Continual Planning with Time-Oriented, Skeletal Plans

Silvia Miksch[1] and **Andreas Seyfang**[1]

Abstract.

In dynamically changing environments a planning system does not have all the required information at the first place and the world state can change, rendering the original plan invalid. Consequently, taking planning to complex, real-world domains calls for close coupling of planning and plan execution, known as continual planning.

In domains where a high degree of adaptability is crucial and little a priori knowledge is available, explorative approaches are suitable. In other domains where comprehensive knowledge is available and human life depends on plan execution, like in medical treatment planning, only approved and validated procedures are admitted. Even more, these procedures constitute a rich asset in the planning task. To combine the utilization of such knowledge with the flexibility necessary in real-world applications, the Asgaard system integrates time-oriented, skeletal planning with real-time monitoring. It features a monolithic framework for the creation, verification, execution, and critiquing of plans with extensive covering of temporal modeling and data abstraction. In this paper, we describe the monitoring and plan adaptation capabilities of Asgaard.

1 INTRODUCTION

Carrying planning into the real-world, one is confronted with many issues not covered by classical planning [13, 15]:

- Large, structured, but incomplete and partially informal domain knowledge must be utilized in the planning process.
- Parts of the environment are beyond the influence of the plan executing agent.
- Actions, goals, and observations have a temporal dimension. Their durations may only partially be known in advance.
- For many cases of uncertainty, the probability of occurrence itself is unknown.
- Plans cannot be undone easily, once they are started. Instead, a traversal plan must be inserted, if a plan should be replaced by another one.

One of the basic approaches to cope with this host of challenges is the introduction of a tightly coupled control loop between the generation and the execution of plans [15], called *continual planning* [5]. This control loop is constituted by the tasks of *environment monitoring*, *plan adaptation*, and *replanning*, as detailed in the remainder of this section.

- **Environment Monitoring.** When executing plans in a real-world environment, monitoring the actual state of the world immediately becomes an issue, as it turns out that there is no perfect model of reality. This implies that there are relations between quantities without defined numerical dependences. To overcome these limitations, a particular environment monitoring is needed which is able to observe the world states with knowledge-rich methods [17].
- **Plan Adaptation.** In all fields of application there are some incidents which are not predictable but whose possible occurrence is well known. In technical domains a familiar example is the failure of a unit. In the medical domain the complications (e.g., caused by infections) are too manifold to be modeled in detail in advance. Still a set of classes of exceptions can be found and for each class some special plan can be devised to implement safe failure or to start some action to undo the effects of the disturbance.
  So under such a scheme of operation, the original plan is not immediately failing if problems arise, but suspended. Meanwhile, another plan is executed to cope with the problem. Of course, there must be some meta-monitoring performed on the plan execution to detect cases, in which the problems are not fixed within a certain time. E.g., if a robot encounters another obstacle while bypassing the first one, it is time to consider taking another route.
- **Replanning.** In cases where prepared alternatives are not enough to cope with unexpected changes in the environment, new plans must be worked out. Compared to planning from scratch, the process of replanning is even more complicated by the fact that currently pursued plans, which are independent from those which failed, stay on the list of agenda. Thus, all decisions must be made in the context of the current situation. Additional plans must be merged with those the agent has already committed to [12].

In section 2 we given an introduction to the Asgaard framework. In section 3 we describe the modules of the Asgaard system concerned with runtime issues: the data-abstraction, the monitor and the execution module. In section 4 we compare Asgaard to related work in the field.

2 THE ASGAARD FRAMEWORK

The Asgaard framework [22] outlines task-specific problem-solving methods to support both design and execution of skeletal plans. This project tries to build a bridge between the planning approaches and the medical approaches, addressing the demands of the medical staff on the one side and applying rich plan management on the other side. For the representation of plans, a time-oriented, intention-based, skeletal plan-representation language, called Asbru, was developed [18].

2.1 Time-Oriented, Skeletal Plans

A common strategy for the representation and the reuse of domain-specific procedural knowledge is the representation of that knowl-

[1] Institute of Software Technology, Vienna University of Technology Favoritenstraße 9-11/188, A-1040 Vienna, Austria, email: {silvia, seyfang}@ifs.tuwien.ac.at

edge as a library of skeletal plans. Skeletal plans are plan schemata at various levels of detail which capture the essence of the procedure, but leave room for execution-time flexibility in the achievement of particular goals. Thus, they are usually reusable in various contexts. The idea was originally proposed by Friedland [8] as a means to reduce the complexity of planning, called skeletal-plan refinement and extended in Asbru by temporal time annotations to cope with the temporal properties of actions and uncertainty about the occurrence of states and events.

2.2 Knowledge Roles in Asbru

In Asbru, a plan consists of a name, a set of arguments, including a time annotation (representing the temporal scope of the plan), and five (optional) knowledge roles: *preferences*, *intentions*, *conditions*, *effects*, and a *plan layout* (plan body). Each knowledge role is an abstract label attached to domain knowledge. The knowledge role indicates the role of this knowledge in the inference process [3].

Preferences constrain the applicability of a plan and guide the decisions in the plan selection process. *Intentions* are high-level goals which support tasks such as critiquing [1] and replanning. They are represented by temporal patterns of actions and states that should be maintained, achieved or avoided. *Conditions* are temporal patterns, sampled at a specified frequency, that lead to transitions between plan states. *Effects* describe the relationship between plan arguments and measurable parameters. *Plan layout* describes the order and frequency of the sub-plans' execution.

All plans and actions have a temporal dimension and the plans' execution is controlled by a number of conditions (`filter`, `setup`, `suspend`, `reactivate`, `abort`, and `complete`). A set of mutually exclusive plan states describes the actual status of the plan during the plan selection and the plan execution. An example of a plan written in Asbru is given in Figure 1.

2.3 Temporal Patterns

Most planning systems represent time as states or instances, so conditions in the plan library can be matched with observed values by simple unification. In contrast, the Asgaard framework provides a rich representation to describe the temporal dimension of values to be observed.

The basic syntactic construct is the temporal pattern. All conditions for the transition from one plan state to another are expressed in terms of temporal patterns. A temporal pattern consists of one or more parameter propositions or plan-state descriptions. Each parameter proposition contains a parameter name, a value description, a context description and a time annotation.

The parameter name refers to a stream of input data or an abstraction thereof (e.g., qualitative state change over time). The value description can be a single value, a value range, or a combination of ranges of qualitative values. The context describes a set of situations under which the temporal pattern as a whole is valid.

The time annotation consists of a temporal reference point and three time ranges limiting start, end, and duration of the period during which the expression described by parameter name and value description must evaluate to true. Each of the six values in the three time ranges can be denoted as unknown. The values are called *Earliest Starting Shift (ESS)*, *Latest Starting Shift (LSS)*, *Earliest Finishing Shift (EFS)*, *Latest Finishing Shift (LFS)*, *Minimum Duration (MinDu)* and *Maximum Duration (MaxDu)*. All these time shifts are related to the common temporal reference point. Different time annotations can have different reference points allowing for different time lines for separate courses of actions.

```
(PLAN plan-A
  (INTENTION:OVERALL-STATE
     (MAINTAIN par-1 (> 50) * *))
  (SETUP-PRECONDITIONS
     (par-1 (< 50) par-raising
      [[_, _], [_, _], [10 MIN,_] ref])))
  (ACTIVATED-CONDITIONS AUTOMATIC)
  (ABORT-CONDITIONS ACTIVATED
     (par-1 (< 30) par-raising
        [[10 MIN, _], [_, _], [1 MIN, _],
         *self*])
     (SAMPLING-FREQUENCY 1 MIN))
  (COMPLETE-CONDITIONS
     (par-1 (> 55) par-raising
        [[_, _], [_, _], [10 MIN, _],
         *self*])
     (SAMPLING-FREQUENCY 1 MIN))
  (DO-SOME-ANY-ORDER
     (plan-AA [[_, _], [_, _], [_, 1 MIN],
        (ACTIVATED plan-A)])
     (plan-AB [[_, _], [_, _], [_, 5 MIN],
        (ACTIVATED plan-A)]))
```

Figure 1. Asbru example code. The purpose of `plan-A` is to decrease `par-1`. This is achieved by alternately applying `plan-AA` and `plan-AB` for a maximum duration of one minute and five minutes, respectively. `plan-A` is automatically activated if `par-1` is below 50 for at least 10 minutes. It is aborted, if `par-1` stays below 30 for one minute, starting at least 10 minutes after invocation of `plan-A` and completed if `par-1` is above 55 for at least 10 minutes.

3 ASGAARD'S RUN-TIME MODULES

In this paper we focus on three modules within the Asgaard framework: data abstraction, environment monitoring, and execution. To accomplish the task of plan management as a whole, a number of other modules are indispensable [15], like visualization of the plans both at design-time and run-time [14], plan verification and validation [6], and plan evaluation and critiquing [1].

3.1 Data-Abstraction Module

The data-abstraction module deals with the transformation of information obtained from sensors or user input into a format suitable for the monitoring module. Basically, a distinction between high-frequency domains, in which sensors deliver input at a rate of several records per minute or second, and low-frequency domains, where data are (often manually) entered several times per day or week, should be maintained. While high-frequency domains suffer more from loads of data which are often erroneous, low-frequency domains often suffer from missing values in a line of samples. Although many of the features of the data-abstraction module are also useful for low-frequency data, it is most crucial for high-frequency data.

The issues of data abstraction fall into three main categories: Data validation, calculation of derived values, and transformation into qualitative information.

3.1.1 Data Validation

Any data obtained from sensors, be it monitors in Intensive Care Units (ICUs) or a sensors in robotics, varies in quality over time. Under good conditions it is fairly reliable and exact, but under circumstances not directly visible to the monitoring process they become very unreliable. E.g., the saturation of oxygen measured through the skin of a patient can only be measured in satisfying quality if the patient does not move. Even small movements cause the readings to oscillate wildly. The data-validation module detects such situations and marks the data as invalid to keep other modules from getting confused by the wrong data. A thorough discussion of data-validation issues, focusing on medical high-frequency domains, can be found in [17].

3.1.2 Calculation of Derived Values

In addition to the original value, a number of statistical measures derived from that value (and its neighbors) are important for some monitoring tasks. Examples for such values are the slope and the standard deviation.

Asgaard's data-abstraction module calculates a linear regression for a time window of constant size sliding over the graph of a parameter as new values arrive. By applying two time windows of different size (e.g., one and five minutes) we obtain both short and longer term trends of the parameters observed. Although this calculation is more computational intensive than simple averaging, it is fast enough in practical application. Additionally, it yields slope and standard deviation as an important byproduct.

For alarming, one wants to know the time at which the value of a parameter will reach a critical limit, if the current trend continues. This can easily be calculated by intersecting the produced regression line with the (horizontal) line representing the limit.

All these derived values are feed into the monitoring module in addition to the original data.

3.1.3 Deriving Qualitative Information

The information described in the above section still has the form of two-dimensional data-points describing the value of a parameter at a certain instant of time.

For the purpose of reasoning on the observed data, two transformations are useful. First, the quantitative values should be transformed into qualitative values. Second, instantaneous measurements which have the same qualitative value, should be concatenated to an interval over time during which the parameter's value stays unchanged. ([16, 21].

The transformation to qualitative values is done using a list of qualitative regions, each comprising the symbolic tag of the region and its numerical limits. These limits are context sensitive: A distance of one meter might be near in one context but far in another. Thus each list of limits is valid for a defined set of contexts only. If the context changes, the change from one set of limits to another should be gradually, otherwise the same quantitative value will map to a different qualitative one in the very next moment.

3.2 Monitoring Module

The monitoring module bridges the gap between the data-abstraction and the plan-execution module. It receives the above mentioned intervals, during which a parameter proposition holds, from the data-abstraction module and stores them in a list of *observed parameter propositions (OPPs)*.

The execution module specifies a list of temporal patterns which are key to future state transitions of instantiated plans (compare Section 2.2). These patterns are denoted *monitored parameter propositions (MPP)*. Whenever an OPP matches a MPP, the execution module is notified.

The time annotations (of MPPs) can, but need not, describe the start or/and the end of the time interval during which the designated parameter must hold a particular value. The start of this interval is called positive or left flank, its end is called negative or right flank.

For the issue of monitoring the environment, the earliest time, at which a certain temporal pattern can evaluate to true is of critical interest.

If only the starting time of a parameter proposition is given, action can be taken after the positive flank has been observed, i.e. after the observed parameter obtains the given value (compare left-hand side of Figure 2). If the minimum duration is given, the specified amount of time must pass after the positive flank before the temporal pattern evaluates to true. Similarly, if the earliest finishing shift is given, this point in time must pass, before the temporal pattern becomes true. If the negative flank occurs before the earliest finishing shift the temporal pattern can only become true if another positive flank occurs which is followed by a negative flank after the earliest finishing shift. These details are illustrated by the right-hand side of Figure 2.

If the time of the negative flank is specified, either by the maximum duration or the latest finishing shift, its occurrence must also be observed before the temporal pattern can be considered true (compare center of Figure 2). This means that phenomena whose temporal extend is fully specified, e.g., labor pains lasting for one to two minutes, can only be observed a posteriori, since – most natural – they must be over before one can say whether they lasted for one to two minutes and not longer.

3.3 Execution Module

Mapping of plans and actual situations in the environment is done on three distinct layers: Plan synchronization, plan adaptation, and replanning.

3.3.1 Plan Synchronization

All plans in Asbru have time annotated conditions for their start (`filter` and `setup` conditions), successful completion (`complete` condition) and failure (`abort` condition). These annotations provide flexible means for denoting temporal constrains on the duration of a plan.

The execution of a plan lasts until its targets are reached or its failure is definitely observed, i.e. it is feedback-driven. A plan which is expected to decrease the temperature to below 20^{o}C within 5 minutes is not just executed for 5 minutes and then stopped without further precautions. Instead it is executed until either the measured temperature is below 20^{o}C (possibly for a certain time of observation, to be sure it stay below the limit) or until the maximum time allotted for decreasing the temperature is elapsed.

3.3.2 Plan Adaptation

Plan adaptation is implemented as plan suspension and plan replacement. Every plan can have a `suspend` condition under which it is

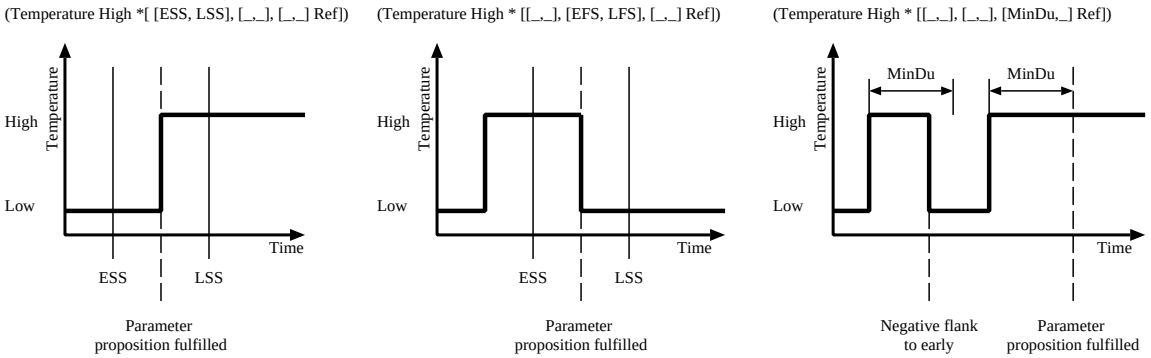

Figure 2. Monitoring changing values according to different parameter propositions: In the first (leftmost) case, only the start is specified, so the parameter proposition is fulfilled in the moment in which the positive flank occurs. In the second case (in the middle) only the end of the interval, during which the temperature must be high, is specified, so the parameter proposition is fulfilled as soon as the negative flank occurs. In the third (rightmost) case, the minimum duration is specified. So the parameter proposition is fulfilled that amount of time after a positive flank, provided that no intermediate negative flank occurred.

temporarily suspended. As soon as the `reactivate` condition is true, the plan is resumed again.

The `abort` condition of a plan specifies the conditions, under which there is no hope to achieve the plan's goal. Aborted plans can be replaced by alternative ones. There are two ways in Asbru's syntax to state alternatives: First, any plan can contain an "`on-abort`"-clause stating a plan to be executed if this plan fails. Second, the plan body can specify groups of alternative plans. The `do-some-any-order`-statement specifies a set of plans which are tried one at a time until all plans specified in the `continuation` condition have completed successfully. If one sub-plan fails, the next one is tried. If the last one in the list failed, the first one which has not already completed successfully is tried again. This is tried until success or expiration of the parent plan's time annotation.

Plans stay active until they are either `completed`, `aborted`, or `suspended`. If an alternative plan turns out to be more attractive then the original one, it is not activated unless the original one fails. This is not as restricting as it seems, since the condition, under which a change to another plan is favorable, can be part of the `abort` condition of the plan to cancel as it is in the `setup` condition of the alternative plan. It ensures the proper implementation of intention of the plan's author.

3.3.3 Replanning

Replanning can occur for two reasons: Plan failure and additional user requests. If a top-level plan fails, the execution module looks for plans (in the plan library), which either have the same intentions, or have effects which remove the reason for the failure of the current plan. In both cases, temporal constraints (as given by the time annotations in the conditions and for the plan as a whole) are observed.

Additional user requests can be specified by either selecting a plan to be executed, an intention, or a goal (in the form of a parameter proposition) to be achieved. As described by [4], users tend to specify goals not only as parameter/value pairs but also as high-level target or as action to be carried out. We thus allow all three forms of specifying the users' demands.

In the first case, the plan is simply started – only plans with satisfied `filter` and `setup` conditions are displayed in the list from which the user may choose. In the second case, the plans which have the same intentions are displayed to the user who selects one of them. In the third case, all plans which have effects leading to the achievement of the stated parameter proposition are displayed to the user, ordered by the likelihood of their influence as given in the `effects`-slot in the plans.

4 RELATED WORK

The increasing emphasis on real-world applications has led researchers to develop approaches that more closely match realistic planning environments [5].

The *Belief-Desire-Intention (BDI) model* of agency [2, 9] inspired several systems similar to Asgaard as well as the design of Asgaard itself. The basic idea is to model the commitment of the agent executing the plans by separating fundamental, long term goals – the desires – which might not even be achievable but still important (e.g., get rich) from short term goals – the intentions – which are inspired by the desires but not necessarily causally linked to them (e.g., work hard). The second, similar important contribution of the BDI model to the planning community was to state clearly that the actions of the planning agent are based solely on beliefs which result from observations and which change over time as opposed to the common closed world model in which the planning agent has complete knowledge and in which the environment does not alter in unforeseen ways.

Asgaard reflects these ideas, first by featuring a distinct knowledge role to express the overall purpose of the plan and second by explicitly modeling those situations, where the perception of the environment at execution time of a plan leads to doubting (and thus `suspend`ing) or discarding (and thus `abort`ing) a particular plan.

Two well-know approaches similar to Asgaard are Cypress [26] and O-Plan [23]. Cypress is a powerful domain-independent framework for defining reactive agents. However, it is an union of heterogeneous, loosely coupled modules, while Asgaard is based on a monolithic design. O-Plan provides a hierarchical planning architecture to support planning and control with temporal and resource constraint handling, but lacks the features of the BDI-architecture, namely the separation, on the one hand, of overall aims and short-times means and on the other hand, of actual world state and beliefs of the executing agent.

In the field of robotics, a number of *reactive control systems* have been developed which execute prepared plans but do not develop new ones (e.g., RAPS [7], PRS [10]).

Rational Based Monitoring (RBM) [25] contributes a new view on the issue of monitoring by proposing alternative-based moni-

tors in addition to plan-based ones. While RBM was developed to control the *planning* process of Prodigy [24] and later adapted for UCPOP [19] by Pollack and McCarthy [20], parts of this concept resemble Asgaard's plan-state model used by the *execution* module. RBM's plan-based usability-condition monitors are implemented by Asgaard's `abort` condition, but quantified-condition monitors are currently missing. Alternative-based monitoring is performed by the `do-*-any-order`-construct in Asgaard, although the currently active plan is not disrupted due to an alternative becoming favorable since this is rarely feasible in practical applications, given the extra costs of changing plans. Still, one can add conditions, under which a shift might appear favorable to the `abort-` or `suspend`-condition to carry it out immediately.

5 CONCLUSION

Asgaard was originally designed for the medical domain. It is most suited for domains with large and complex, but partly vague and incomplete knowledge. In the medical field, direct control over the planning process is crucial, since the user community, namely the medical staff, would refuse a system which does not offer maximum transparency and coverage safety issues in the decision process. Suitable plans are found through search in plan space applying meta-plans called "skeletal plans" [8]. These plans describe a set of sub-targets to be reached by means of conditions, which must be fulfilled. While the skeleton of the final solution is defined by the skeletal plan, it does not define exactly, which sub-plans to select to reach the target. Instead, such plans are found by matching their intentions, their conditions, and the context, for which they are defined, with those demanded by the skeletal plan.

In this paper, we described Asgaard's monitoring and plan-adaptation capabilities in dynamically changing environments. Asbru's hierarchical structure and knowledge roles facilitate the acquisition and maintenance of knowledge from domain experts. The monitoring module presented ensures proper synchronization of plan execution with a changing environment. The plan adaptation capability handles problems arising during execution of a plan, which have been foreseen at planning time. Replanning handles situations not expected at the time the original plan was created by issuing a new plan. This multi-step approach allows smooth reaction to unexpected situations ranging from slight parameter deviations to total mission failure.

ACKNOWLEDGEMENTS

This project is supported by "Fonds zur Förderung der wissenschaftlichen Forschung - FWF" (Austrian Science Fund), P12797-INF.

REFERENCES

[1] A. Advani, K. Lo, and Y. Shahar, 'Intention-based critiquing of guideline-oriented medical care', in *Proceedings of the 1998 AMIA Annual Symposium*, ed., C. G. Chute, pp. 483–487, Orlando, FL, (1998).

[2] M.E. Bratman, *Intention, Plans and Practical Reason*, Havard University Press, Cambridge, MA, 1987.

[3] *CommonKADS Library for Expertise Modelling*, eds., J. Breuker and W. v. d. Velde, IOS Press, Amsterdam, 1994.

[4] M.T. Cox and M.M. Veloso, 'Controlling for unexpected goals when planning in a mixed-initiative setting', in *Progress in Artificial Intelligence: Eight Portuguese Conference on Artificial Intelligence*, eds., E. Costa and A. Cardoso, pp. 309–318, Berlin, (1997). Springer.

[5] M.E. desJardins, E.H. Durfee, C.L. Ortiz, and M.J. Wolverton, 'A survey of research in distributed, continual planning', *AI Magazine*, **4**, 13–22, (1999).

[6] G. Duftschmid, *Knowledge-Based Verification of Clinical Guidelines by Detection of Anomalies*, Ph.D. dissertation, Vienna University of Technology, 1999.

[7] R.J. Firby, 'Task networks for controlling continuous processes: Issues in reactive planning', in *Second International Conference on Artificial Intelligence Planning Systems*, pp. 49–54, (1994).

[8] P.E. Friedland and Y. Iwasaki, 'The concept and implementation of skeletal plans', *Journal of Automated Reasoning*, **1**(2), 161–208, (1985).

[9] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge, *The Belief-Desire-Intention Model of Agency*, Springer Publishers, 1998.

[10] M.P. Georgeff and F.F. Ingrand, 'Decision-making in an embedded reasoning systems', in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Michigan, (1989).

[11] W. Horn, Y. Shahar, G. Lindberg, S. Andreassen, and J. Wyatt, eds. *Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM99)*, volume 1620 of *Lecture Note in Artificial Intelligence*, Berlin, 1999. Springer.

[12] J.F. Horty and M.E. Pollack, 'Evaluating options in a context', in *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-98)*, ed., Itzhak Gilboa, pp. 249–262, San Francisco, (1998). Morgan Kaufmann.

[13] J.F. Horty and M.E. Pollack, 'There's more to life than making plans: Plan management in dynamic, multi-agent environments', *AI Magazine*, **4**, 71–83, (1999).

[14] R. Kosara and S. Miksch, 'Visualisation techniques for time-oriented, skeletal plans in medical therapy planning', In Horn et al. [11], pp. 291–300.

[15] S. Miksch, 'Plan management in the medical domain', *AI-Communications*, **4**, (1999).

[16] S. Miksch, A. Seyfang, W. Horn, and Popow C., 'Abstracting steady qualitative descriptions over time from noisy, high-frequency data', In Horn et al. [11], pp. 281–290.

[17] S. Miksch, A. Seyfang, W. Horn, C. Popow, and F. Paky, 'Methods of temporal data validation and abstraction in high-frequency domains', in *Medical Data Mining and Knowledge Discovery*, ed., Krzysztof Cios, Springer, (2000). forthcoming.

[18] S. Miksch, Y. Shahar, and P. Johnson, 'Asbru: A task-specific, intention-based, and time-oriented language for representing skeletal plans', in *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*, eds., E. Motta, F. van Harmelen, C. Pierret-Golbreich, I. Filby, and N. Wijngaards, Milton Keynes, UK, (1997). The Open University, Milton Keynes, UK.

[19] J.S. Penberthy and D.S. Weld, 'UCPOP: A sound, complete, partial-order planner for ADL', in *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, eds., Charles Rich, William Nebel, and Bernhard Swartout, pp. 103–114, Cambridge, MA, (1992). Morgan Kaufmann.

[20] M.E. Pollack and C. McCarthy, 'Towards focused plan monitoring: A technique and an application to mobile robots', in *IEEE International Symposium on computational intelligence in robotics and automation*, (1999).

[21] Y. Shahar, 'A framework for knowledge-based temporal abstraction', *Artificial Intelligence*, **90**(1-2), 267–98, (1997).

[22] Y. Shahar, S. Miksch, and P. Johnson, 'The Asgaard Project: A task-specific framework for the application and critiquing of time-oriented clinical guidelines', *Artificial Intelligence in Medicine*, **14**, 29–51, (1998).

[23] A. Tate, B. Trabble, and J. Dalton, 'O-Plan: A knowledge-based planner and its application to logistic', in *Advance Planning Technology*, 213–239, Morgan Kaufmann, (1996).

[24] M.M. Veloso, J. Carbonell, M.A. Pérez, D. Borrajo, E. Fink, and J. Blythe, 'Integrating planning and learning: The Prodigy architecture', *Journal of Experimental and Theoretical Artificial Intelligence*, **7**(1), 81–120, (1995).

[25] M.M. Veloso, M.E. Pollack, and M.T. Cox, 'Rationale-based monitoring for planning in dynamic environments', in *Fourth International Conference on Artificial Intelligence Planning and Scheduling*, (1998).

[26] D.E. Wilkins, K.L. Myers, J.D. Lowrance, and L.P. Wesley, 'Planning and reacting in uncertain and dynamic environments', *Journal of Experimental and Theoretical AI*, **7**(1), 197–227, (1995).