Fully Deadline-Coupled Planning: One Step at a Time*

Madhura Nirkhe, Sarit Kraus, Donald Perlis
Department of Computer Science
and
Institute for Advanced Computer Studies
University of Maryland,College Park, MD 20742

Abstract

In planning situations involving tight deadlines a commonsense reasoner may spend substantial amount of the available time in reasoning toward and about the (partial) plan. This reasoning involves, but is not limited to, partial plan formulation, making decisions about available and conceivable alternatives, plan sequencing, and also plan failure and revision. The key observation is that the *time taken in reasoning about a plan brings the deadline closer*. The reasoner should therefore take account of the passage of time during that *same* reasoning, and this accounting must continuously affect every decision under time-pressure. Step-logics were introduced as a mechanism for reasoning situated in time. We employ them here to create a step-logic planner that lets a time-situated reasoner keep track of an approaching deadline as she/he makes (and enacts) her/his plan, thereby treating *all* facets of planning (including plan-formation and its simultaneous or subsequent execution) as deadline-coupled.

1 Introduction

Meta-planning is the usual proposal for reasoning about the reasoning process. But that takes time too! Action occurs in the mere form of thinking or reasoning. In [?], it is argued that traditionally, actions in AI are viewed as separate from the planning process which leads to those actions. Even when the two are intertwined, as in real-time, dynamic or reactive planning, the planning effort is treated as a different kind of beast, not an action itself. Just as it is essential to understand certain features of actions in order to make an intelligent choice of actions in a plan, it is neccesary to reflect upon features of planning to make intelligent decisions in going about planning. When the reasoning is not *in* but *about* deadline situations the time for meta-planning does not enter the computation. In some commonly encountered situations the time taken by meta-planning may be very short. But what of highly novel settings in which one cannot *a priori* assign expected utilities to various conceivable options or refinements? Then the planner is forced to decide on utilities and other factors in real time. In these cases it seems unlikely that such meta-planning will always have a modest time cost. Clearly, the emphasis then is not on searching for a theoretically optimal plan, but one which is speculated to work within the deadline. The reasoner must have the flexibility to interleave planning and execution, not only because there may not be enough time to wait until a complete plan is formulated, but because future planning actions may depend upon the outcomes of earlier executions.

*This is an extended version of our paper [?]. This research was supported in part by NSF grant IRI-8907122, and in part by U.S. Army Research Office grant DAAL03-88-K0087. Our thanks to Michael Miller, James Hendler and Chitta Baral for their helpful comments.

We employ the mechanism of step-logics [?, ?, ?, ?] toward solving this problem. In contrast to other formalisms for commonsense reasoning, step-logics give the reasoner the ability to recognize that her reasoning takes time. What is of special interest to us is not an "ultimate" plan computed in a static world, but a plan which evolves in time in a changing world [?]. Every activity of the reasoner is carried out in fundamental time units called *steps*. The reasoner's thought activity is treated in the same manner as her other activities in the outside world. The two are allowed to take place concurrently with each other and with other changes in the world, in particular, with the ticking of a clock. The reasoner has a (largely) declarative inference engine, with some procedural rules[1].

In what follows,we present an illustrative example; sketch the structure of our program and show a few important steps of the output.

1.1 An Illustration

To elaborate on the fully deadline-coupled planning problem, we present an illustrative domain, which we call the *Nell & Dudley Scenario*[2]: Nell is tied to the railroad tracks as a train approaches. Dudley must formulate a plan to save her and carry it out before the train reaches her. If we suppose Dudley has never rescued anyone before, then he cannot rely on having any very useful assessment in advance, as to what is worth trying. He must deliberate (plan) in order to decide this, yet as he does so the train draws nearer to Nell. We want to prevent Dudley from spending so much time seeking a theoretically optimal plan to save Nell, that in the meantime the train has run Nell down. Moreover, we want Dudley to do this without much help in the form of expected utilities or other prior computation. Thus he must assess and adjust (meta-plan) his on-going deliberations *vis-a-vis* the passage of time. His total effort (plan, meta-plan and action) must stay within the deadline. He must, in short, reason in time about his own reasoning in time.

Our approach has many concerns in common with existing research in planning and temporal reasoning[3]. However, a *fully* deadline-coupled planner has an important qualification that

[1]The time taken for executing such procedures, however, is itself accounted for and declaratively represented, and the results of such procedures are also in declarative form. An example is the WET procedure discussed later.

[2]This problem was first mentioned in the context of time-dependent reasoning by McDermott [?], and more recently discussed in [?].

[3] [?, ?, ?, ?, ?]. However, these works do not account for the time taken for meta-planning. Indeed, this is stated in [?] (page 402): "Here we will not worry about the cost of meta-reasoning itself; in practice, we have been able to reduce it to an insignificant level".

Dean [?] proposed a computational approach to reasoning about events and their effects occurring over time. Dean, Firby and Miller [?] subsequently designed FORBIN, a planning architecture that supports hierarchical planning involving reasoning *about* deadlines, travel time, and resources. Dean and Boddy [?] formulated an algorithmic approach to the solution of time-dependent planning problems by introducing "anytime algorithms" which capture the notion that utility is a monotonic function of deliberation time. Here also, the time for computation is not accounted for : "The time required for deliberation scheduling will not be factored into the overall time allowed for deliberation. For the techniques we are concerned with, we will demonstrate that deliberation scheduling is simple, and, hence, if the number of predicted events is relatively small, the time required for deliberation can be considered negligible," [?] (page 50). [?] demonstrated deliberation scheduling for a time-dependent planning problem involving tour and path planning for a mobile robot.

[?] formulated a world model based on temporal logic which allows the problem solver to gather constraints on the ordering of actions without having to commit to it when a conflict is detected. [?] discusses how a planner can reason about the difficulty of its tasks, and depending on available time, produce reasonable if not optimal solutions. [?] and [?] use a first-order temporal logic model to describe complex synchronization properties of parallel multiagent domains. [?] explore the relation between agent design and environmental factors. They use plans to constrain the reasoning. In [?] a computational approach to temporal reasoning is presented in which a problem solver is forced to make predictions and projections about the future and plan in the face of uncertainty these efforts fail to meet: in addition to determining the current time, estimating the expected execution time of partially completed plans and being able to discard alternatives that are deadline-infeasible, it must also have a built-in way of accounting for all the time spent as a deadline approaches. This means not only accounting for the time of various segments (procedures in the more usual approaches), but also the time for this very accounting for time! Step-logics are used here as a way to do this without the vicious circle of "meta-meta-meta…" hierarchies.

2 Using Step-Logics

Step-logics were introduced as a general-purpose tool to represent reasoning situated in time[4]. An essential feature of step-logics is that inferences are parametrized by the time taken for their inference, and these time parameters can themselves play a role in the specification of the inference rules and axioms. The most obvious way time parameters can enter is via the predicate expression $Now(i)$, indicating the time is now i. Each *step* of reasoning advances i by 1. Step-logics offer a natural representation of the evolving process of reasoning. A *step* is a fundamental unit roughly characterized by the time it takes the reasoner (Dudley, for the purpose of this paper) to draw one inference in the on-going chain of reasoning; it is also the time it takes to perform a primitive action. Observations are inputs from the external world, and may arise at any step i. When an observation appears, it is considered a belief (or fact) in the same time-step.

At each new step i, the only information available to Dudley upon which to base his further reasoning is a snap-shot of his deduction process completed up to and including step $i-1$. During step i, Dudley applies all available inference rules in parallel, but only to beliefs at step $i-1$; new beliefs thus generated at step i through applications of inference rules are not available for use in further inference until step $i+1$. At any given step then, Dudley's beliefs (observations plus inferred beliefs) are *not* the logical closure of his observations, but rather only the (finite) subset he has deduced so far. Actually, this is not strictly correct, since beliefs are not necessarily inherited from one step to the next. The most obvious one is $Now(i)$, which is believed at step i but not at $i+1$[5].

A commonsense reasoner typically has to reason with incomplete information. Thus, default conclusions may have to be made which may later be contradicted on the basis of more information. In step-logics, there is in general[6] time ahead to initiate repairs to one's reasoning, and hence contradictions are not necessarily bad; indeed, they can be seen as very useful clues that something needs to be attended to[7].

---

and incomplete knowledge; time-maps are described here. [?] examine the complexity of temporal reasoning problems involving events whose order is not completely known. We refer the reader to [?] for a general survey of related work on planning.

[4] [?, ?, ?, ?].

[5] Thus step-logics are inherently nonmonotonic, in that further reasoning always leads to retraction of some prior beliefs.

[6] In deadline situations of course there is only limited time ahead, and so the reasoning becomes more delicate. Part of our work has focused on this issue.

[7] [?, ?]. While we do not at present have a completely satisfactory method for resolving all contradictions, we do have techniques for temporarily disarming them.

3 Workings of a Step-Logic Planner

We have created a representational language for some variations of the illustrative Nell & Dudley deadline problem.[8] A few sample axioms and inference rules can be found in the Appendices. At all times Dudley retains his belief about the hard deadline which he must meet. The partial plan is a temporally ordered list of triplets. Each triplet consists of an *action*, preceded and followed, respectively, by a list of *conditions* and *results*. The conditions are not necessarily preconditions, they may need to be true over all or some of the duration of the action. A triplet is written within square brackets [...] and an ordered list of triplets is enclosed within curly brackets {...}. An *action* may be complex or primitive (atomic). A primitive action takes one time step to perform. Primitive actions are deleted from the plan once they have been acted upon. A null plan indicates a solved goal.

As Dudley develops a partial plan to save Nell, he continuously refines his estimate of the time to carry the plan to completion, making sure it will not overshoot the deadline. This we call the *working estimate of time* (WET for short).[9] The WET is Dudley's calculation of how long his partial plan (formed as of the previous step) will take to execute. This he adds to the current time and compares the result to the deadline to make sure the plan is not hopeless. As long as it is not, he declares it *Feasible*, and continues refining and/or putting it into execution. Dudley updates the WET when an action with a known difference between its start and finish times is made part of the plan. Currently, Dudley does not have a procedure to estimate the duration of tasks with unspecified time intervals. We are incorporating such a procedure, and the time taken to execute it is also our concern.

A projection is typically made in the context of a plan. The context of a plan at step i consists of all *Facts* at the step along with the actions in the plan and their results. The context of a *null* plan is made of *Facts* alone. If there are one or more plans at step i, a projection is made in the context of each plan, including the null plan. The projection rule is in accordance with the default of persistence. It projects[10] each formula α in the context set at step i to the maximum interval over which it can persist, up to the point when it clashes with a formula β in the context set involving the same predicate, which is in contradiction[11] with α. If there is no such β the projection ranges up to ∞.

Our approach can be best described by a term which we call *parallel projection*. That is, the entire known state of the world at one moment is used to determine the (expected) state at the next moment. Since step-logics are built around the idea of specifying what is known (e.g., proven) *so far*, all predicates can be simultaneously reconsidered at each new time step.

Planning decisions involve future course of actions, and these must be based on what is *foreseeable*[12] in the current projection in the context of each current partial plan. When a condition C_A is not foreseeable, Dudley attempts to find an axiom of the form $A_1, \ldots, A_k \rightarrow C_A$ and uses it to refine the plan, thus making the condition foreseeable.

[8]This version has been implemented in PROLOG.

[9]The WET is one of our concessions to procedural methods: we do not require Dudley to figure out how to do arithmetic but rather allow that he already knows. But we do require him to note the passage of time *during* the execution of the procedure.

[10]Projections (and persistences) have been studied by numerous authors; see e.g., [?] , [?], [?], [?],[?], [?] and [?]. Our treatment is along the lines of time-maps of [?], [?].

[11]A formula $X(S : F, \ldots)$ is in *direct contradiction* with $\neg X(S : F, \ldots)$. A formula $X(S : F, U, \ldots)$ is in *uniqueness contradiction* with $X(S : F, V, \ldots)$ if $X(S : F, U, \ldots) \rightarrow \neg X(S : F, V, \ldots)$ whenever $U \neq V$. A set of formulas Z *du-contradicts* a formula δ, written as $(Z \rightsquigarrow \neg\delta)$, if there is a formula $\alpha \in Z$ which is in direct or uniqueness contradiction with δ.

[12]A condition C_A is said to be foreseeable if $C_A \in$ **Facts** or ($C_A \in$ **Proj** and **Facts** $\not\rightsquigarrow \neg C_A$).

Dudley may start to act on a primitive action in the partially developed plan, not waiting for the plan to reach completion. At the same time he continues planning [?], [?]. His predicted projections and observations are compared; conflicts resolved in favor of the latter. While acting, projections can suffice to render the condition for a primitive action foreseeable, when the condition is not directly observed, provided the projections do not contradict any facts or new observations.

3.1 Some Illustrative Steps

To illustrate our efforts in a bit more detail, we present below some portions of the output from our PROLOG program that implements the ideas we have been discussing. For more technical details see [?]. Here Nell is a distance of 35 'paces' from Dudley when he first realizes (step 0) that the train will reach her in 50 time units. He begins to form a plan, seen below in step 1 as **Ppl** (partial plan), and refines the plan in subsequent steps. **Deadline** is 50 in this example, d is Dudley, n is Nell, h denotes home and r the railtrack. The subscript *obs* indicates that the wff it is attached to is the result of an observation. Subscripted t's indicate times (step numbers). **Proj** stands for projection; *save*, that appears as argument to **Ppl**, **Proj** and **Feasible** in step 1, is a label naming the plan he is forming. $X(S:T,\ldots)$ denotes that the predicate X holds over the interval $S:T$. A point interval $T:T$ is written simply as T. The "$\bullet\!\!\to$" as it appears in $X(S:T\bullet\!\!\to R,\ldots)$ denotes that X is intended to hold beyond $S:T$ up to R (by default). Its use in a result of an action indicates that the result must be preserved for use in a later segment of the plan. The difference between ":" and "$\bullet\!\!\to$" will make itself clear in the next section. The number at the right bottom corner of a triplet denotes its place in the plan sequence.

0: $\mathbf{Facts}(\{At(0,d,h)_{obs}, Tied(0,n,r)_{obs}\})$, $\mathbf{Deadline}(50)$, $\mathbf{Goal}(out\_of\_danger(50,n,r))$

1: $\mathbf{Facts}(\{At(0,d,h), Tied(0,n,r)\})$, $\mathbf{Deadline}(50)$, $\mathbf{Unsolved}(Goal(out\_of\_danger(50,n,r)))$,
$\mathbf{Ppl}(save,1,\{out\_of\_danger(50,n,r)\})$, $\mathbf{Proj}(save,1,\{At(1:\infty,d,h), Tied(1:\infty,n,r)\})$,
$\mathbf{WET}(save,0)$, $\mathbf{Feasible}(save,0)$

2: $\mathbf{Facts}(\{At(0,d,h), Tied(0,n,r)\})$, $\mathbf{Ppl}\left(save, 2, \left\{ \begin{bmatrix} \neg Tied(t_1,n,r) \\ Pull(t_1:t_2,d,n,r) \\ Out\_of\_danger(t_2 \bullet\!\!\to 50,n,r) \end{bmatrix}_1 \right\},\right.$
$\{t_2 \le 50, t_1 = t_2 - 1\}), \mathbf{WET}(save,0),\ \ldots$

3: $\mathbf{Ppl}\left(save, 3, \left\{ \begin{bmatrix} At(t_3:t_4,d,r) \\ Release(t_3:t_4,d,n,r) \\ \neg Tied(t_4 \bullet\!\!\to t_1,n,r) \end{bmatrix}_1 \begin{bmatrix} \neg Tied(t_1,n,r) \\ Pull(t_1:t_2,d,n,r) \\ Out\_of\_danger(t_2 \bullet\!\!\to 50,n,r) \end{bmatrix}_2 \right\},\right.$
$\{t_2 \le 50, t_1 = t_2 - 1, t_3 = t_4 - 3, t_4 \le t_1\})$
$\mathbf{Proj}(save,3,\{At(1:\infty,d,h), Tied(1:\infty,n,r), Out\_of\_danger(t_2+1:\infty,n,r),\})$,
$\mathbf{WET}(save,1),\ \ldots$

4: $\mathbf{Ppl}\left(save, 4, \left\{ \begin{bmatrix} At(t_6,d,L) \\ Run(t_6:t_7,d,L:r) \\ At(t_7 \bullet\!\!\to t_3,d,r) \end{bmatrix}_1 \begin{bmatrix} At(t_3,d,r) \\ Release(t_3:t_4,d,n,r) \\ \neg Tied(t_4 \bullet\!\!\to t_1,n,r) \end{bmatrix}_{2\ldots} \right\},\right. \{t_2 \le 50, t_1 = t_2 - 1,$
$t_3 = t_4 - 3, t_4 \le t_1\ t_6 < t_7, t_7 \le t_3\})$
$\mathbf{Proj}(save,4,\{\ldots, Tied(1:t_4-1,n,r), \neg Tied(t_4+1:\infty,n,r), \ldots\})$, $\mathbf{WET}(save,4),\ \ldots$

⋮

To understand the step-like reasoning here, consider the workings of Step 4. Looking at the belief set at the end of Step 3, Dudley reasons that since he is not projected to be at the railroad track by t_3 (he is projected to be at h and $h \ne r$). He finds an axiom which tells him that

Run is a possible action to be at another place. However, he still does not know the location L from which he must *Run* and hence, this is still unbound. This will be bound to h in step 5, when he will attempt to make the condition for the *Run* foreseeable, by first looking up the projection. Notice also his projection in the context of the plan *save* at the end of step 4; he no longer believes that Nell will be *Tied* until infinity, he has trimmed down that range to now extend only up to $t_4 - 1$, at which time he is projected to have *Released* her.

Dudley's plan formulation and simultaneous execution continues, until at step 46 he has saved Nell, 4 steps before the deadline $Ddl = 50$.

3.2 The Knots May Be Too Tight, a Knife May Be Needed

In this research, we incrementally consider more complex scenarios, so that by abstracting from them we can identify more critical issues and enhance the framework with additional time-situated planning capability. Suppose that Dudley thinks that a knife may be required to cut the difficult knots around Nell, and that he should plan for that contingency. He knows of a knife in the house, he projects it to be there when he needs to use it. Requiring a knife corresponds to a compound condition for the action $Cut\_ropes(S : F, \ldots)$, namely, $At(S : F, d, r) \wedge Have(S : F, d, knife)$. We introduce an inference rule (Appendix B, Rule 11) whereby Dudley can subsequently formulate two plans, one in which he plans to satisfy $Have(\ldots)$ before $At(\ldots)$ and the other in which this order is reversed. Both conditions, must, however, hold up to the time they are needed for the *Cut_ropes* action. This is where the "$\bullet\!\!\rightarrow$" comes in use. We introduce another inference rule (Appendix B, Rule 12) that enables Dudley to notice that when the result of an action is expected to be preserved up to the time when it is to be used, a plan in which it must be undone in order to satisfy the condition for a subsequent action, is in fact inefficient, and can be frozen in favor of another plan.

This rule, though domain independent, does not claim to handle every situation involving conjunctive goals. It can be thought of as one heuristic aid used by commonsense reasoners in limited time to help in plan selection. In the second plan, picking up the knife requires Dudley to be at home (the same location as the knife), and this violates his attempt to achieve $At(t_{11}, d, r)$ and preserve it until the time t_4 when he will finish untying Nell. This rule fires and he chooses to proceed with the first plan in favor of the second. We demonstrate below a few key steps in this reasoning.

$\vdots$

$$\begin{array}{ll}
\mathbf{3:} & \mathbf{Facts}(\{At(0,d,h), At(0,knife,h), Tied(0,n,r)\}), \mathbf{Deadline}(50), \\
& \mathbf{Unsolved}(Goal(out\_of\_danger(50,n,r))), \\
& \mathbf{Ppl}(save, 3, \left\{ \begin{bmatrix} At(t_3 : t_4, d, r) \wedge Have(t_3 : t_4, d, knife) \\ Cut\_ropes(t_3 : t_4, d, n, r) \\ \neg Tied(t_4 \bullet\!\!\rightarrow t_1, n, r) \end{bmatrix}_{1\ldots} \ldots \right\}, \\
& \{\ldots, t_3 = t_4 - 3, \ldots\}) \\
& \mathbf{Proj}(save, 3, \{At(1:\infty, d, h), At(1:\infty, knife, h), Tied(1 : t_4 - 1, n, r), \ \ldots\})\ldots \\
\mathbf{4:} & \mathbf{Ppl}(save1, 4, \left\{ \begin{bmatrix} At(t_6, d, L_1) \wedge At(t_6, knife, L_1) \\ Pick\_up(t_6 : t_7, d, knife) \\ Have(t_7 \bullet\!\!\rightarrow t_4, d, knife) \end{bmatrix}_1 \begin{bmatrix} At(t_8, d, L_2) \\ Run(t_8 : t_9, d, L_2 : r) \\ At(t_9 \bullet\!\!\rightarrow t_4, d, r) \end{bmatrix}_{2\ldots} \right\}, \\
& \{\ldots, t_6 = t_7 - 1, \ldots\}) \\
& \mathbf{Ppl}(save2, 4, \left\{ \begin{bmatrix} At(t_{10}, d, L_4) \\ Run(t_{10} : t_{11}, d, L_4 : r) \\ At(t_{11} \bullet\!\!\rightarrow t_4, d, r) \end{bmatrix}_1 \begin{bmatrix} At(t_{12}, d, L_3) \wedge At(t_6, knife, L_3) \\ Pick\_up(t_{12} : t_{13}, d, knife) \\ Have(t_{13} \bullet\!\!\rightarrow t_4, d, knife) \end{bmatrix}_{2\ldots} \right\}, \\
& \{\ldots, t_{12} = t_{13} - 1, \ldots\})
\end{array}$$

$$\begin{aligned}
&\mathbf{Proj}(\mathit{save}1, 4, \{\mathit{At}(1:\infty, d, h), \mathit{At}(1:\infty, \mathit{knife}, h), \mathit{Tied}(1:t_4-1, n, r),\ \ldots\})\\
&\mathbf{Proj}(\mathit{save}2, 4, \{\mathit{At}(1:\infty, d, h), \mathit{At}(1:\infty, \mathit{knife}, h), \mathit{Tied}(1:t_4-1, n, r),\ \ldots\})\ldots
\end{aligned}$$

$$\mathbf{5:}\quad \mathbf{Ppl}(\mathit{save}1, 5, \left\{ \begin{bmatrix} \mathit{At}(t_6, d, h) \wedge \mathit{At}(t_6, \mathit{knife}, h) \\ \mathit{Pick\_up}(t_6 : t_7, d, \mathit{knife}) \\ \mathit{Have}(t_7 \bullet\!\!\rightarrow t_4, d, \mathit{knife}) \end{bmatrix}_1 \begin{bmatrix} \mathit{At}(t_8, d, h) \\ \mathit{Run}(t_8 : t_9, d, h : r) \\ \mathit{At}(t_9 \bullet\!\!\rightarrow t_4, d, r) \end{bmatrix}_{2\ldots} \right\},$$

$$\{\ldots, t_6 = t_7 - 1, \ldots\})$$

$$\mathbf{Ppl}(\mathit{save}2, 5, \left\{ \begin{bmatrix} \mathit{At}(t_{10}, d, h) \\ \mathit{Run}(t_{10} : t_{11}, d, h : r) \\ \mathit{At}(t_{11} \bullet\!\!\rightarrow t_4, d, r) \end{bmatrix}_1 \begin{bmatrix} \mathit{At}(t_{12}, d, h) \wedge \mathit{At}(t_6, \mathit{knife}, h) \\ \mathit{Pick\_up}(t_{12} : t_{13}, d, \mathit{knife}) \\ \mathit{Have}(t_{13} \bullet\!\!\rightarrow t_4, d, \mathit{knife}) \end{bmatrix}_{2\ldots} \right\},$$

$$\{\ldots, t_{12} = t_{13} - 1, \ldots\}) \quad \ldots$$

$$\mathbf{6:}\quad \mathbf{Freeze}(\mathit{save}2, 5),\ \ldots$$

$$\vdots$$

4 Conclusions and Future Work

This is the summary of an intermediate phase of a long-term project of which the eventual goal is to model a flexible reasoner with a large database of knowledge ranging over a variety of domains. Our efforts thus far are evidence that a logic-based real-time planner is feasible. Step-logics allow us to treat reasoning and action in an analogous framework, and thus provides a built-in platform for real-time reasoning. More is underway, especially regarding competing plans and interacting subplans. One of the crucial issues we are attempting to address is how to decide on the basis of the current partial plan(s) and their working estimate(s) of time, "whether to act now or deliberate further". In the future we also wish to extend our implementation in various ways, some of the immediate concerns are: to tackle issues involving the frame problem [?] and ramifications and qualifications [?, ?], to strengthen the working estimates of time for unfinished plans, to find explicit representations for extended actions, and to handle failure and error recovery, perhaps also to incorporate perceptual reasoning[13] and a limited capability for qualitative reasoning.

A SAMPLE AXIOMS[14]

Relevant to moving:

- $\mathit{Run}(T_1 : T_2, Y, L_1 : L_2) \rightarrow \mathit{At}(T_2, Y, L_2)\ , T_2 = T_1 + (L_2 - L_1)/v_Y$[15]
- $\mathit{condition}(\mathit{Run}(T_1 : T_2, Y, L_1 : L_2), \mathit{At}(T_1, Y, L_1))$
- $\mathit{result}(\mathit{Run}(T_1 : T_2, Y, L_1 : L_2), \mathit{At}(T_2, Y, L_2))$

Relevant to untying and releasing:

- $\mathit{Pull}(T : T+1, X, L) \rightarrow \mathit{Out\_of\_danger}(T+1, X, L)$
- $\mathit{condition}(\mathit{Pull}(T : T+1, X, L), \neg \mathit{Tied}(T, X, L))$
- $\mathit{result}(\mathit{Pull}(T : T+1, X, L),\ \mathit{Out\_of\_danger}(T+1, X, L))$

[13]This ties to spatial reasoning, and to aspects of a plan that involve getting more information; for instance Dudley may have to move in order to see whether Nell is tied. This in turn relates to existing work ([?], [?]) on ignorance and perception.

[14]These constitute a large part of the current set of axioms and inference rules. [?] gives a more comprehensive initial set.

[15]v_Y is Y's speed while running.

- $Pick\_up(T:T+1,Y,X) \rightarrow Have(T+1,Y,X)$
- $result(Pick\_up(T:T+1,Y,X), Have(T:T+1,Y,X))$
- $condition(Pick\_up(T:T+1,Y,X),$
  $At(T:T+1,X,L) \wedge At(T:T+1,Y,L))$

B SAMPLE INFERENCE RULES[14]

1. Agent looks at the clock

$$\frac{i: \ldots}{i+1: \ldots, \mathbf{Facts}(i+1, \{\ldots, Now(i+1)\})}$$

2. Modus Ponens(MP)

$$\frac{i: \ldots, \mathbf{Facts}(i, \{\ldots, \alpha, \ldots, (\alpha \rightarrow \beta)\})}{i+1: \ldots, \mathbf{Facts}(i, \{\ldots, \beta\})}$$

3. Inheritance(INH)

$$\frac{i: \ldots, \mathbf{Facts}(i, \{\ldots, \alpha\})}{i+1: \ldots, \mathbf{Facts}(i+1, \{\ldots, \alpha\})}$$

4. Observations become instant beliefs

$$\frac{i: \ldots}{i+1: \mathbf{Facts}(\ldots, \alpha); \alpha \in \mathbf{OBS}(i+1)}$$

5. Forms the first partial plan

$$\frac{i: \mathbf{Goal}(G)}{i+1: \mathbf{Ppl}(p, i+1, \{G\}), \mathbf{Feasible}(p, i)}$$

6. Finds a triplet for the goal

$$\frac{i: \mathbf{Ppl}(p, i, \{G\}), \mathbf{result}(A, R_A), \mathbf{condition}(A, C_A), A \rightarrow G}{i+1: \mathbf{Ppl}(p, i+1, \left\{ \begin{bmatrix} C_A \\ A \\ G \wedge R_A \end{bmatrix} \right\})}$$

7. Computes the WET

$$\frac{i: \mathbf{Ppl}(p, i, \left\{ \begin{bmatrix} C_{A_1} \\ A_1(s_1 : f_1, \ldots) \\ R_{A_1} \end{bmatrix} \ldots \begin{bmatrix} C_{A_k} \\ A_k(s_k : f_k, \ldots) \\ R_{A_k} \end{bmatrix} \right\})}{i+1: \mathbf{WET}(p, i, \sum_{j=1}^{k}(f_j - s_j))}$$

the summation is limited to those j values for which $(f_j - s_j)$ is known

8. Keeps track of feasibility

$$\frac{i: \mathbf{Ppl}(p, i, \{\ldots\}), \mathbf{Deadline}(Ddl), \mathbf{WET}(p, i-1, n)}{i+1: \mathbf{Feasible}(p, i)}$$

if $n + i \leq Ddl$.

9. Projection(PROJ)

$$\frac{i : \ldots, \mathbf{Facts}(i, \{\ldots, X(S:F, \ldots)\}), \mathbf{Context\_Set}(i, p, CS)}{i+1 : \ldots, \mathbf{Proj}(i+1, \mathit{null}, \{\ldots, X(S:R, \ldots), \ldots\})}$$

if $(S:R)$ is the maximum interval such that
$\neg\exists(T)\{S \leq T \leq R \wedge (CS \rightsquigarrow \neg X(T, \ldots))\}$ [16]

10. Refine a non-primitive action

$$\frac{i : \mathbf{Ppl}(p, i, \left\{ \ldots \begin{bmatrix} C_A \\ A \\ R_A \end{bmatrix} \ldots \right\}), Q_1 \wedge \ldots \wedge Q_k \rightarrow A}{i+1 : \mathbf{Ppl}(p, i+1, \left\{ \ldots \begin{bmatrix} C_{Q_1} \\ Q_1 \\ R_{Q_1} \end{bmatrix} \ldots \begin{bmatrix} C_{Q_k} \\ Q_k \\ R_{Q_k} \end{bmatrix} \ldots \right\})}$$

provided every condition in C_A is *foreseeable*

11. Spawn the generation of two plans on encountering a compound condition[17]

$$\frac{i : \mathbf{Ppl}(p, i, \left\{ \ldots \begin{bmatrix} C'_A(R:S, \ldots) \wedge C''_A(V:W, \ldots) \\ A \\ R_A \end{bmatrix}_j \ldots \right\}), A' \rightarrow C'_A, A'' \rightarrow C''_A}{i+1 : \begin{array}{l} \mathbf{Ppl}(p_1, i+1, \left\{ \ldots \begin{bmatrix} \ldots \\ A' \\ C'_A(P:Q \bullet\!\!\rightarrow S) \end{bmatrix}_{j-2} \begin{bmatrix} \ldots \\ A'' \\ C''_A(T:U \bullet\!\!\rightarrow W) \end{bmatrix}_{j-1} \ldots \right\}) \\ \mathbf{Ppl}(p_2, i+1, \left\{ \ldots \begin{bmatrix} \ldots \\ A'' \\ C''_A(T:U \bullet\!\!\rightarrow W) \end{bmatrix}_{j-2} \begin{bmatrix} \ldots \\ A' \\ C'_A(P:Q \bullet\!\!\rightarrow S) \end{bmatrix}_{j-1} \ldots \right\}) \end{array}}$$

12. Freeze a plan when it is found to be inefficient

$$\frac{i : \mathbf{Ppl}(p, i, \left\{ \ldots \begin{bmatrix} \ldots \\ A_j \\ R_{A_j}(P:Q \bullet\!\!\rightarrow S) \end{bmatrix} \ldots \begin{bmatrix} C_{A_k}(V:W, \ldots) \\ A_k \\ \ldots \end{bmatrix} \ldots \right\})}{i+1 : \mathbf{Freeze}(p, i)}$$

if $P:S$ and $V:W$ overlap, and R_{A_j} and C_{A_k} are in direct or uniqueness contradiction

References

[Allen and Koomen, 1983] Allen, J. and Koomen, J. 1983. Planning using a temporal world model. In *Proceedings of the 8th Int'l Joint Conference on Artificial Intelligence*, pages 741–747.

[Boddy and Dean, 1989] Boddy, M. and Dean, T. 1989. Solving time-dependent planning problems. In *Proceedings of IJCAI-89*, pages 979–984, Detroit, Michigan.

[Charniak and McDermott, 1985] Charniak, E. and McDermott, D. 1985. *Introduction to artificial intelligence.* Addison-Wesley, Reading, Mass.

[Cohen and Levesque, 1990] Cohen, P. and Levesque, H. 1990. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261.

[Davis, 1988] Davis, D. E. 1988. Inferring ignorance from the locality of visual perception. In *Proceedings, AAAI-88*, St. Paul, Minnesota.

[16] for a definition of $\rightsquigarrow$ see footnote 12

[17] This rule can be easily generalized to more than two conjuncts in a condition.

[Dean and Boddy, 1988a] Dean, T. and Boddy, M. 1988. Reasoning about partially ordered events. *Artificial Intelligence*, 36(3):375–399.

[Dean and Boddy, 1988b] Dean, T. and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings, AAAI-88*, pages 49–54, St. Paul, Minnesota.

[Dean and McDermott, 1987] Dean, T. and McDermott, D. 1987. Temporal data base management. *Artificial Intelligence*, 32(1):1–55.

[Dean *et al.*, 1988] Dean, T., Firby, R. J., and Miller, D. 1988. Hierarchical planning involving deadlines, travel time and resources. *Computational Intelligence*, 4:381–389.

[Dean, 1984] Dean, T. 1984. Planning and temporal reasoning under uncertainty. In *IEEE Workshop on Principles of Knowledge Based Systems*, Denver, Colorado.

[Dean, 1987] Dean, T. 1987. Intractability and time-dependent planning. In *Reasoning about Actions and Plans*, pages 245–266. Morgan-Kaufmann, Los Altos, CA.

[Drapkin and Perlis, 1986a] Drapkin, J. and Perlis, D. 1986. A preliminary excursion into step-logics. In *Proceedings SIGART International Symposium on Methodologies for Intelligent Systems*, pages 262–269. ACM. Knoxville, Tennessee.

[Drapkin and Perlis, 1986b] Drapkin, J. and Perlis, D. 1986. Step-logics: An alternative approach to limited reasoning. In *Proceedings of the European Conf. on Artificial Intelligence*, pages 160–163. Brighton, England.

[Elgot-Drapkin and Perlis, 1990] Elgot-Drapkin, J. and Perlis, D. 1990. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(1):75–98.

[Elgot-Drapkin, 1988] Elgot-Drapkin, J. 1988. *Step-logic: Reasoning Situated in Time.* PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland.

[Georgeff and Lansky, 1988] Georgeff, M. and Lansky, A. 1988. Reactive reasoning and planning. In *Proceedings AAAI-88*, pages 677–682.

[Ginsberg and Smith, 1987a] Ginsberg, M. L. and Smith, D. E. 1987. Reasoning about action I: A possible worlds approach. In Brown, F. M., editor, *Proceedings of the 1987 Workshop on The Frame Problem*, pages 233–258, Lawrence, KS. Morgan Kaufmann.

[Ginsberg and Smith, 1987b] Ginsberg, M. L. and Smith, D. E. 1987. Reasoning about action II: The qualification problem. In Brown, F. M., editor, *Proceedings of the 1987 Workshop on The Frame Problem*, pages 259–287, Lawrence, KS. Morgan Kaufmann.

[Haas, 1985] Haas, A. 1985. Possible events, actual events, and robots. *Computational Intelligence*, 1.

[Haugh, 1987] Haugh, B. A. 1987. Simple causal minimizations for temporal persistence and projection. In *Proceedings of the sixth national conference on artificial intelligence*, pages 218–223.

[Hendler *et al.*, 1990] Hendler, J., Tate, A., and Drummond, M. 1990. Systems and techniques: AI planning. *AI Magazine*, 11(2):61–77.

[Horvitz *et al.*, 1989] Horvitz, E., Cooper, G., and Heckerman, D. 1989. Reflection and action under scare resources: Theoretical principles and empirical study. In *Proceedings of IJCAI-89*, pages 1121–1127, Detroit, Michigan.

[Horvitz, 1988] Horvitz, E. J. 1988. Reasoning under varying and uncertain resource constraints. In *Proceeding, AAAI-88*, pages 111–116, St. Paul, Minnesota.

[Kanazawa and Dean, 1989] Kanazawa, K. and Dean, T. 1989. A model for projection and action. In *Proceedings of IJCAI-89*, pages 985–990.

[Kautz, 1986] Kautz, H. 1986. The logic of persistence. In *Proceedings, AAAI-86*, pages 401–405.

[Kraus and Perlis, 1989] Kraus, S. and Perlis, D. 1989. Assessing others' knowledge and ignorance. In *Proc. of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 220–225, Charlotte, North Carolina.

[Kraus *et al.*, 1990] Kraus, S., Nirkhe, M., and Perlis, P. 1990. Deadline-coupled real-time planning. In *Proceedings of 1990 DARPA workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 100–108, San Diego, CA.

[Lansky, 1986] Lansky, A. 1986. A representation of parallel activity based on events, structure, and causality. In *Reasoning about Actions and Plans*, pages 123–159. Morgan-Kaufmann, Los Altos, CA.

[Lansky, 1988] Lansky, A. 1988. Localized event-based reasoning for multiagent domains. *Computational Intelligence*, 4:319–340.

[McDermott, 1978] McDermott, D. 1978. Planning and acting. *Cognitive Science*, 2:71–109.

[McDermott, 1982] McDermott, D. 1982. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6.

[McDermott, 1987] McDermott, D. 1987. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412.

[Morgenstern, 1991] Morgenstern, L. 1991. Knowledge and the frame problem. In *The Frame Problem in Artificial Intelligence*. JAI Press.

[Nilsson, 1983] Nilsson, N. 1983. Artificial intelligence prepares for 2001. *AI Magazine*, 4(4):7–14.

[Perlis *et al.*, 1990] Perlis, D., Elgot-Drapkin, J., and Miller, M. 1990. Stop the world! – I want to think! Submitted to *International J. of Intelligent Systems*.

[Pollack and Ringuette, 1990] Pollack, M. E. and Ringuette, M. 1990. Introducing the tileworld: Experimentally evaluating agent architectures. In *Proceedings, AAAI-90*, pages 183–189.

[Russell and Wefald, 1989] Russell, S. and Wefald, E. 1989. Principles of metareasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*. Morgan-Kaufman.

[Wilensky, 1983] Wilensky, R. 1983. *Planning and understanding*. Addison Wesley, Reading, Mass.