002.
19. Martha E. Pollack. Planning technology for intelligent cognitive orthotics. In *Proceedings of the Sixth International Conference on AI Planning Systems (AIPS)*, Toulouse, France, 2002.
20. Martha E. Pollack and John F. Horty. There’s more to life than making plans: Plan management in dynamic environments. *AI Magazine*, 20(4):71–84, 1999.
21. K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120:81–117, 2000.
22. Kostas Stergiou and Manolis Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. In *15th National Conference on Artificial Intelligence (AAAI)*, 1998.
23. S. Thrun, J. Langford, and V. Verma. Risk sensitive particle filters. *Advances in Neural Information Processing Systems*, 14, 2002.
24. Ioannis Tsamardinos. *Constraint-Based Temporal Reasoning Algorithms, with Applications to Planning*. PhD thesis, University of Pittsburgh, Pittsburgh, PA, 2001.
25. Ioannis Tsamardinos and Martha E. Pollack. Efficient solution techniques for disjunctive temporal problems. Under review. Available from the authors upon request., 2002.
26. Ioannis Tsamardinos, Martha E. Pollack, and Philip Ganchev. Flexible dispatch of disjunctive plans. In *To appear in the 6th European Conference on Planning*, 2001.
27. R J Wallace and E C Freuder. Dispatchable execution of schedules involving consumable resources. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling*, 2000.