Abstract

Title of Dissertation: Time-Situated Reasoning within Tight Deadlines and Realistic Space and Computation Bounds

Madhura Nirkhe, Doctor of Philosophy, 1994

Dissertation directed by: Professor Joseph Ja' Ja', and
Associate Professor Donald Perlis
Department of Electrical Engineering

We develop an effective representational and inferential framework for *fully deadline-coupled* planning and acting in *hard*, *non-extensible*, *unforeseen* deadline situations. While meta-planning is the usual proposal for reasoning about the reasoning process, few formalisms to date acknowledge that meta-planning takes time too. In traditional mechanisms, the planning effort is treated as a different kind of beast, not an action itself that takes time. A paradigmatic time-critical problem scenario was chosen: *Nell, Dudley and the railroad tracks.* Nell is tied to the railroad tracks and a train approaches. Dudley, the agent, must formulate a plan to save her and carry it out before the oncoming train reaches her.

We design a time-situated inference mechanism based on an underlying framework of active logics. We demonstrate the generality of our formal methods by using them to tackle projection issues in some real-time versions of the canonical Yale Shooting Problem.

In a realistic setting, an agent planning under time-pressure must also measure up to two other crucial resource limitations as well, namely, space and computation bounds. We introduce a limited short-term memory combined with a primitive relevance mechanism, and a limited-capacity inference engine. We propose heuristics to maximize an agent's chances of meeting a deadline in this enhanced framework.

Logical omniscience, and inferential closure are computational impossibilities for an agent embedded in a real environment. We construct a variation on active logics for which there is a sound and complete modal semantics. It overcomes the key obstacle of closure under consequence. This result illustrates the similarity and differences between active-logic approaches to knowledge and belief, and previous modal approaches.

To summarize the novel elements in this dissertation: we present a declarative planning framework based on active logics which can account for all the time taken to plan, making it suitable for deadline situations. We describe a novel treatment of temporal projection in a real-time setting, a modal semantics for active logics, and additional results in reasoning under resource limitations. Several aspects of this design have been implemented and tested.

Time-Situated Reasoning within Tight Deadlines and Realistic Space and Computation Bounds

by

Madhura Nirkhe

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1994

Advisory Committee:

Professor Joseph Ja' Ja', Chair/Advisor
Associate Professor Donald Perlis, Research Advisor
Assistant Professor Ronald Greenberg
Assistant Professor V. S. Subrahmanian
Assistant Professor Michael Morreau

Dedication

To Surabhi and Vivek

Acknowledgements

I am filled with a deep sense of gratitude as I think of the people without whose help and guidance, I would not have been where I am today. Foremost, I would like to express sincere thanks to Don Perlis, my research advisor. I have learnt a lot from him, on the subject of artificial intelligence, and more importantly about the process of conducting research. He taught me how formulate and approach problems, and how to pursue them with patience and perseverance. His encouragement and support has been invaluable in this work. During interruptions such as the birth of our daughter Surabhi and our move to Florida, Don has been especially considerate and accommodating, always ready to spend quality time with me in these two very eventful years in my life.

I wish to thank my academic advisor Joseph Ja' Ja', for showing undeterring confidence in my abilities when I needed it most, and for providing me with silent encouragement and reassurance during the most difficult times.

I wish to thank the other members of my committee, Michael Morreau, V.S. Subrahmanian and Ronald Greenberg for their input and comments. Michael Morreau has added rigor to the modal active logic portion of the thesis; I thank him for his time and efforts.

I wish to thank Jack Minker, to whom I could turn any time I needed help. I thank him for being there for me since my first semester at Maryland, for standing steadfast by me, and for being so caring and understanding.

Thanks are due to my friend and co-researcher Sarit Kraus, who has been a source of inspiration to me. Sarit has practically been a co-advisor to me during the entire research. I have admired her conscientious and industrious manner as a researcher. She has provided me with timely advice, both professional and personal. As a caring mother of three who does not compromise on either career or personal obligations,

and as a person who takes pride in her work for its own sake, she has been a rare example of a successful woman in science with whom it has been a special privilege and pleasure to do research.

I do not have enough words to thank Vivek, who is my best friend, confidant, and husband. He always backed me to go ahead and pursue whatever captured my interest. He urged me to never ever give up. He showed confidence in my strengths and has helped me overcome many a weakness. My relationship with him has been the biggest source of romance, joy and strength to me in the last twelve years. Vivek has shared every moment of parenthood with me since Surabhi's birth in 1992, always putting my career and interests before his own. In the past year, he has often been a single father while I spent long hours with the computer. He is a wonderful and caring father, and a very compassionate person with a golden heart. Thank you for everything, Vivek, I could not have done it without you.

I wish to thank our beautiful daughter Surabhi who has brought us unbounded satisfaction and pleasure, filling our lives with her sweet little presence. For a two year old, she has been so understanding about her mother having to spend time away from her! She has abundantly returned our love and caused me to view my life in a very different perspective during this time. Vivek and Surabhi deserve a good portion of the credit for this accomplishment.

I wish to thank my parents Usha and Achyut for kindling my curiosity and initiating in me a passion for learning from a very tender age. My father nurtured our interest in science by spending endless hours with us as children, nudging us to ask questions like "tell me why". Both my parents, through practice and example taught us to seek knowledge and light. They have made my joys and sorrows their own as far back as I remember. I owe them a big part of what I am today. I also wish to thank them for making the first year of Surabhi's life so special and memorable.

I wish to thank my brother Niranjan for providing me the challenge to finish my

doctorate before his. He beat me hands down, nevertheless. I am proud to have a brother like him. I have shared so many precious moments with him; I will cherish them forever. I wish to thank my sister Vineeta and her family for their support and love. I owe special gratitude to Anasuya, who has been part of our family since I was three months old. I thank her earnestly for her complete and selfless love, and for the many tears she has shed on account of me and my siblings. She deserves an important share in this credit.

I wish to thank Vivek's parents, Usha and Madhukar for their love and encouragement in the course of the last decade. They have always wholeheartedly supported me in my pursuits. I wish to thank Vivek's mother for specially coming from India at very short notice to help us in the first months after Surabhi's birth. At times they have been more than a mother and father to me. I also wish to thank Vivek's sister Varsha and her family, and his sister Vidya for all their love and affection during these years.

I have several friends to thank who have helped shape my thought processes in the last decade of my life, and who have made my stay in the U.S. so very enjoyable. In particular, I wish to mention P.J. Narayanan, who has been a special friend. I have learnt a lot from him. Thanks PJ for the many interesting conversations, for the happy days of trekking and cross-country driving, for babysitting Surabhi in Pittsburgh, and for being a very close friend! I wish to thank Dheeraj for his friendship. Dheeraj was always ready to extend a hand when you needed it. I will remember the many special long talk sessions I have had at his home. I would like to say that Raghu, Sunil, Venu, Rao, Chaitali, Raja, Shreekant, Indrani, Tridib, Anuradha, Jaggannathan, Vidya, Ksri, Sekar, SC, Laurie, Sonia, Malini, Lalitha, Sharmila, Shamik, Manas, Mani, among others filled my life here with a sense of belonging. Thanks everyone for your company, for the laughs, for the innumerable cups of tea and the long cack sessions.

I wish to extend special thanks to Nirupama for providing me a home away from

home when I visited Maryland in the last year. Niru and her room-mates Liji, Romita and Vandana always welcomed me with a smile and shared my happy and tense moments. I thank Sharmila for her hospitality and friendship. I thank Shamik, and Manas and Mani for their ever willing help and company.

I thank my office-mates Carolina, Parke, Jose Alberto, Terry, Michael Miller, Yuan, Chitta, Shekhar, Lev, and Jarek for their company, for keeping my desk for me, and for the champagne. I thank Felicia for her ready help and sweet smiles. I thank Carolina and Jose Alberto for listening to boring details of my proofs and for providing very valuable feedback.

I thank Amy Weinberg for her support (both financial and personal) in the course of the research assistantship that I did under her supervision, and for letting me have the key to her office.

I wish to thank Dianne O'Leary and the cs-women group at Maryland for the brown bag lunches. Thanks to Christine Hofmeister, Jo Atlee, Kathleen Romanik, Suzanne Stevenson, Clare Voss, Liz White, Anne Wilson – it has been great knowing you. I thank the "csd.grad.moms" for their encouraging presence. Thank you for sharing so much with me. I wish to thank Nancy Lindley for her warmth, concern and cheerfulness.

I thank NSF (grants IRI-8907122 and IRI-9123460) and the U.S. Army Research Office (grant DAAL03-88-K0087) for providing financial support for parts of this research.

Lastly, I thank all those whose names I may have failed to mention here. I thank you all.

Table of Contents

List of Figures

Chapter 1

Introduction

In planning situations involving tight deadlines a commonsense reasoner may spend a substantial amount of the available time in reasoning toward and about a (partial) plan of action. But the time taken in such reasoning brings the deadline closer, and this ever-changing circumstance must be taken into account if the reasoner is to successfully meet the deadline. This dissertation investigates the theoretical and practical problems in designing a fully deadline-coupled real-time planning and reasoning mechanism.

1.1 How to (plan to) meet a deadline between *Now* and *Then*: The problem and our approach

All agents, whether human or automated, that function in the real-world are subject to the fact that time is spent as their reasoning (deliberation) progresses. Deliberation is time consuming. Action occurs in the mere form of thinking or reasoning. In "Stop the world: I want to think" [PEDM], 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. We propose a formalism that treats thinking and acting uniformly with respect to the resources they consume.

Dean and Boddy [BD94] define *inferential* actions as computational procedures that have no effect on the world external to the agent, but affect the internal state of the agent. Inferential actions may provide the basis for selecting a physical action. Under time-pressure, an agent must allocate enough inference time to account for both inferential and physical actions. Agents invariably have bounded computational resources. In a dynamic environment, they run the risk that the assumptions on which their planning was based will change as they reason. New opportunities and options may become available while old ones are lost as the agent's deliberations progress. Design of a rational agent in a real environment poses several challenges that are newly recognized to be part of AI planning.

Some activities can be looked upon as *reactive* or reflex behaviors that require very little deliberation. Explicit execution-time reasoning is eliminated by compiling all the decisions about what to do in particular situations [AC87, Bro91, RK89]. Clearly, these have significant bearing on the survival and effective functioning of an agent in its environment, especially in a world where changes are rampant but regular. Recent radical research in AI has prompted the claim that reactive activity is the *sole* component of intelligent behavior, and can be engineered through a careful composition of very elementary, ready-to-use behavioral units [Bro91]. Although the debate as to whether the completely reactive non-deliberative behavior can live up to its stronger claims still goes on, we and others [DF89, Doy82, PR90, IG90] believe that deliberation is often the crucial component that highlights intelligent behavior in non-trivial problem solving.

1.1.1 Foreseen and unforeseen situations

We roughly characterize commonsense planning and acting problems into two categories. In one, which we call the category of *foreseen situations*, even though a potentially complex plan of action is required, the plan is *canned* and is readily available to

the agent. Here the *planning time* has no bearing on the actual use of the formulated solution. E.g., the belabored process of inventing an effective CPR (cardiopulmonary resuscitation) procedure is not of consequence while administering CPR to a motor accident victim in an emergency situation; only the amount of time required to enact the procedure is. Similarly, a fire-drill that has been rehearsed and mastered is available to a fireperson to execute in case of fire. In both of these, although significant time has gone into the synthesis of the plan of action, it is spent *a priori* and hence does not factor into the planning time.

On the other hand, in many situations plans are not readily available to be enacted. The agent must either formulate a fresh solution from scratch or combine and adapt existing solutions in challenging ways to solve the problem at hand. These we will call *unforeseen situations*. In these situations deliberation is essential in the formulation of a plan. Even in solutions that employ techniques such as CBR (case based reasoning) which rely on prior experiences, deliberation is required in the phases of retrieval, matching, and refitting of existing solutions to form a novel solution [Ham89]. In unforeseen situations, not all planning can be done a priori, and hence involves inevitable expenditure of time.

1.1.2 Deadlines

It is possible to characterize situations on the basis of the total time available to an agent to formulate and enact a solution. For some problems it is not enough to come up with a solution, or even the "best" solution. It is very important to come up with a solution *in time*. This brings us to the issue of deadlines. Deadline situations tilt the emphasis in problem solving from optimality towards feasibility. We find it useful to classify deadline situations along two orthogonal dimensions. Along the first dimension, deadlines can be classified as *hard* or *soft* In soft deadline situations, there are rewards to meeting the deadline. Should the deadline be overshot, the returns

diminish with the margin by which the deadline is overshot. However, all is not lost if the deadline is not met. An example of a soft deadline scenario is "getting to the concert by 5:00 p.m. to reserve the best seats". In contrast, hard deadline scenarios present the all or nothing case. The rewards drop to zero once the deadline is missed, therefore all efforts must concentrate on feasibility first, then optimality. What is not feasible is not worthwhile. An example of a hard deadline situation is a pilot of a helicopter planning to rescue a soldier who is stranded on enemy ground. Along a second dimension a deadline may be classified as being *non-extensible* or *extensible*. By an extensible deadline, we mean one which can be extended by means of efforts made by the agent. However, the agent has to plan to extend the deadline and, the time to formulate and execute this plan must factor into the total time towards the previously defined deadline. There is a possibility that the plan to extend the deadline may not succeed. The following is an example of an extensible deadline scenario: It is the night of July 30, and a student Paul has a deadline to submit an assignment by August 1. He realizes that there isn't enough time to finish the assignment and that he could do better given more time. He decides to make a plea to the professor for an extension on medical grounds. Paul must spend a significant amount of time on July 31, feigning an illness convincing enough for a doctor to give him a certificate when he visits him. This requires planning and execution of the "extend-deadline" plan. Should his plan succeed, Paul has a lot to gain. Should it not though, and the doctor sees through his attempts, valuable time is lost from working on the assignment.

1.1.3 When is a plan good enough?

Planning effort typically consists of method (appropriate operators or sequences thereof) selection, and subproblem combination. In fully deadline-coupled planning, all the planning effort must be in real-time and must be accounted for in terms of how long the clock ticks while the deliberations proceed. A second concern is how

to decide how much to plan so that the best plan may be found. Inferences need to be controlled, and monitored with respect to the deadline. What is meant by "the best plan" is different for different researchers engaged in controlling inference. Most decision-theoretic research is concerned with optimality with respect to a set of predefined criteria that are specific to the given problem domain. Given that deliberation is costly, in that it uses resources that could be put to use elsewhere, it then becomes necessary to map the cost of each unit of deliberation against the marginal benefit of thinking. We do not solve the problem of finding the "best" plan. Our effort is to give the inferential apparatus to a time-situated agent to enable her to reason in time about how long her reasoning takes, and to be able to make judgements on the feasibility of her efforts. We do not require that the agent should always succeed in finding a feasible plan to solve the problem within the deadline; rather, our concern is that that she should know whether or not she will be able to successfully reason about the fact as time progresses.

1.1.4 Nell, Dudley and the railroad tracks

In this dissertation, we focus on the problem of real-time, real-world planning in *unforeseen, fully deadline-coupled* situations. According to the above classification of deadlines, our problem concerns *hard, and non-extensible* deadlines for the most part. The deadline is hard in the sense that there is no merit to any solution that overshoots the deadline. The deadline is non-extensible, in that that the agent has no means to extend the deadline. Our solution has both reactive and deliberative components. An agent in the real-world constantly receives input from her environment in the form of observations. She must make timely changes to her belief set continuously. In this sense our agent is reactive. On the other hand, she does not have a prespecified plan of action for every stimulus from the environment and must therefore plan for her goals. We address theoretical and practical issues in developing a reasoning mechanism for

an automated agent in a hard and non-extensible deadline situation. The interesting scenarios are those in which the deadline is very tight, i.e. there is very little time for thought and action. We will present several variations of a paradigmatic hard deadline scenario to incrementally illustrate our approach and solutions. The dramatic life-and-death scenario chosen is the following: *Nell & Dudley and the railroad tracks.*[1] Little Nell is tied to the railroad tracks and the agent Dudley must figure out and enact a plan to save her in time before an oncoming train approaches. This is an unforeseen situation, and if he 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 reason in time about his own reasoning in time.

1.1.5 Active logics

The platform that we have chosen for our work is called *active logics*. Active logics are generalizations of *step-logics* which were introduced by Elgot-Drapkin and Perlis [EDP90]. A step-logic is an inference mechanism situated in time which allows the modeling of the agent's reasoning process as it unfolds, one-step-at-a-time. Long chains of reasoning take as much time as the cumulative sum of individual reasoning steps that form the chain, and this fact is *represented* in active logic. Each theorem is marked uniquely by a time-stamp. The logic thus effectively represents what has been

[1]This problem was first mentioned in the context of time-dependent reasoning by McDermott [McD78], and more recently discussed in [CL90].

proven up to any given point in time, as opposed to traditional logics which focus only on a limiting (cumulative) theorem set. More details and features of step-logics that guided our choice are described in Chapter 2.

1.1.6 The frame problem

Inherently, the planning problem has been demonstrated to be computationally intractable [Cha87]. even in very simple domains, and under simplifying assumptions such as the *static world assumption* [AHT90], which forbids simultaneous actions, or erratic random changes during the course of planning or execution. With the added constraints we have imposed on our problem characterization, we have an even more difficult problem at hand. We have selected a largely declarative logic-based framework for our purpose. We have selected the logic approach (to be elaborated in section 2.3). Frame problems (temporal projection, qualification and ramification problems) arise in any formal framework, and have generated a controversy over the relative merits of a declarative mechanism. A planner may not have complete knowledge about the conditions and effects of its actions. A problem that has assumed canonical status in this regard is the Yale Shooting problem (YSP). We will use the YSP (real-time versions of it) as a yardstick to demonstrate solutions to the above frame problems. In contrast to procedural approaches, we regard this to be a satisfying feature of the temporal reasoning rules developed for our fully deadline-coupled planning mechanism.

1.1.7 Space and computation bounds

Time is a critical resource, and our formalism is aimed largely at addressing time-related issues. In a realistic setting, Dudley must also measure up to two other crucial resource limitations as well, namely, space and computation bounds. Step-logics originally suffered from two crucial drawbacks. As theorems are proven, the belief set, although still finite, can grow very rapidly, making it unrealistic for the inference en-

gine to derive all possible new conclusions in *parallel*. We describe these concerns and offer some solutions by introducing a limited-memory architecture and a limited inference-capacity engine into the active-logic framework.

1.1.8 A modal semantics for active logics

Most formal approaches do not have an appropriate representational framework to tackle time-situated reasoning problems such as the above. They assume that an agent is able to reason forever in a timeless present as if the world had stopped for the agent's benefit. Resource limitations have been of some concern in formal work. In particular, the problem of *logical omniscience* has received attention in the epistemic logic literature. It concerns the difficulty with the classical Hintikka possible world semantics [Hin62] that the agent always knows the logical consequences of her beliefs. However, no existing works have provided a semantics that can address the issue of how the reasoning progresses vis a vis the passage of time. Although work in temporal logic involves reasoning *about* time (e.g., [All84, McD82, MS87]), time is not treated as a crucial resource that must be carefully rationed by the agent as it is being spent in every step of reasoning. In [ED88a], a very limited first attempt was made at providing a semantics associated with step-logics by defining the notion of a step-model. Although active logics have since been characterized and implemented, no further attempts have been made to give a formal semantics for the step-like reasoning process. In the final part of this dissertation, we provide a modal semantics for active-logics that is intended to bridge the gap between previous modal approaches to knowledge and belief, and time-situated frameworks such as step-logics which have a means for attributing time to the reasoning process.

1.2 Historical overview

Research in planning has seen a dramatic shift of focus in the recent years. Traditional planning systems assumed the availability of complete knowledge at planning time, so that once a plan was generated by the planner, it was assumed that it would be carried out successfully by the executor. The shift has been motivated by the realization that planners that rely on the traditional assumptions of complete knowledge perform very poorly in significantly large or complex domains, in dynamic environments, or under time-pressure and other resource limitations. AI research in planning has progressed from static well-defined domains to dynamic, unpredictable domains, planning and acting are closely coupled and interleaved, and increasing efforts have been invested in designing agents who can reason about their own resource limitations in the course of their planning.

[AHT90] offers a comprehensive overview of planning techniques and the recent shift of interest towards more dynamic, complex and situated domains. Of interest to us are planning formalisms that incorporate (a) representation and reasoning about time, causality, and intentions (b) uncertainty of plan execution and (c) sensation and perception of the real-world. [AHT90] also gives a chronology of planning systems dating from 1960 thru 1990. They provide useful classification of planners based on how they handle goal ordering and interactions, abstraction, search, and on how they combine planning and execution.

To give a glimpse of the evolution of planning systems, we mention a few classical systems here:

- STRIPS [FN71] was the earliest and the most influential planning system. Its representation of world model and plan steps in terms of a set of operators and add-delete lists has influenced the knowledge representation of our planner, e.g., in the use of action triplets in plans.

- NOAH (Nets of Action Hierarchies) [Sac75] reasoned about the intrinsic non-linearity of plan steps and some schemes in which the temporal ordering as well as conditional iterations and loops can be represented. NOAH's procedural nets represented meta-planning knowledge. This task-specific information was written in procedural form in SOUP (Semantics of User's Problem). It used "critics" which are outside advisors that performed decision making regarding non-linearity and plan optimization.

- NONLIN [Tat77a] introduced the notion of goal structure and the use of typed preconditions.

- DEVISER [Ver83] was an extension of NONLIN that handled reasoning *about* time and events.

- MOLGEN [Ste81] introduced the least commitment principle. They introduced constraint formulation and propagation techniques in planning.

- NASL [McD82] was the first approach to planning that relaxed the "static world assumption" and allowed for execution of each step as it was generated in a (partial) plan. This made the planner more "reactive" but less likely to reason about the effects of immediate action. (Our work in active-logic planning is close to NASL in the way it advocated planning and acting as two equally important aspects of problem solving.)

- PMM [HR79] provided an opportunistic planning framework in a blackboard architecture. They introduced the concept of cognitive specialists which are *external* to the planner and which influence planning decisions; They showed how to utilize measures of worth of invoking these specialists.

Our work borrows from these and other early planning systems. Our chief contribution is to embody planning and meta-planning within a uniform framework, allowing

a fully time-situated treatment where all the time spent in planning and acting is fully accounted for by the automated planner itself.

Up until mid 1980's planning systems revolved around the basic model to plan in advance (off-line) and then *prove* that the plan would work. Then, real-world automated planning needs of urgency, uncertainty and unknowns began to be studied.

Distinct viewpoints exist on ways that deliberation and action may be combined in planning. These fall into uniform and layered architectures. The former uses a single representation and control structure for both action and deliberation (our work is in this category) while the latter uses different algorithms to implement functions at various layers.

Our work overlaps with various other research efforts: those that deal with newer aspects of planning and acting (reactive, deliberative, deadline-driven); those that address temporal projection and other frame issues in formal work; and those that address logical omniscience and bounded rationality. A description of related work appears at the end of the relevant chapters of the dissertation.

1.3 Research contributions

This dissertation reports work in fully deadline-coupled planning and reasoning. The following mark the milestones in this research:

- **An active logic for fully deadline-coupled planning and reasoning**
  - Revision and extension of the original step-logic framework for deadline-coupled planning.
  - A knowledge representation for actions, plans, contexts and temporal projections and time estimates.
  - Design of an inference engine with domain-independent rules for time-management, planning and decision making.

  - A real-time truth maintenance mechanism that detects contradictions within planning contexts and induces time-situated corrective action.

- **A time-situated treatment of frame issues**
  - Addressing the temporal projection problem.
  - Addressing the qualification and ramification problems.
  - Solution to real-time versions of the Yale Shooting Problem.
- **An active logic with space and computation constraints**
  - Design of a model of reasoning with limited memory.
  - Modification of the inference engine to restrict parallelism.
  - Proof of the adequacy of the limited memory model.
- **Development of a modal semantics for active logics**
  - Addressing the problem of logical omniscience.
  - Soundness and completeness of the modal active logic for thinking in time.
- **Implementation of some versions of the inference engine in Prolog**

1.4 Outline

This dissertation report is organized as follows. Chapter 2 is an introduction to step-logics and active logics. It describes unique features of these logics that make them suitable for fully deadline-coupled planning and reasoning. Chapter 3 gives details of the knowledge representation of the deadline-coupled active-logic mechanism. Chapter 4 describes the rules for temporal inferencing. Chapter 5 shows interesting applications of the temporal reasoning to various versions of the Yale Shooting problem and provides solutions to aspects of the frame problem in the real-time setting. Chapter 6

describes the inside workings of the inference engine. Chapter 7 discusses the estimation of time and how it can be conjectured for various action types. Chapter 8 shows how the active logic can be improved to address concerns of space and computation bounds. Lastly, Chapter 9 offers a semantics for a modal active logic that is inspired by the active logic for planning and shows that it is sound and complete. Chapter 10 lists the conclusions and suggests directions for future work.

Chapter 2

The active logic platform

Our work in deadline coupled planning has been part of a larger study of of time-situated reasoning undertaken at the Univ. of Maryland since 1986. It uses and extends the research involving special formalisms, jointly grouped under the name *active logics* which all share the basic property of time-situatedness. They are logics that treat time not merely as an external entity to be reasoned *about*, but rather as a feature guiding the inferences within. They are inspired by the view of logic as having an evolving *life* (see [Nil83, Tho90]), in which theorems are proven, and sometimes later disbelieved, as it interacts with its environment. They combine procedural and declarative methods, since, while the actual workings are implementation dependent, the behavior is nevertheless formal in many respects: it is largely governed by explicit declarative axioms and rules of inference.

Early work in active logics began with the development of a theoretical tool called the step-logics. Step-logics [ED88b, EDP90, ED91] were introduced to model a commonsense agent's ongoing process of reasoning in a changing world[1]. In the next section we briefly describe step-logics, their structure and intent. In Section 2.2 we elaborate on desirable features of active-logics that prompted us to for our particular

[1]Step-logics have also been used for multi-agent coordination without communication using focal points [KR92], and for the introduction of new expressions into the language over time [Mil92].

$$
\begin{array}{lccc}
0: & & \emptyset & \\
\vdots & & & \\
i: & \ldots & \alpha & \ldots \\
i+1: & \ldots & \alpha \rightarrow \beta, \beta \rightarrow \gamma & \ldots \\
i+2: & \ldots & \beta & \ldots \\
i+3: & \ldots & \gamma & \ldots \\
\vdots & & &
\end{array}
$$

Figure 2.1: Step-logic studies

problem. In Section 2.3 we comment on the relative merits of a declarative framework, and inspect active logics for deadline-coupled planning in this light.

2.1 Step-logics: An inference mechanism

Traditional logics are *static* entities, in that they represent the world from a timeless present and have the property of *logical omniscience*, that is they are the deductive closure of all theorems every proven by the agent. Step-logics [ED88a, EDP90] were introduced to battle these two evils, in an effort to model a commonsense reasoner with limited reasoning capability.

A step-logic is characterized by a language, observations and inference rules. A *step* is defined as a fundamental unit of inference time. Beliefs are parameterized 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 expression $Now(i)$, indicating the time is now i. Observations are inputs from the external world, and may arise at any step i. When an observation appears, it is considered a belief in the same time-step. Each step of reasoning advances i by 1. At each new step i, the only information available to the

agent upon which to base his further reasoning is a snap-shot of his deduction process completed up to and including step $i-1$. Figure 2.1, adapted from [ED88a] illustrates three steps in a step-logic with Modus Ponens ($\frac{i:\alpha, \alpha \rightarrow \beta}{i+1:\beta}$) as one of its inference rules.

An agent starts out with a finite (possibly empty) set of beliefs at time step 0. Step-logics are deterministic in that at each subsequent step i, all possible conclusions that result from one application of all rules of inference applied to the previous step are drawn. Thus the set of beliefs is always finite[2].

Three mechanisms that are possible aspects of an agent-theory: self-knowledge (S), time (T) and retraction (R) were proposed as the basis of developing an array of eight step-logics. They were arranged in increasing sophistication with respect to S, T and R : SL_0 : none; SL_1 : S; SL_2 : T; SL_3 : R; SL_4 : S,R; SL_5 : S,T; SL_6 : R,T; and SL_7 : S,T,R. A given step-logic is characterized by its own inference and observation functions and a language. We note here that the finiteness attribute of step-logics is what renders a computationally decidable treatment of the self-knowledge feature possible for those logics that have it. Pictorially, a step of reasoning, which can be regarded as a snap shot of the reasoning process thus far, is shown as :

$$\mathbf{i} : \alpha, \beta, \ldots$$

The i stands for the step number, and the beliefs α, β, ... are held at step i. The ellipsis indicate finitely many other beliefs.

2.1.1 Time stamps on inferences

In step-logics there is the important distinction of capturing *when* a theorem was proven. Thus there is the notion of *i-theoremhood* for every step i. Proof of an

[2]The original step-logic mechanism had no means however, to keep this set quite small, as large number of conclusions were added to the belief set at each step from all possible inferences, rendering the belief set computationally intimidating, though still finite. We address this drawback and suggest solutions in chapter 8.

i-theorem is based on the inference rules that characterize the particular step-logic. Beliefs are not automatically *inherited* from one step to the next. Like all other *thinking* actions, inheritance captures the latent capacity for a reasoner to decide which of her current beliefs she wants to continue to hold at the next time step. It then seems plausible that a careful monitoring of what will be inherited to the next time-step may serve as a commonsense mechanism to dispel or distrust some current beliefs on the basis of *what else has come to be believed.* A rational agent may not want to continue to believe α and $\neg\alpha$ when they are seen to appear as a direct contradiction. Contradictions and their handling are an important aspect of limited reasoning. In the inference engine that has been built for deadline-coupled planning contradictory beliefs are identified within the context of a particular partial plan. Contradictory beliefs among different contexts are not a cause for alarm. They are given special treatment that allows for a real-time truth maintenance to spring into action to resolve the contradiction and allow its scope to be restricted in future theorem-proving.

We follow the general rule schema (RS) defined in [Mil92] which represents the structure of an inference rule for active logics:

$$\textbf{RS:} \qquad \begin{array}{ll} \mathbf{i-j}: & \alpha_{i-j_1}, \ldots, \alpha_{i-j_m} \\ \vdots & \vdots \\ \mathbf{i}: & \alpha_{i_1}, \ldots, \alpha_{i_n} \\ \hline \mathbf{i+1}: & \beta_1, \ldots, \beta_p \end{array}$$

where $i, j \in \mathbf{N}$ and $(i-j) \geq 0$. The schema **RS** captures the idea that at any step of the reasoning process the inference of β_1 through β_p as $(i+1)$-theorems is mandated when all of α_{i-j_1} through α_{i-j_m} are $(i-j)$-theorems, and all of α_{i-j+1_1} through α_{i-j+1_r} are $(i-j+1)$-theorems, ..., and all of α_{i_1} through α_{i_n} are i-theorems.

A particular special instance of this schema is a class of inference rules is obtained when $j = 0$. This corresponds to those rules for which only beliefs at the previous step (step i instead of steps i thru $i - j$) are used to make inferences in step $i + 1$. This useful form is used in all the inference rules in the step-logics of Elgot-Drapkin. Below, we describe three rules of the original step-logic in this form which we have included in our active logic. The rule of observation is modified slightly in the active logic for planning. As described in chapter 3 to follow, Dudley maintains a context of each plan being developed. We restrict the reasoning to each context. Since observations concern the environment, they enter the context of each plan (including a special structure called the null plan) at the step during which they are made.

2.1.2 Making note of what time it is

This rule is intended to let an agent know what the current time is, at a particular step in its reasoning. The current time is represented by a special predicate in the language: **Now**(i) denotes the belief that "it is now time i." This indexical knowledge essentially must change with the ticking of a clock. Thus, the belief **Now**(i) is one belief that is *never* inherited from one step to the next.[3]

AGENT LOOKS AT CLOCK

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

[3]It is possible to think of an agent who is not *always* aware of what time it is. It is easy to write an inference rule whereby the agent either periodically or randomly checks the clock to check the time. However, as described in Chapter 3, this may make the process of time stamping the other beliefs with the "now" value rather involved. The inference rules are constructed in such a fashion that using the belief regarding the current time, the agent can keep track of the latest belief in a chain of planning to be used in subsequent reasoning.

2.1.3 Inheritance

Another rule, called INH for *inheritance*, addresses the issue of theoremhood persisting from one step to the next. INH mandates the appearance (i.e., inference) of a wff α at step $i+1$, if α is an i-theorem.

INHERITANCE (INH)

$$\frac{\mathbf{i}: \alpha}{\mathbf{i+1}: \alpha}\ ^{\dagger}$$

†If α is not $Now(i)$

In Chapter 5 we will present temporal inferencing. We do encounter contradictions *within* a context set, and have devised a special rule for carrying over formulas in a context set over to the next step. However, we still inherit the entire formula $CS(i, p, \ldots)$ to the next step, while using the special rules to decide the formula $CS(i+1, p, \ldots)$ to be proven at step $i+1$. The inheritance rule allows us to capture the entire history of the reasoning thus far. The time step numbers that appear within the formulas on the other hand are the time-stamps that mark when the particular theorem came to be proven. Active logics are unique in this respect.

2.1.4 Observations

The capacity to observe, and thereby to incorporate changes in its environment into the reasoning account for the agent's situatedness in the real world. Observations are acquired through the following inference rule and represent extra-logical axioms or facts presented to the agent.[4]

[4]The way observations are handled in the active logic work thus far is by defining a function $OBS : \mathcal{N} \rightarrow 2^{\mathcal{L}}$. It is not reasonable to assume that the sensory capabilities of the agent can automatically acquire sufficient information about the world. The cognitive and physical capabilities of the agent must be used to augment perceptions. The process of directing action to aid perception is quite complex and we will not offer solutions to it. In some examples later, we hint at a mechanism whereby an agent may plan to make a particular observation upon which future planning is dependent.

OBSERVATION (OBS)

$$\frac{\mathbf{i}: \quad}{\mathbf{i+1}: \ \alpha}^{\ddagger}$$

‡If $\alpha \in OBS(i+1)$

2.2 Unique features of active logics

Step-logics provide a way to reason about time in time. As a natural extension, Elgot-Drapkin proposed a direction for future work where this very accountability of deliberation time would be an indispensable feature. That topic is one of deadline situations. Active logics are particularly attractive as a time-situated mechanism for dealing with deadline situations in a changing world. It is a formal mechanism that is designed to account for *all* the time in an agent's reasoning, without getting into the infinite regress of meta-meta-meta ... levels of reasoning. In this section we describe features of active logics that relate and contrast it to conventional commonsense reasoning systems.

Capturing reasoning-time as an action: Reasoning actions occur concurrently with other physical actions of the agent and with the ticking of a clock. The agent can not only keep track of the approaching deadline as he enacts his plan, but can treat other facets of planning (including plan formulation and its simultaneous or subsequent execution and feasibility analysis) as deadline-coupled. Related to this feature of active-logics is the fact that there is no longer a one final theorem set. Rather, theorems (beliefs) are proven (believed) at certain times and sometimes no longer believed at later times. Provability is time-relative and best thought of in terms of the agent's ongoing lifetime of changing views of the world. This leads to the issue of contradictions below.

It involves detecting missing information, performing planning to obtain the required knowledge and then proceed with the goal.

Capacity to interact with the environment: Active logic reasoning is situated in time by its very definition. An additional feature is that the agent can also be embedded in its environment. The capacity to take note of changes in the environment, to get feedback on its own actions, or input from the results of other agents' actions allows the agent to adjust its reasoning to unprecedented observations.

Handling contradictions: An agent reasoning with active logic is not omniscient, i.e., his conclusions are not the logical closure of his knowledge at any instant, but rather only those consequences that he has been actually able to draw.[5] Also, since commonsense agents have a multitude of defeasible beliefs, they often encounter contradictions as more knowledge is obtained and default assumptions have to be withdrawn. While a contradiction completely throws an omniscient agent off track (the swamping problem), the active-logic reasoner is not so affected. The agent only has a finite set of conclusions from his past computation, hence contradictions may be detected and resolved in the course of further reasoning.

Nonmonotonicity: Active logics are inherently nonmonotonic, in that further reasoning always leads to retraction of some prior beliefs. The most obvious one is $Now(i)$, which is believed at step i but not at $i+1$. The nonmonotonic behavior enables the frame-default reasoning that the commonsense agent must be capable of [MH69].

2.3 Merits of a (largely) declarative framework

In recent years real-time planning systems have largely been geared towards robot-planning. For most of these systems the goal is not so much to model human commonsense behavior in real-time but to come up with an effective program that will perform certain tasks in the real-time domain. These approaches are largely proce-

[5] Konolige [Kon86a], Levesque [Lev84] and Fagin and Halpern [FH88] proposed systems in which the agents are not omniscient. However, the inference time is not explicitly captured in their systems.

dural. Implementation of procedural methods tends to be much more time efficient while the logic based inference engine often shows poor performance speeds. But, the use of logic provides a unifying framework for the system that is hard to obtain in unstructured hybrid designs.

To reason *about* the plan is to make meta-level decisions about the structures in the knowledge base. For this, the KR must be powerful enough to express the complexities of the reasoning, while remaining simple enough to be the object of reasoning.

Procedural knowledge makes it extremely difficult to capture generalizations or domain-independent abstractions. They tend to prevent the same piece of knowledge from entering different computations. We believe that unlike completely procedural approaches, we have a principled way of incorporating the issues in deadline coupled planning into the knowledge representation. One principled feature is the notion of the clock. Our formal treatment of time makes the KR issues challenging but interesting because of the novel capability to reason about the reasoning process itself. One advantage of having as much as possible in declarative form instead of control procedures is the possibility of dynamically changing the parameters or the inference rules either as a result of learning or as a function of the context of reasoning. We note that humans often regard some parameters as dynamic in their reasoning; e.g., people ask for paper and pencil or seek help from other persons or storage aids when it appears that a particular problem has a higher memory requirement and can not be "solved in one's head". In this case, the size of the short term memory is a parameter that if expressed declaratively instead of being hardwired into the control structure, allows more flexibility. In Chapter 8 we suggest the size of the memory as a parameter that is represented as a belief in the declarative framework.

The choice of the declarative formalism allows the language to serve as a representational scheme, and the logical deduction to serve as the paradigm for the inference engine.

Among the advantages of a declarative framework that uses logic as the main knowledge representation mechanism is the hope of coming up with a formal semantics that gives precise descriptions of the meaning of expressions. This is a nontrivial task for active logics and we will describe our successful efforts in that direction in Chapter 9.

Lastly, we remark that the declarative framework has been an asset to us while incrementally designing the active-logic deadline-coupled planning mechanism. We have attempted to provide commonsense inference rules inspired largely by human behavior for formulating plans in deadline situations, for making decisions, as well as for drawing default conclusions and later revising them in time. The design has had a particularly nice additive property, in that more capability could be added to the inference engine without having to rewrite the simpler inference rules. We observed a natural modularity which we think can be attributed at least partially to the declarative knowledge representation.[6]

2.4 Summary and related work

In this chapter we gave a brief introduction to step-logics, the original instances of active logics.

Turing's original proposal for a definition of intelligence was associated with the notion of a continuous *temporal* property of an agent (the famous Turing test challenges the automated agent to withstand a process of interrogation and emerge indistinguishable from a human). Russell and Wefald [RW91] define a *limited rational agent*. SOAR [LNR87] is the most ambitious attempt to build an intelligent agent architecture. Kurt Konolige [Kon86a] argues that the property that marks a situated agent who reasons

[6] Curiously, analogous findings are reported in [Bro86] regarding a totally non-declarative design methodology. In the *subsumption architecture*, non-declarative complex behaviors are added on to simpler ones.

about states of the world and about the effects of its own actions is that the agent draws conclusions from an initial set of beliefs but he/she does not necessarily derive all the logically possible ones.

The above research is representative of what that can be grouped under the "bounded rationality" [Sim82] approach to commonsense reasoning. Simon's hypothesis is that there is no perfect deliberation. Organisms adapt well enough to *satisfice*; they do not in general *optimize*. Deduction models challenge the hypothesis of *logical omniscience* in various ways. Konolige [Kon86a] suggests a deduction model that is deductively closed under a set of rules, though not necessarily *consequentially* closed, since the rules may not be logically complete. The assumption of deductive closure greatly simplifies the technical problems by disregarding any particular control strategy. It is suggested that systems that are not deductively closed can be modeled by employing a *low cost bound* on derivations, thereby deriving only those proof trees whose depth is bounded. This extension is not part of the model. The model, however, ignores the time element present in the inferencing, assuming that the agent can perform the necessary computations in a time interval which is relatively short with respect to its ability to act. In contrast, we are concerned with bounded rationality of a time-situated nature.

In Chapter 9 we give detailed comparison of our work with other works ([FH88, Lev84] etc.) that attempt to circumvent the logical omniscience problem. Most of these formal approaches are still constrained to a *static* model of reasoning.

Step-logics have a place in this spectrum of research towards the design of rational agents that are constrained by various resource limitations. They model a resource-limited agent's ongoing process of reasoning.

Chapter 3

Knowledge representation

Fundamental to reasoning is the representation of knowledge; how the information is symbolized and manipulated. In this chapter we provide a brief description of the knowledge representation for our deadline-coupled planning mechanism based on active logics. No sharp definition of active logics will be given, but they include step-logics described in Chapter 2, as well as the active logic here. They all involve beliefs coming and going as part of the time-situated inference. The active logic that we develop here (we will call it *PAL* for *planning active logic*) is characterized by a language $\mathcal{L}$, an observation function *OBS*, and a set of inference rules *INF*. According to the characterization of step-logics described in 2, this active logic falls in the SL_6 category, since it embodies mechanisms relating to time and retraction. The retraction mechanism for this active logic is different from the one suggested in the step-logic work. We prefer to call it *internal retraction*. It is designed to affect the list of formulas within the predicate that describes a context set (to be elaborated below). Thus, the entire belief is not retracted but some formulas from the list within a predicate are retracted.[1] The inference engine contains rules that are domain-independent, i.e. are applicable to any instance of a planning problem, but are specific to planning and acting in deadline situations. Axioms, which are fed into the system before the

[1] Retraction is implemented as lack of inheritance.

starting time 0 capture the domain specific knowledge that is required for the particular solution.

The formal treatment is general, employing the Nell and Dudley scenario only in the minor domain-dependent details of the axioms used, so that the design may be tested on a specific scenario. We will repeatedly refer to "Dudley" as the generic automated agent.

3.1 The language $\mathcal{L}$

The language $\mathcal{L}$ is a first-order language. There are many special predicates corresponding to the logical components of the planning system. Terms expressing actions or fluents appear inside the scope of other predicates.[2]

3.1.1 Actions and fluents with time arguments

The traditional approach to planning has been to consider the world to be in a particular *state* or *situation* which is described by a set of formulas which are true in it. We have a dynamic representation of the world as an evolving theory that changes with changing time. The step-numbers serve as natural indices to represent a model of the world at any given point in time. In contrast to the situation calculus formalism, the world changes state with the passage of time, the occurrence of an event or the execution of an action is not required to cause the state change. The minimal change occuring in the belief set is the presence of a new **Now** value at every time step. Thus, there is no inherent difficulty in representing *simultaneous* events or actions.

[2]Terms containing time interval arguments appear in lists denoted by C_A, R_A within the action triplets, and also in beliefs **CS**, **Proj** and **Ppl**. A detailed description of these follows. There is a predicate symbol for each action, event or fluent in the language. They are in fact treated as if they are "quoted". We omit the quotes to keep the long strings readable. Thus the beliefs of the agent that we will describe shortly are still first order formulas.

In particular, every event or action is concurrent with the ticking of a clock. Events and actions correspond to the common sense notion of *happenings* in the world of the agent. Examples of event are $Fall(5, book, floor)$, $Close(10, jack, door)$ meant to represent the happening that "the book fell on the floor at time 5" and "Jack closed the door at time 10", respectively. Actions are special events that require an agent to *perform* them. The latter of the two events above, namely $Close(10, jack, door)$ is an action.

An agent usually has knowledge about certain *properties* that hold of objects or world states. For example, $On(7, Block\_A, Block\_B)$ in the blocks world describes the property of "On-ness". $Color\_Of\_Eyes(10, nell, blue)$ describes the property of "eye-color". These we call as fluents. Fluents usually tend to persist unless specific events or actions occur that result in change.

We will represent an action, event or a fluent in our language by a formula $X(S : F, Args)$ which consists of a predicate name X, a time argument and a list of other arguments that may denote other properties, including space or agent names. The time argument is a time interval $S : F$ over which it holds, S and F being the start and finish points of the interval. The other arguments follow and are denoted by $Args$ for easy reference. An example of a formula is $At(1 : 5, dudley, home)$. Using the standard Prolog convention, capital letters will be used to denote variable names and small letters used to denote constants or ground atoms. We often wish to express formulas that hold only over the duration of their interval $S : F$ and do not continue to hold beyond F by persistence. Most of the agent actions (e.g., Run, $Shoot$) fall in this category. By $S : \overline{F}$ we denote the time intervals in formulas with these actions. We use the shorthand S for $S : S$ to denote a point time interval. These formulas for actions, events and fluents which have predicate symbols are used in "quoted" form when they appear inside the predicates for context sets, projections and plans, so as to still maintain a first order language for the agent's beliefs. We do not explicitly

show the quotes in the text.

Further, we have four different forms of formulas. $X(S:T, Args)$ denotes that X holds over interval $S:T$, $\neg X(S:T, Args)$ denotes that $\neg X$ holds over $S:T$. Further, $X_c(S:T, Args)$ (resp., $\neg X_c(S:T, Args)$) are used to denote that not only does the agent believe in X (resp., $\neg X$) over the interval, but that the agent has reason to believe that the time point S could be a possible point of change of the fluent from $\neg X$ to X (resp., from X to $\neg X$) in the event that $\neg X$ (resp., X) holds in the interval ending in S. For example, $On(2:4, floor, ball)$ should be read as the ball was on the floor in the interval $2:4$ and $On_c(2:4, floor, ball)$ is read the ball was on the floor over the interval $2:4$ and probably wasn't there before 2.

Either of $\{X(S:F, Args), X_c(S:F, Args)\}$ are defined to be in *direct contradiction* with either of $\{\neg X(S:F, Args), \neg X_c(S:F, Args)\}$. A *uniqueness contradiction* exists between formulas $X(S:F, Args1, U, Args2)$ and $X(S:F, Args1, V, Args2)$ if $X(S:F, Args1, U, Args2) \rightarrow \neg X(S:F, Args1, V, Args2)$ whenever $U \neq V$. E.g., there is a uniqueness contradiction between $At(5, Dudley, home)$ and $At(5, Dudley, railroad)$. A formula α *du-contradicts* a formula δ, if it is in direct or uniqueness contradiction with δ. The same definitions of contradiction extend to X_c and the negated versions.

Formulas may enter the agent's belief set either by means of an observation or as the result of a deduction. If a formula α resulted from an inference rule whose antecedents were $\beta_1, \ldots, \beta_n$, then a *derivation* of α is the set S containing all of $\beta_1, \ldots, \beta_n$, and each S_j in the derivation of β_j. It turns out that we are interested only in the *default* formulas (obtained thru temporal projection inference rule) that are used in deriving α. Default formulas will be defined later in this chapter. Whenever we wish to call attention to these default formulas, we use annotated formulas such as $X(S:F, Args)[\beta_{j_1}, \ldots, \beta_{j_k}]$ to denote that the proof of α contains the default formulas $\beta_{j_1}, \ldots, \beta_{j_k}$. Such an annotated formula has the status of a default and is as feasible as the weakest default in its annotation as explained in Sections 4.1 and 4.3.

3.1.2 Action triplets

An action triplet denoted by $[C_A, A, R_A]$ consists of an action formula A preceded and followed, respectively, by a list C_A of *conditions* C_A and a list R_A of *results*. A is a formula with an action. A condition or a result is a formula containing a fluent. The conditions may need to be true over all or some of the duration of the action. An action may be complex or primitive (atomic). A primitive action takes one time step to perform. Various action types and their representations are described in 7.2.

3.2 Nucleus of the temporal reasoning

In this section we outline the structure of some key beliefs and the predicates used to represent them.

3.2.1 Partial plans

A partial plan p is a ternary predicate **Ppl**. Thus

$$i: \ \mathbf{Ppl}(i, p, Triplet\_List)$$

denotes a partial plan being developed at step i with the name p. The $Triplet\_List$ is an ordered list of action triplets. We will sometimes use $\mathbf{Ppl_{i,p}}$ to refer to the $Triplet\_List$ with respect to i and p.

A special plan with the name *null* is a plan with no actions in it. A predicate $\mathbf{Ppl}(i, null, \ldots)$ describes a null plan, and is concurrently manipulated with those for the other partial plans. It is the only partial plan predicate when when no planning is in progress. We use the null plan to maintain a thread of reasoning in which all the theorems that are proven have either observations or axioms as their premises. This is useful when we think of Dudley as a witness or passive observer in a certain scenario who simply makes mental note of the observations and draws conclusions from them, without planning to perform any actions of his own.

3.2.2 Contexts

Dudley simultaneously develops alternative plans towards attaining his goals or subgoals. Each of these partial plans (including the null plan) defines a context within which reasoning can be done about the expected state of the world if the plan were to be carried to completion. In the context of a particular partial plan, the actions in the plan and their effects and extended effects can be treated as facts as far as the state of the world resulting in successful execution of the plan is concerned. In the next chapter we will describe an inference rule (similar to Modus Ponens) that allows for inferences to be made within a particular context. The context also serves as a basis for applying temporal projection to obtain default formulas.

The ternary predicate $\mathbf{CS}(i, p, Context\_List)$ denotes the **Context_set** for a plan p at step i. The list $Context\_List$ consists of quoted formulas (we omit the quotes for readability), and includes all of the facts (observations)[3], formulas corresponding to actions in the plan and formulas that the agent deduces to be true in the state of the world resulting from the successful execution of plan p. We will often use $\mathbf{CS_{i,p}}$ to denote the list $Context\_List$. All formulas corresponding to a given predicate name X are kept sorted in the $Context\_List$ with the time interval $S : F$ of each formula as the key. The context set changes with time as the plan undergoes modification and as inferences are made in the context of the plan.

3.2.3 The predicate that keeps track of time

Now(i) denotes Dudley's belief that the time is currently i. This is one belief that is never carried over to the next time step. The Now predicate serves to couple the step number with the index used in a predicate to decide which predicates will be used in

[3]Actually it only consists of the subset of facts that is relevant to the particular partial plan. Chapter 8 deals with space bounds on the reasoning and proposes a relevance mechanism to keep the reasoning directed to a particular partial plan for a duration of time.

the inferencing at the current step.

For example consider the step 5:

$$\mathbf{5} : \mathbf{Proj}(5, p, \{\ldots\}), \mathbf{Proj}(4, p, \{\ldots\}), \mathbf{Now}(\mathbf{5}), \ldots$$

At step 5, the rule that specifies both $\mathbf{Now(i)}$ and $\mathbf{Proj}(i, p, \{\ldots\})$ in its antecedent will only pick $\mathbf{Proj}(5, p, \{\ldots\})$ to manipulate, the other formula with the earlier time stamp (step 4) will be inherited and will contribute to the history of the reasoning. The way most of the inference rules are designed, they pick the latest belief. In case there can not be an exact match between the step number and the time stamp on a predicate the latest will have to be explicitly chosen if so desired.

3.2.4 Projections

Firing of an inference rule corresponds to a *think* action. Dudley's non-defeasible beliefs are treated as *facts*[4]. Observations incorporate into beliefs in the same time step. Theorems whose premises consist of facts alone are also regarded as facts.

At each step i, the ternary predicate $\mathbf{Proj}(i, p, Proj\_List)$ denotes the projection that is formed in the context of each partial plan p that is in progress, based on the default of persistence[5]. The i denotes the step number, and $Proj\_List$ is a list of quoted formulas. We will often use $\mathbf{Proj_{i,p}}$ to denote the list $Proj\_List$ with respect to i and p. The formulas in the $Proj\_List$ are derived by the application of the temporal projection rule 4.1 described in chapter 4. They are not observations, and have a default status among the formulas in Dudley's beliefs in the context in which they appear. If, in the course of Dudley's reasoning, as time progresses, a projection

[4] Strictly speaking though, the agent only has beliefs, never facts, since even observations are not etched in stone, and may very well change over time. In all the problems that we tackle though, we will treat observations and facts synonymously.

[5] Projections (and persistences) have been studied by numerous authors; see e.g., [Wil83a, Kau86, McD87]. Our treatment is along the lines of time-maps of [DM87].

is contradicted by later observations, or by theorems whose premises do not contain defaults, it must be disbelieved, and so must any formula that used it as a premise.

3.2.5 Working estimate of time and feasibility

The belief **WET**(i, p, N) denotes the *working estimate of time* for the plan p computed as of step i of reasoning is the integer N. WET computation is revised at each step by an inference rule and the feasibility of the plan p is continuously checked by making sure that the sum of N and i does not exceed the deadline. The binary predicate **Feasible**(i, p) denotes the belief that the plan p is feasible as of step i.

3.2.6 A deadline-coupled goal

The belief **Goal**(i, G, D) is maintained to denote that Dudley has a belief to meet a goal G by a deadline D. In all the examples pertaining to the Nell and Dudley scenario, we have a single goal "to save Nell" which is coupled to the deadline that Dudley has previously computed (perhaps from the expected time of arrival of the train, extrapolated from its current position and speed).[6] The binary predicate **Unsolved**(i, G) indicates that the goal G is unsolved as of step i.

3.3 Summary

In this chapter we have provided a glimpse inside the active logic planner to view the prominent belief predicates used to represent the declarative knowledge of the agent. In the next chapter we describe the inference rules that manipulate them.

---

[6] An agent must be able to infer goals from a current situation. This topic has received a great deal pf attention and treatment (e.g. [Wil83a]), but is is not within the scope of our work. We will assume that Dudley treats his belief about a deadline-coupled goal as an axiom acquired through direct observation or prior inferencing process to which we will not allude.

Chapter 4

Temporal reasoning

This chapter describes Dudley's inference mechanism for temporal reasoning. We describe three inference rules that are crucial for this: temporal projection rule (TP), context set revision rule (CSR) and the restructured modus ponens rule (RMP). In all the active logic scenarios we have underlined new formulas in the context sets or projections, to highlight the differences with the corresponding beliefs that were inherited from the previous step. In each step the TP rule derives a new projection in the current context and the RMP and CSR rules together deduce a new context set.

4.1 Temporal projection rule (TP):

The temporal persistence rule (TP) effectively *smoothes* beliefs over time intervals which present gaps in the agent's knowledge. At each step, $\mathbf{Proj_{i,p}}$ holds the results of the temporal projection rule applied to the context set $\mathbf{CS_{i-1,p}}$ of the previous step. 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 active-logics are built around the idea of specifying what is known (e.g., proven) *so far*, and all context sets, and all formulas in the *Context_List* can be simultaneously reconsidered at each new time step.

Here is a description of the TP rule applied to atomic formulas corresponding to a given function name X in the context set at step i in order to constitute $\mathbf{Proj_{i+1,p}}$ for the partial plan p. Note that formulas of the form $X(S : \overline{F}, Args)$ are not projected, since they are intended to represent mostly agent actions that do not persist. Formulas $X(S : F, Args)$ each containing the function name X are kept sorted in $\mathbf{CS_{i,p}}$ with $(S : F)$ as the key. Only functions corresponding to *fluents* (i.e. those formulas without the bar on top as in $S : \overline{F}$) are eligible for projection. The formulas in $\mathbf{Proj_{i+1,p}}$ are strictly those that are obtained by persistence of those in $\mathbf{CS_{i,p}}$. Let α_j and α_{j+1} denote consecutive formulas in sorted order in $\mathbf{CS_{i,p}}$ and let α_l denote the last formula in this order. The TP rule is described below.[1] Figure 4.1 shows a pictorial description of the TP rule, with the dashed lines denoting the intervals that are filled with the projection.

1. If α_j is of the form $X(S_j : F_j, Args)$ and α_{j+1} is of the form $X(S_{j+1} : F_{j+1}, Args)$ then $\mathbf{Proj_{i+1,p}}$ contains $X(F_j + 1 : S_{j+1} - 1, Args)$ whenever $F_j < S_{j+1}$.

2. If α_j is of the form $X(S_j : F_j, Args)$ and α_{j+1} is of the form $X_c(S_{j+1} : F_{j+1}, Args)$ then $\mathbf{Proj_{i+1,p}}$ contains $X(F_j + 1 : S_{j+1} - 1, Args)$ whenever $F_j < S_{j+1}$.

3. If α_j is of the form $X(S_j : F_j, Args)$ and α_{j+1} is of the form $\neg X(S_{j+1} : F_{j+1}, Args)$ then $\mathbf{Proj_{i+1,p}}$ does not speculate over the truth or falsity of X over $F_j + 1 : S_{j+1} - 1$. The projection rule will smooth over this interval when further information about a possible point of time where the value of X changes becomes available.

4. If α_j is of the form $X(S_j : F_j, Args)$ and α_{j+1} is of the form $\neg X_c(S_{j+1} : F_{j+1}, Args)$ then $\mathbf{Proj_{i+1,p}}$ contains $X(F_j + 1 : S_{j+1} - 1, Args)$ whenever $F_j < S_{j+1}$.

[1] For brevity, we only describe the rule for $\alpha_j = X(S_j : F_j, Args)$. The same applies to $\alpha_j = X_c(S_j : F_j, Args)$. The dual form involving $\neg X$ is similar.

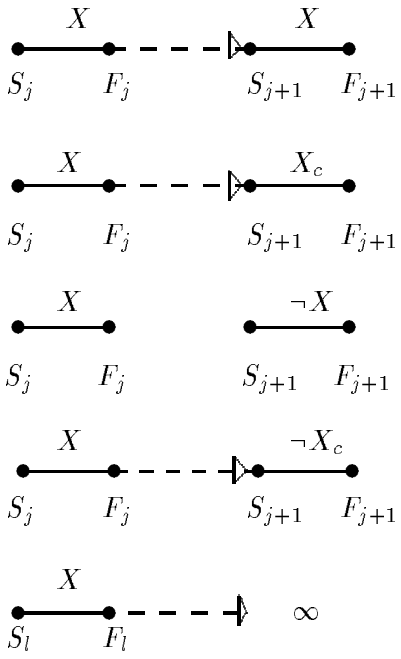

Figure 4.1: Pictorial description of the TP rule

5. If α_l (with the latest interval in $\mathbf{CS_{i,p}}$ corresponding to X is of the form $X(S_l : F_l, Args)$ or $X_c(S_l : F_l, Args)$ then $\mathbf{Proj_{i+1,p}}$ contains $X(F_l + 1 : \infty, Args)$.

If any of α_j and α_{j+1} in the context set du-contradict, the projection is frozen for the interval in dispute until the contradiction is resolved. In case of a contradiction, it may sometimes be necessary to break up the formulas into two or more parts to identify the extent of the contradiction over some sub interval. e.g., when $X(5)[Y(4)]$ and $\neg X(1:6)$ are in a context set, the latter must be split into $\{\neg X(1:4), \neg X(5), \neg X(6)\}$ to identify the range of the contradiction.

In case of a contradiction in the context set, between two formulas, α_j and α_{j+1} in the sorted order, Dudley's TP rule attempts to decide on which of the contradicting formulas to project, until the contradiction is resolved. Often times, one of the contradicting formulas is a fact while the other is weaker since it is only based on projections, and the contradiction is resolved in subsequent steps. The projection is

frozen for subsequent formulas α_{j+2}, α_{j+3} , ... in the sorted list. However, when a fact $\alpha_k = X(S_k : F_k, Args)$ is encountered further down the chain, Dudley can resume projection starting with the interval $S_k : F_k$ by applying the above procedure to the remainder of the sorted list with α_k as its first element. Projections are not made in the interval between the contradiction and S_k, until the contradiction is resolved. [Mil92] gives an account of contradiction handling in step-logics that addresses the problems of lingering consequences and the difficulties in designing recovery mechanisms. We handle only those contradictions that are the result of default beliefs derived from temporal projection. These are *internal contradictions* within the context set. We have relatively simple recovery and a simple way to deal with lingering consequences using the CSR rule that follows.

Example

Consider a scenario with a bucket filled with oil and a ball lying on the floor. This example illustrates an application of the TP rule to $\mathbf{CS_{15,p}}$ to yield $\mathbf{Proj_{16,p}}$.

$\mathbf{15} : \ldots, \mathbf{CS}(15, p, \{\ldots, filled(0), filled(4), \neg filled_c(7),$

$filled(10), on(0, floor, ball), \neg on_c(5, floor, ball), \ldots\}), \ldots$

$\mathbf{16} : \ldots, \mathbf{Proj}(16, p, \{\ldots, filled(1:3), filled(5:6), filled(11:\infty),$

$on(1:4, floor, ball), \neg on(6:\infty, floor, ball), \ldots\}), \ldots$

4.2 A restructured Modus Ponens rule (RMP):

Instead of applying MP in its familiar form : viz. from α and $\alpha \rightarrow \beta$ deduce β, we choose a representation in clause form and apply a restructured MP in accordance with our philosophy to let earlier defaults play out their effects completely to result in an anticipated state of the world to which later defaults may be applied if necessary. A formula which is a fact has no justification attached to it. All axioms are treated as facts. A formula α which was derived using one or more projections $\beta_1, \beta_2 \ldots$ is

only as feasible as the weakest projection, and is itself classified as a default. Such a formula is annotated with the projections used in its derivation and is written as $\alpha[\beta_1, \beta_2, ...]$.

Let $\neg\alpha_1 \vee \neg\alpha_2 \vee \ldots \vee \neg\alpha_n \vee \beta$ be the expression in clause form that appears in the context set $\mathbf{CS_{i,p}}$ of a plan p at step i. This may either be an axiom or an observation. We formulate a rule that adds new atomic formulas (with or without justifications) derived within a context to $\mathbf{CS_{i+1,p}}$. We use the terms, *finished* and *unfinished* to describe a resolution where the results are atomic and non-atomic formulas respectively. The rule only carries over the result of a finished resolution to the context set at the next step.[2]

The process of resolution can be outlined as follows:

- All the α_j from $\mathbf{CS_{i,p}}$ which are facts are first used to resolve. If there are no facts that are eligible for resolution, the resolution is not carried out at all. There must be at least one fact among the resolvents for the RMP to fire[3].

- Subsequently, if the resolution is unfinished, members from $\mathbf{CS_{i,p}} \cup \mathbf{Proj_{i,p}}$ which

[2]Since we have the luxury of applying all rules to all formulas at every step, not much is to be gained by adding the results of an unfinished resolution to the context set. We wait to get more information later such as from observations or from further deductions so that an atomic formula can be derived. This serves the purpose of limiting the size of the context set. It is possible to write a version of the RMP that will also add the results of unfinished formulas to the context set, and would effectively function the same but use more space.

[3]The motivation behind this is to reduce the large number of formulas resulting from applying RMP to projections alone, since what can be derived thus is also obtained by the combined effect of RMP and TP. This reduces the actual number of formulas in the context set without loss of any meaningful commonsense conclusions. As an example, consider the axioms $Alive(T) \rightarrow \neg Dead(T)$. Suppose $Alive(0)$ is the only formula in the context set. By TP, the agent would add $Alive(1:\infty)$ to the projection. and by RMP, $\neg Dead(0)$ to the context-set. Note that $\neg Dead(1:\infty)$ will subsequently be in the projection, and there is no need to additionally have $\neg Dead(1:\infty)[Alive(1:\infty)]$ in the context set. Hence this is a reasonable way to curtail the size of the context set.

are themselves defaults are next tried. All formulas from $\mathbf{Proj_{i,p}}$ as well as those formulas in $\mathbf{CS_{i,p}}$ that are annotated with projections qualify as defaults. From among those in $\mathbf{Proj_{i,p}}$, those with earlier time parameters are used before the ones with later parameters. For the annotated formulas, the annotation with the latest time parameter that was used in the derivation is used to decide the priority[4].

- The result of the resolution β is then annotated with all the projections used, either directly, or in the annotation of resolving formulas from the context set. The annotations are attached in square brackets to the formulas. This provides the basis for a real-time truth maintenance mechanism which is useful in resolving contradictions.

- If a projection α with a later time than the time of β is used in the RMP application, β is not added to the context set. It is discarded. Thus the axioms are used to derive future conjectures based on projections of current beliefs, but prevented from using future projections to derive past conclusions and from jumping to extreme conclusions as we demonstrate in Section 5.1.

- If $\beta = X(S : F, \ldots)$ has its time interval $S : F$ such that S is later than the time intervals of all the α_j used in the resolution, then it is marked as $X_c(S : F, \ldots)$ in the context set, to denote that it could be a potential point of inflection in the value of the function X.[5] This marking is of help in deciding whether to project X such as in the TP rule above.

[4] In case of a tie, we draw all the conclusions resulting from the use of the projections with identical time intervals, one at a time. This may result in implicit contradictions. In this situation, what the system deduces an expected contradiction.

[5] There is an implicit causality assumption here; earlier events are potential causes in axioms for changes but later events are useful only in explanations of past values, not responsible for changing the past values. Note that we say a *potential* point of change.

Example

This example illustrates two applications of RMP (in steps 4 – 6) following Dudley's observation of someone dropping a ball into a full bucket. Given the axioms that a ball dropped into a full bucket results in a spill, Dudley concludes that the floor will no longer be dry[6].

Axioms(These are part of the $Context\_List$ of every context set):

$\neg Filled(T) \vee \neg Drop(\overline{T}, ball) \vee Spill(\overline{T+1})$

$\neg Spill(\overline{T}) \vee \neg Dry(T+1, floor)$

4 : **CS**$(4, null, \{\ldots, Dry(0, floor), Filled(0), \underline{Drop(\overline{4}, ball)}\})$,
Proj$(4, null, \{\ldots, Dry(1:\infty, floor), Filled(1:\infty)\})$

5 : **CS**$(5, null, \{\ldots, \underline{Spill_c(\overline{5})[Filled(4)]}, Dry(0, floor), Filled(0), Drop(\overline{4}, ball)\})$,
Proj$(5, null, \{\ldots, Dry(1:\infty, floor), Filled(1:\infty)\})$

6 : **CS**$(6, null, \{\ldots, Spill(\overline{5})[Filled(4)], Dry(0, floor),$
$\underline{\neg Dry_c(6, floor)[Filled(4)]}, Filled(0), Drop(\overline{4}, ball)\})$,
Proj$(6, null, \{\ldots, Dry(1:\infty), Filled(1:\infty)\})$ [7]

The RMP rule is used in extending the context set. This allows Dudley to compute the extended effects of actions. It also allows him to deduce the future consequences of his planning as it interacts possibly with the actions of other agents or with events observed in the world. It allows for reasoning with the current projection by letting earlier events play out their consequences in an anticipated future before later events.

[6] This is an extended effect of the spill; we will elaborate later on the significance of this type of reasoning.

[7] Note that there is a contradiction between the **CS** and the **Proj**. It is resolved in the next step.

4.3 Context set extension and revision rule (CSR):

The CSR rule ensures that the context set is always kept updated to match the most current projection, and the state of the world in which the agent is situated. As explained before, formulas are annotated by the projections which are used to support them in future conjectures. In the event that the projections cease to hold as of "now", the formulas that are supported by them are dropped from the context set in the revision process. The revision is a kind of real-time truth maintenance. The CSR rule also plays the important role of resolving contradictions in a time situated manner.

Following is a description of the rule used in deciding the contents of $\mathbf{CS_{i+1,p}}$ based on the contents of $\mathbf{CS_{i,p}}$, $\mathbf{Proj_{i,p}}$ and $\mathbf{Ppl_{i,p}}$. In Phase I we decide a set $\mathbf{Candidates_{i,p}}$ selected from $\mathbf{CS_{i,p}}$ which are formulas to be considered as candidates for retention. Phase II decides which members of $\mathbf{Candidates_{i,p}}$ will make it to $\mathbf{CS_{i+1,p}}$.

Phase I (Select candidate formulas to inherit):

1. If two formulas α and δ in $\mathbf{CS_{i,p}}$ du-contradict each other, then the following criteria are used in deciding which of them go to $\mathbf{Candidates_{i,p}}$[8].

   (a) If α is a fact, while δ is a default (is annotated with a projection), select α and reject δ to go into $\mathbf{Candidates_{i,p}}$.

   (b) If α and δ are both defaults, select neither[9] to go into $\mathbf{Candidates_{i,p}}$.

[8] Note that we do not encounter situations in which facts (direct or indirect descendents of observations alone) contradict each other. Observations with different time intervals involving the same function may well disagree, but these are not contradictory.

[9] Where both are defeasible beliefs, a working strategy is to not inherit either of them, and to continue the reasoning to see if one of them will reappear in the face of stronger evidence.

2. Formulas that are not part of a contradiction go into $\mathbf{Candidates_{i,p}}$.

Phase II (Choose among candidate formulas):

1. A formula α from $\mathbf{Candidates_{i,p}}$ which is a fact is inherited to $\mathbf{CS_{i+1,p}}$.

2. A formula $\alpha[\beta_1, \beta_2, \ldots, \beta_k]$ which is a default is inherited unless for some $1 \leq j \leq k$, $\beta_j \notin \mathbf{Proj_{i,p}}$. Also, if any of $\beta_1, \ldots, \beta_k$ now appear as facts in $\mathbf{CS_{i,p}}$, they are removed from the annotation.

3. Formulas corresponding to actions that are added to the plan in the previous step are added to the $\mathbf{CS_{i+1,p}}$[10].

We remark here, that there are two more rules which fire to add formulas to $\mathbf{CS_{i+1,p}}$. One is the OBS rule which add new observations that are made in step $i+1$ to $\mathbf{CS_{i+1,p}}$. The other is RMP, which was described before in detail.

Example

In step 5 in this example, Dudley concludes that there must have been a spill at step 5 based on the projection that the bucket was still filled at step 4. At the same time, however, he is told by a reliable observer that the bucket was in fact not filled at step 4 and adopts it as a fact.

The projection catches up at step 6 to no longer believe $Filled(4)$. As a result of CSR, $Spill(\overline{5})[Filled(4)]$ and $\neg Dry_c(6, floor)[Filled(4)]$ are no longer inherited to the context set at step 7. Note that the projection at step 7 already reflects a wet floor. This will also be subsequently revised in step 8 by an application of the TP rule, since $\neg Dry_c(6, floor)[Filled(4)]$ is no longer in $\mathbf{CS_{7,null}}$.

[10] Formulae in the context set are in fact, doubly annotated in the implementation, with the projections if any used in their derivation, and with the action(s) in the plan that are used in their derivation. Should the plan get revised to no longer require any of these actions, the corresponding formula is not inherited in the CS.

$\mathbf{4}$: $\mathbf{CS}(4, null, \{\ldots, Dry(0, floor), Filled(0), \underline{Drop(\overline{4}, ball)}\})$,
$\mathbf{Proj}(4, null, \{\ldots, Dry(1:\infty, floor), Filled(1:\infty)\})$

$\mathbf{5}$: $\mathbf{CS}(5, null, \{\ldots, \underline{Spill(\overline{5})[Filled(4)]}, Dry(0, floor), Filled(0), \underline{\neg Filled(4)},$
$Drop(\overline{4}, ball)\})$, $\mathbf{Proj}(5, null, \{\ldots, Dry(1:\infty, floor), Filled(1:\infty)\})$

$\mathbf{6}$: $\mathbf{CS}(6, null, \{\ldots, Spill(\overline{5})[Filled(4)], Dry(0, floor), Filled(0), \neg Filled(4),$
$\underline{\neg Dry_c(6, floor)[Filled(4)]}, Drop(\overline{4}, ball)\})$,
$\mathbf{Proj}(6, null, \{\ldots, Dry(1:\infty, floor), \underline{\neg Filled(5:\infty)}\})$

$\mathbf{7}$: $\mathbf{CS}(7, null, \{\ldots, Dry(0, floor), Filled(0), \neg Filled(4), Drop(\overline{4}, ball)\})$,
$\mathbf{Proj}(7, null, \{\ldots, \underline{Dry(1:5, floor)}, \underline{\neg Dry(7:\infty, floor)}, \neg Filled(5:\infty)\})$

$\mathbf{8}$: $\mathbf{CS}(8, null, \{\ldots, Dry(0, floor), Filled(0), \neg Filled(4), Drop(\overline{4}, ball)\})$,
$\mathbf{Proj}(8, null, \{\ldots, \underline{Dry(1:\infty, floor)}, \neg Filled(5:\infty)\})$
$\vdots$

4.4 Summary and related work

This chapter described three key rules that form the core of the temporal inferencing. The TP rule performs temporal projections in the current context of each plan. The RMP rule extends the *Context_List* of the current context set to enrich it, enabling the agent to reason about the changing environment and the extended effects of its actions. The CSR rule ensures that the context set is kept consistent with the most current projection. Chapter 5 shows how this machinery for temporal reasoning, although designed with deadline-coupled planning applications in mind tackles frame issues in a deadline-coupled way.

The issue of temporal projection has been extensively studied in the AI literature. In particular, much of effort was devoted to the problem of forward temporal

projections, or predictions that are necessary for planning. This is the problem of determining all the facts that will true during a future time period, given a partial description of the facts that are known. Numerous solutions have been proposed to the temporal projection problem [Gel88, Geo87, Hau87, Kau86, Lif87b, Lif87a, Mor88, Pea88, Sho88, Bak89] to mention a few.

They may be grouped into four categories:

1. chronological minimization [Sho88, Kau86]
2. causal minimization [Hau87, Lif87b]
3. model approach [MS88, Ams91, GS87]
4. time map approach [DM87]

Our projection mechanism has commonalities with some of the chronological minimization approaches, notably those of Shoham [Sho88], Lifschitz [Lif87a], and Kautz [Kau86]. In our approach as well as theirs, defaults are applied forward in time, so that earlier events play out their consequences for later ones. However, these approaches specialize in forward temporal projection problems and can handle backward projections, but cannot solve explanation problems or be used by an active agent who may obtain new information while doing projections. Our projection mechanism provides an active agent the capability to revise its conclusions, in light of new observations, to give explanations to previous events, and to use its predictions in planning.

We prefer a different term for our projection approach: *parallel projection*. That is, the entire known state of the world at one moment (may be in the context of a plan) is used to determine the (expected) state at the next moment. Since active logics are built around the idea of specifying what is known (proven) *so far*, all predicates can be simultaneously reconsidered at each new time step. Persistence of a specific predicate (e.g., At(1:5, dudley,home)) from one moment to another depends on everything that is

(known to be) true at the earlier moment, because in fact that is how things are really affected in the world. This is also the central idea behind the resolution rule(RMP) described in Chapterchapter:temporal.

The chronological minimization approaches were criticized by Kautz [Kau86], in that, by always granting preference to later changes over earlier ones, the agent may draw artificial conclusions about the time of an action that caused a change. The causal minimization approaches [Hau87, Lif87b] introduce a special "Causes" predicate which they then minimize using circumscription. A drawback is that the axioms cannot be written in an unrestricted logic and therefore it may be difficult to use it in a planning system. Changes that have known causes are preferred over unjustifiable changes. However, this means that when no knowledge about a cause exists, changes cannot be reasoned with correctly. Uncertainty is tackled to some extent in [Hau87] through the notion of "potential causes" when the agent knows about the occurrence of an action but is uncertain about its success. We do not yet have a mechanism to handle this type of uncertainty (we assume that all actions in the agent's plan succeed unless observed otherwise).

Among the other original approaches, Morgenstern and Stein [MS88] provide an elegant solution to both backward and forward projections. However, in their solution, there is no time attached to the reasoning; a meta-reasoner concludes (supposedly sometime after all the observations) all the facts true in a "chronicle", given the partial description. They select models using the criterion of "fewer unmotivated actions". However, there is no acting agent who can actually compute these models and check which one is preferred.

Ginsberg and Smith [GS87] present an approach of reasoning about action and change using possible worlds. The approach involves keeping a single model of the world that is updated when actions are performed. The update procedure involves constructing the nearest possible world to the current one in which the consequences

of the actions under consideration hold. There is no explicit notion of time in this approach and the reasoning done by an "outside" reasoner, hence the time of reasoning is not a concern.

Dean and McDermott [DM87] present techniques for temporal database management. They allow two types of prediction in their system: *projection* and *refinment*. Their system is based on a temporal map that can be described by a graph in which the nodes are instants of time associated with beginning and ending events, and the arcs connecting these nodes describe relations between pairs of instants. We use ordered lists as a simpler data structure and in our framework time is associated with events and predicates, and not the other way around as in [DM87];[11] however, the projection is done in a similar way.

As in previous systems we compared with, Dean and McDermott describe their mechanism as "reasoning about time from the outside. It's as though all of what you know about the past, present and future is laid out in front of you." We consider reasoning done by an agent in time. It has only the past and the present in front of it, and it takes the passage of time into its reasoning process.

[11] Our projection mechanism is similar to that of [DM87], but we distinguish the case where a change occurs but it is not clear when it occurs. In particular, if α_j is of the form $X(S_j : F_j, Args)$ and α_{j+1} is of the form $\neg X(S_{j+1} : F_{j+1}, Args)$ then $\mathbf{Proj_{i+1,p}}$ does not speculate over the truth or falsity of X over $F_j + 1 : S_{j+1} - 1$. The projection rule will smooth over this interval when further information about a possible point of time where the value of X changes becomes available.

Chapter 5

Frame issues in real-time planning

5.1 Time-situated variations of the YSP

Chapter 4 described Dudley's time-situated reasoning mechanism. Other aspects of the planner related to time estimates and feasibility analysis with respect to tight deadlines are described in Chapter 7 and also in Chapter 6. We demonstrate here how this mechanism applies to several real-time variations of the Yale Shooting Problem [HM87] appropriate to active logics. The first is a *witness* scenario where Dudley is a witness to the scene of the crime. In it we show how Dudley draws the intuitive conclusion that Fred must be dead on observing a shoot action, and discard the unintuitive one where the gun mysteriously gets unloaded just before the shooting. We present two developments of the witness scenario which are of a *detective* nature where Dudley must offer a reasonable explanation about actions in the past, to fit his present observations. On seeing Fred alive at a later time, the same mechanism allows him to continue to perform belief revision to account for "why things went wrong"[MS88]. The last is a *killer* scenario where Dudley formulates a plan to kill Fred by a certain deadline and reasons that Fred is expected to be dead in the context of his plan to carry out a shoot action.

In the *classical* YSP problem, there is a certain ambiguity about the role of the

reasoner. There the reasoning is itself timeless, presumably it takes place after all the events in question. Our treatment is significant in that Dudley the reasoner can reason *in* time about the events in progress and adjust his reasoning to suit new observations.

Witness: We suppose that the reasoner is an eyewitness on the scene of the crime: Dudley sees Fred and sees a loaded gun at time 0, but no action is observed. Then, following a wait period during which nothing happens, at time 4 Dudley sees the gun being fired at Fred, but cannot see what happens to Fred after that. Dudley is then draws the commonsense conclusion that Fred has been killed. Here we are mimicking Baker's simplified version of the YSP (no loading action[HM87] is required, and the *wait* action occurs between steps 0 and 4).

Below we sketch the steps in this reasoning.

Axioms(These are part of every context set):

$\neg Loaded(T) \vee \neg Shoot(\overline{T}) \vee \neg Alive(T+1)$;

$\neg Alive(T) \vee Alive(0:T)$

0: $\mathbf{CS}(0, null, \{\underline{Alive(0)_{obs}}, \underline{Loaded(0)_{obs}}\})$,

$\mathbf{Proj}(0, null, \{\})$

Dudley observes that a gun is loaded, and that Fred is alive. Dudley is a passive eyewitness. Hence the reasoning context is that of a *null* plan. There is no projection yet regarding either *Alive* or *Loaded*.

1: $\mathbf{CS}(1, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(1, null, \{\underline{Alive(1:\infty)}, \underline{Loaded(1:\infty)}\})$

Rules yielding new conclusions: TP. There are no new observations.

2: $\mathbf{CS}(2, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(2, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

Wait period. No new actions or conclusions.

3: $\mathbf{CS}(3, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(3, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

Wait period.

4: $\mathbf{CS}(4, null, \{Alive(0)_{obs}, Loaded(0)_{obs}, \underline{Shoot(\overline{4})_{obs}}\})$,

$\mathbf{Proj}(4, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

Rules yielding new conclusions: OBS. A shooting is observed by Dudley.

5: $\mathbf{CS}(5, null, \{Alive(0)_{obs}, \underline{\neg Alive_c(5)[Loaded(4)]}, Loaded(0)_{obs}, Shoot(\overline{4})_{obs}\})$,

$\mathbf{Proj}(5, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

Rules yielding new conclusions: RMP. In the RMP application to the axiom $\neg Loaded(T) \vee \neg Shoot(\overline{T}) \vee \neg Alive(T+1)$ in clause form, $Shoot(\overline{4})$ is resolved with first since it is a fact. Next, $Loaded(4)$ is used in favor of $alive(5)$ in the resolution due to the projection being at an earlier time. It is allowed to play its effects before the projection at the next time is considered. The projection $Loaded(4)$ used in the inference is used to annotate the inference $\neg Alive_c(5)[Loaded(4)]$, and the result is noted as a possible point of inflection in the value of the predicate *Alive*.

6: $\mathbf{CS}(6, null, \{Alive(0)_{obs}, \neg Alive_c(5)[Loaded(4)], Loaded(0)_{obs}, Shoot(\overline{4})_{obs}\})$,

$\mathbf{Proj}(6, null, \{\underline{Alive(1:4)}, \underline{\neg Alive(6:\infty)}, Loaded(1:\infty)\})$

Rules yielding new conclusions: TP. Note that *Alive* is no longer projected to infinity but only until time 4 according to the new context set information. Also $\neg Alive$ is projected from time 6 onwards. The projection for *Loaded* remains unchanged.

$\vdots$

The witness version of the YSP gives the intuitive answer: In the context of the *null* plan, Fred must have died at step 5 as a result of the shooting, provided of course, that the default regarding the gun staying loaded up until step 4 is indeed

true. $\neg Alive(5)[Loaded(4)]$ is still defeasible, only as good as $Loaded(4)$ really, and is treated as a default.

Detective: To illustrate the real-time nature of our reasoning process, we now consider two different developments of the witness scenario following step 6. In one of them, at step 7, Fred is seen to be alive and walking about, in another, at step 7, the gun is examined, and found to be in fact unloaded. Suppose that the shooting itself does not unload the gun. What conclusions can Dudley make in each of these developments of the former scenario? In short, how can Dudley play detective in real-time?

Development 1: But Fred is alive!

7: $\mathbf{CS}(7, null, \{\ldots, Loaded(0)_{obs}, Shoot(\overline{4})_{obs}, Alive(0)_{obs}, \neg Alive_c(5)[Loaded(4)],$
$\underline{Alive(7)_{obs}}\})$, $\mathbf{Proj}(7, null, \{Alive(1:4), \neg Alive(6:\infty), Loaded(1:\infty)\})$

Rules yielding new conclusions: OBS. Dudley sees that Fred is alive.

8: $\mathbf{CS}(8, null, \{\ldots, Loaded(0)_{obs}, Shoot(\overline{4})_{obs}$
$Alive(0)_{obs}, \underline{Alive(1:6)}, \neg Alive_c(5)[Loaded(4)], Alive(7)_{obs}\})$
$\mathbf{Proj}(8, null, \{\underline{Alive(1:4)}, \underline{Alive(8:\infty)}, Loaded(1:\infty)\})$

Rules yielding new conclusions: RMP, TP. $Alive(1:6)$ is derived by applying RMP to $\neg Alive(T) \vee Alive(0:T)$ in clause form. $Alive(7)$ being a fact, is used. The TP rule tries to make the best of all the *Alive* formulas in the context set to yield a projection. It will straighten out in the next step.

9: $\mathbf{CS}(9, null, \{\ldots, Loaded(0)_{obs}, \underline{\neg Loaded(4)}, Shoot(\overline{4})_{obs}, Alive(0)_{obs}, Alive(1:6),$
$Alive(7)_{obs}\})$, $\mathbf{Proj}(9, null, \{\underline{Alive(8:\infty)}, Loaded(1:\infty)\})$

Rules used to yield new conclusion: RMP, CSR, TP. The RMP rule derives $\neg Loaded(4)$ from $Alive(5)$, $Shoot(\overline{4})$ and $\neg Loaded(T) \vee \neg Shoot(\overline{T}) \vee \neg Alive(T+1)$. The CSR rule must deal with a contradiction in the context set. $Alive(5)$ is a fact, while the default is $\neg Alive_c(5)[Loaded(4)]$. $Alive(5)$ makes it into the $\mathbf{Candid_{8,null}}$ set and is subsequently inherited, while the default is rejected. The TP rule also confronts the

contradiction, but handles it with ease, since $Alive(0:7)$ are all facts. The projection is resumed quickly past the contradiction due to the presence of later facts.

10: $\mathbf{CS}(10, null, \{\ldots, Loaded(0)_{obs}, \neg Loaded(4), Shoot(\overline{4})_{obs}, Alive(0)_{obs}, Alive(1:6),$
$Alive(7)_{obs}\}), \mathbf{Proj}(10, null, \{\underline{Alive(8:\infty)}, \underline{\neg Loaded(5:\infty)}\})$

Rules yielding new conclusions: TP. The TP rule changes the projection to reflect the new knowledge that the gun was unloaded at step 4. Note, that no projections are made on whether the gun was loaded at steps 2 or 3.

$\vdots$

Development 2: Look at this gun, it is not loaded!

7: $\mathbf{CS}(7, null, \{Loaded(0)_{obs}, \underline{\neg Loaded(7)_{obs}} Shoot(\overline{4})_{obs},$
$Alive(0)_{obs}, \neg Alive_c(5)[Loaded(4)], \}),$
$\mathbf{Proj}(7, null, \{Alive(1:4), \neg Alive(6:\infty), Loaded(1:\infty)\})$

Rules yielding new conclusions: OBS. In this development of the eyewitness scenario, Dudley observes that the gun is unloaded at step 7. This reduces the feasibility of the $Loaded(4)$ default, and Dudley's conclusions take a more skeptical turn. Note that there is already a disagreement between the formulas in $\mathbf{Proj_{7,null}}$ and $\mathbf{CS_{7,null}}$. It takes the agent one step to get the projection to match the changed context set.

8: $\mathbf{CS}(8, null, \{Loaded(0)_{obs}, \neg Loaded(7)_{obs}, Shoot(\overline{4})_{obs},$
$Alive(0)_{obs}, \neg Alive_c(5)[Loaded(4)], \}),$
$\mathbf{Proj}(8, null, \{Alive(1:4), \neg Alive(6:\infty), \underline{\neg Loaded(8:\infty)}\})$

Rules yielding new conclusions: TP. The projection is revised. Since 7 is not believed to be a point of change of the $Loaded$ predicate (Dudley merely stumbles upon the unloaded gun, he does not see it being unloaded), no conjecture about $Loaded$ is made over the $1:6$ interval.

9: $\mathbf{CS}(9, null, \{Loaded(0)_{obs}, \neg Loaded(7)_{obs}, Shoot(\overline{4})_{obs}, Alive(0)_{obs}, \}),$
$\mathbf{Proj}(9, null, \{Alive(1:4), \neg Alive(6:\infty), \neg Loaded(8:\infty)\})$

Rules yielding new conclusions: CSR. $\neg Alive_c(5)[Loaded(4)]$ is not inherited since the projection in the annotation is not supported at the current step.

10: $\mathbf{CS}(10, null, \{Loaded(0)_{obs}, \neg Loaded(7)_{obs}, Shoot(\overline{4})_{obs}, Alive(0)_{obs}\})$,

$\mathbf{Proj}(10, null, \{\underline{Alive(1:\infty)}, \neg Loaded(8:\infty)\})$

Rules yielding new conclusions: TP. The Projection adjusts to the change in the *Alive* formulas in the context set. In the face of the observation that the gun was not loaded at step 7, Dudley reserves the conclusion regarding Fred's death from the shooting[1]. Should more evidence become available at a later time, this conclusion may be reinforced again.

$\vdots$

Both the above are explanation scenarios. The elegant recovery of the agent is to be attributed to the non-monotonic inference process that initiates changes to restore consistency. In the first development, when Fred is found to be alive past the time of the shooting, Dudley can successfully conclude that the gun must have been unloaded between the loading and the time of the shooting. Furthermore, *he can successfully change his on-going reasoning model to reflect these changes.* In the second development, when the gun is found to be unloaded at a later time, under the assumption that shooting does not unload the gun, Dudley chooses to be ambivalent about Fred's death. Even though he had imagined earlier that Fred would be dead, now he no longer has the same confidence in the projection, it must be changed, and the conclusion about Fred's death refrained from making, until more is known about the time of the unloading.

Killer: At time 0, Dudley sees that there is a loaded gun, and that Fred is alive. He is playing the killer, and has the goal $\neg Alive(5)$. He must formulate a plan to kill Fred. Our temporal projection mechanism must allow Dudley to support the intuitive

[1]Since Dudley doesn't know when the gun was unloaded, he doesn't conclude $\neg Loaded(4)[Alive(5)]$ or $\neg Alive(5)[loaded(4)]$. $Loaded(0)$ is not projected due to the fourth item of the TP rule, since 7 is not believed to be a point of change of the *Loaded* predicate, as explained above. The conclusion $\neg Loaded(4)[Alive(5)]$ is prevented by the fourth item of the RMP rule, i.e., if a projection α with a later time than the time of β is used in the RMP application, β is not added to the context set.

extension that Fred will be dead in the context of Dudley's plan. This is crucial for Dudley to 'imagine' the unfolding of his plans. We show a few illustrative steps from this killer scenario:

0: $\mathbf{Goal}(kill, \neg Alive(5), 5)$,

$\mathbf{Proj}(0, null, \{\})$,

$\mathbf{CS}(0, null, \{\underline{Alive(0)_{obs}}, \underline{Loaded(0)_{obs}}\})$

Dudley's goal results in the formation of a partial plan. In this paper we do not describe the details of plan formation. The context set of the partial plan in the first step in its formation is the context set of the null plan. Also, the first projection is the projection in the null context.

1: $\mathbf{Goal}(kill, \neg Alive(5), 5)$,

$$\mathbf{Ppl}(1, kill, \left\{ \left[\begin{array}{c} Loaded(4) \\ Shoot(\overline{4}) \\ \neg Alive(5) \end{array} \right]_1 \right\})$$

$\mathbf{CS}(1, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(1, null, \{\underline{Alive(1:\infty)}, \underline{Loaded(1:\infty)}\})$,

$\mathbf{CS}(1, kill, \{\underline{Alive(0)_{obs}}, \underline{Loaded(0)_{obs}}\})$

$\mathbf{Proj}(1, kill, \{\underline{Alive(1:\infty)}, \underline{Loaded(1:\infty)}\})$,

Dudley formulates a plan to achieve the goal.

2: $\mathbf{Goal}(kill, \neg Alive(5), 5)$,

$$\mathbf{Ppl}(2, kill, \left\{ \left[\begin{array}{c} Loaded(4) \\ Shoot(\overline{4}) \\ \neg Alive(5) \end{array} \right]_1 \right\})$$

$\mathbf{CS}(2, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(2, null, \{Alive(1:\infty), Loaded(1:\infty)\})$,

$\mathbf{CS}(2, kill, \{Alive(0)_{obs}, \underline{Shoot(\overline{4})}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(2, kill, \{Alive(1:\infty), Loaded(1:\infty)\})$

$Shoot(\overline{4})$ is added to the context of the "kill" plan, since the action was added to the plan at the previous step.

3: $\mathbf{Goal}(kill, \neg Alive(5), 5)$

$$\mathbf{Ppl}(3, kill, \left\{ \left[\begin{array}{c} Loaded(4) \\ Shoot(\overline{4}) \\ \neg Alive(5) \end{array} \right]_1 \right\})$$

$\mathbf{CS}(3, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(3, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

$\mathbf{CS}(3, kill, \{Alive(0)_{obs}, \underline{\neg Alive_c(5)[Loaded(4)]}, Shoot(\overline{4}), Loaded(0)_{obs}\})$,

$\mathbf{Proj}(3, kill, \{Alive(1:\infty), Loaded(1:\infty)\})$

In the context of his plan to kill, Dudley concludes that Fred will die. Within the context of his plans, his actions are treated as facts. It will take it another step to reach the same conclusion in the *null* plan.

4: $\mathbf{Goal}(kill, \neg Alive(5), 5)$, $\mathbf{Ppl}(4, kill, \underline{\{\}})$,

$\mathbf{CS}(4, null, \{Alive(0)_{obs}, Loaded(0)_{obs}, \underline{Shoot(\overline{4})_{obs}}\})$,

$\mathbf{Proj}(4, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

$\mathbf{CS}(4, kill, \{Alive(0)_{obs}, Shoot(\overline{4})_{obs}, \neg Alive_c(5)[Loaded(4)], Loaded(0)_{obs}\})$,

$\mathbf{Proj}(4, kill, \{\underline{Alive(1:4)}, \underline{\neg Alive(6:\infty)}, Loaded(1:\infty)\})$

Dudley has acted upon his 'kill' plan, and the action triplet is removed from the partial plan, in accordance with the inference rules for planning and acting (not described here). Shoot action is observed, and appears in the context of both plans. In the context of his plan to kill, he concludes that Fred is dead from the shooting provided the projection that the gun stayed loaded is true.

$\vdots$

5.2 Two more aspects of the frame problem

The temporal persistence rule (TP) tackles the projection frame problem, in deciding what fluents persist in the context of a given plan and for how long. The Context Set Revision rule (CSR), together with the restructured Modus Ponens rule (RMP) deals with the ramification problem, by allowing the agent to reason about the extended

effects of its actions in the context of its plans. This allows Dudley to form a comprehensive mental image of the future and to extend it in time in accordance with his changing environment. The qualification problem must also be addressed if Dudley must anticipate the success of his actions in a plan. The treatment of various versions of the YSP illustrated Dudley's ability to handle the temporal persistence problem. In this section we examine how the active-logic planning mechanism offers solutions to the qualification and ramification problems.

5.2.1 The qualification problem

In a nutshell, the qualification problem is the impossibility of specifying the (pre)conditions required for an action. A famous example (due to McCarthy) is the 'banana in the tailpipe' problem. There are an infinite number of eventualities that can potentially interfere in the successful execution of an action toward its intended effect. It is impossible for the reasoner to account for all of these during the plan formulation or execution phases.

Apparently, in case of human commonsense reasoners, the knowledge representation for an action is determined by a complex learning mechanism, but once developed, the agent simply maintains a short list of preconditions that he/she deems necessary for the 'normal' execution of an action and takes them into account while formulating his/her plans. This has the tremendous advantage of keeping the planning process simple. But agents do not function in a static world. Hence the plans are not crisp. The agent is constantly observing and 'thinking' about his/her plan in the interval between formulation and execution. He/she has a great deal of world knowledge that makes it possible to recognize the presence of qualifications. Commonsense agents can incorporate these qualifications into the reasoning to alter their plans in suitable ways. The 'normal' conditions for an action are those that the agent is accustomed to accounting for while forming one's plan.

There are two aspects of the qualification problem. One falls into the category of learning. This is the problem of identifying whether a given condition is a qualification for a particular action. For example, a child learns through (repeated) trials that an object is impossible to lift with its fingers when it is hot. Thus $Hot(T, X)$ is a potential hindrance to the attempted action $Lift(T, object)$. Until the condition is not known to be a qualification (the axiom $Hot(T, object) \rightarrow Hinders(Lift(T, object))$ is not known), the agent attempts to lift the hot object and subsequently fails, unaware of the qualification. Thus agents need suitable learning capabilities to identify qualifications. The other aspect falls within the realm of planning. This is the problem of incorporating the *known qualifications* into the planning process *when considered necessary*. The human commonsense agent goes about its way, thinking only of the 'normal' conditions until some pointers to a potential qualification are seen in his/her knowledge base. Our real-time automated reasoner plans in a changing world. Observations can trigger further inferencing. Especially in deadline situations, these may be very crucial to making timely modifications to one's plans.

Consider the banana in the tailpipe. Suppose human agent Steve drives to work everyday. As he steps out of the house towards the car, Steve thinks of the only condition in his knowledge representation for the action of starting the engine – he just fumbles in his pocket to make sure he has the keys. If he just did not know that having a banana in the tailpipe can hinder the action of starting the car, he might think nothing of it even if he sees one sticking prominently in the tailpipe. This of course, will lead to subsequent failure to start the car, and maybe particularly expensive in a deadline situation. However, if he knows that the banana is a hindrance to the action of starting the car, and he notices it in the tailpipe, Steve, as a commonsense human agent immediately realizes that the plan of starting the car will not work. it needs modification. Possible modifications may include efforts to remove one or more of the responsible causes, such as getting rid of the banana, or may involve alternate

planning to achieve the goal such as deciding to walk to work instead.

We would like our automated agent to have the capability to take care of a known qualification in its planning. The context set of the plan is continuously revised by the agent. As long as there is no formula in the context set indicating a hindrance to any of the actions in thee plan, the agent concludes that it is ok to go ahead with the current action. This may lead to failure under some circumstances. However, in others, with the added real-time capacity to take notice of changes in one's environment, the agent may notice a qualification in time to modify the plan suitably.

We assume that Dudley has an inference rule that enables him to replan once a hindrance has been noticed. Here we only encapsulate it in the phrase *modify plan*. The details of the replanning have been omitted.

Inference rule to take note of a hindrance:

$$\frac{i : CS(i, p, \{\ldots, Hinders(A), \ldots\}), Ppl(i, p, \{\ldots, [C_A, A, R_A], \ldots\})}{i+1 : \textit{modify plan } p}$$

Axioms(these are part of every context set):

$\neg Start(T : \overline{T+1}, engine) \vee Running(T+1, engine)$

$condition(Start(T : \overline{T+1}, engine), Have(T, key))$

$\neg In(T, banana, tailpipe) \vee \neg Start(T : \overline{T+1}, engine) \vee Hinders(Start(T : \overline{T+1}, engine))$

0: **Goal**$(car\_start, Running(6, engine), 6)$,

Proj$(0, null, \{\})$,

CS$(0, null, \{\underline{Have(0, key)_{obs}}\})$

1: **Goal**$(car\_start, Running(6, engine), 6)$,

Ppl$(1, car\_start, \left\{ \left[\begin{array}{c} Have(5, key) \\ Start(5 : \overline{6}, engine) \\ Running(6, engine) \end{array} \right]_1 \right\})$,

CS$(car\_start, 1, \{Have(0, key)_{obs}\})$, **Proj**$(1, car\_start, \{\})$

2: **Goal**$(car\_start, Running(6, engine), 6)$,

$$\mathbf{Ppl}(2, car\_start, \left\{ \left[\begin{array}{c} Have(5, key) \\ Start(5:\overline{6}, engine) \\ Running(6, engine) \end{array} \right]_1 \right\}),$$

$\mathbf{CS}(2, car\_start, \{Have(0, key)_{obs}, \underline{Start(5:\overline{6}, engine)}, \underline{In(2, banana, tailpipe)_{obs}}\}),$

$\mathbf{Proj}(2, car\_start, \{\underline{Have(1:\infty, key)}\})$

3: $\mathbf{Goal}(car\_start, Running(6, engine), 6),$

$\mathbf{CS}(3, car\_start, \{Have(0, key)_{obs}, Start(5:\overline{6}, engine),$

$\underline{Running_c(6, engine)}, In(2, banana, tailpipe)_{obs}\}),$

$\mathbf{Proj}(3, car\_start, \{Have(1:\infty, key), \underline{In(3:\infty, banana, tailpipe)}\})$

$$\mathbf{Ppl}(3, car\_start, \left\{ \left[\begin{array}{c} Have(5, key) \\ Start(5:\overline{6}, engine) \\ Running(6, engine) \end{array} \right]_1 \right\}),$$

4: $\mathbf{Goal}(car\_start, Running(6, engine), 6),$

$\mathbf{CS}(4, car\_start, \{Have(0, key)_{obs}, Start(5:\overline{6}, engine), Running_c(6, engine\},$

$In(2, banana, tailpipe)_{obs}, \underline{Hinders(Start(5:\overline{6}, engine))[In(6, banana, tailpipe)]},$

$\mathbf{Proj}(4, car\_start, \{Have(1:\infty, key), In(3:\infty, banana, tailpipe), Running(6:$

$\infty, engine)\})$ $\mathbf{Ppl}(4, car\_start, \left\{ \left[\begin{array}{c} Have(5, key) \\ Start(5:\overline{6}, engine) \\ Running(6, engine) \end{array} \right]_1 \right\}),$

At step 4, the context set of plan *car_start* contains $Hinders(Start(5:\overline{6}, engine))[In(6, banana, tailpipe)]$. The agent notices this in step 5, and moves on to modify the plan, either by removing the $Start(5:\overline{6}, engine)$ action from the plan and finding an alternate one, or by removing the cause of the hindrance. We do not include details of the replanning flowing this step, since the focus of this work is the frame and temporal issues.

A relevance mechanism brings into short term focus only those conditions that may be relevant to the processing of an action at the current time. This is achieved by integrating a model of a short-term memory and an algorithm guided by heuristic rules for bringing those formulas which are relevant to the agents current line of reasoning.

For example, if the agent is not currently aware of a banana in the tailpipe, the particular axiom regarding the hindrance is not in focus, hence not in the current context set. Thus, the above formalism, combined with a relevance mechanism offers a fuller treatment of the qualification problem. Details of the space constrained agent are in [NKP93].

5.2.2 The ramification problem

This is the difficulty in explicitly recording all the consequences of an action. An action often produces, in addition to its intented effect, a variety of ramifications (or side-effects) that are often dependent on the environment in which the action is performed, upon the domain (integrity) constraints existing at the time, etc. STRIPS like systems could not handle this problem at all. Effects of actions were canned into add and delete lists. We would like the automated reasoner to have the flexibility to reason about extended effects that are within the realm of its inference capacity and permitted under his deadline constraints. We illustrate this with the help of an extension of the example scenario mentioned before in 4. To recapitulate:

Dudley is standing next to a bucket, at location $bucket\_loc$, with a ball in his hand. The bucket is filled to the brim with oil. Dudley may be the household robot assigned with normal clean up goals, and routinely drops objects into buckets (trash cans). In his 'normal' sphere of activity he seldom encounters buckets filled with oil. Dudley has the compound goal $In(5, ball, bucket) \wedge at(10, Dudley, nextroom)$, i.e., to drop the ball in the bucket and then go to the next room. His knowledge representation of the $Drop(\overline{T}, Object)$ action is determined by his everyday experience. It has the condition $At(T, Dudley, bucket\_loc) \wedge Have(Dudley, Object)$ and has the result $In(T+1, bucket, Object)$. However, when Dudley drops the ball in the bucket, the oil spills all over the floor, and hinders Dudley's next action of walking to the next room, since he slips on the oil and falls.

Can Dudley reason about this obvious extended ramification of his drop action? He knows that the bucket is full. He has the axiom that dropping an object will cause the oil in the bucket to spill. The spilled oil will cause the floor to get wet, and will hinder his walk. As in the above scenario, ramifications are of particular interest when the ramification of one action (Drop) becomes a qualification for another (Walk)[2].

Our representation of the contexts of the plans, and the continued reasoning within the context allows Dudley to take care of obvious ramifications of his actions.

Axioms:

$\neg Drop(T : \overline{T+1}, ball, bucket) \vee In(T+1, ball, bucket)$

$condition(Drop(T : \overline{T+1}, ball, bucket), At(T, dudley, bucket\_loc) \wedge have(T, ball, dudley))$

$\neg Filled(T, bucket, oil) \vee \neg Drop(T : \overline{T+1}, ball) \vee Spill(\overline{T+1}, oil)$

$\neg Spill(T : \overline{T+1}, oil) \vee \neg Dry(T+1, floor)$

$\neg Walk(\overline{T1 : T2}, dudley, L1 : L2) \vee At(15, dudley, L2)$ (v*(T2 -T1) = L2 - L1, where v is Dudley's speed.)

$condition(Walk(\overline{T1 : T2}, dudley, L1 : L2), At(15, dudley, L1))$

$Dry(T, floor) \vee \neg Walk(S : F, dudley) \vee Hinders(Walk(S : F, dudley)), S \leq T \leq F$

5: Goal$(p, In(15, ball, bucket) \wedge At(20, dudley, nextroom, 20),$

$$\mathbf{Ppl}(5, p, \left\{ \begin{array}{l} \left[\begin{array}{c} At(14, dudley, bucket\_loc) \wedge Have(14, dudley, ball) \\ Drop(14 : \overline{15}, ball) \\ In(15, ball, bucket) \end{array} \right]_1 \\ \left[\begin{array}{c} At(15, dudley, bucket\_loc) \\ Walk(15 : \overline{18}, dudley, bucket\_loc : nextroom) \\ At(18, dudley, nextroom) \end{array} \right]_2 \end{array} \right\})$$

$\mathbf{CS}(5, p, \{Filled(0, bucket)_{obs}, Have(0, ball, dudley)_{obs}, At(0, dudley, bucket\_loc)_{obs},$

$Dry(0, floor)_{obs}, Drop(14 : \overline{15}, ball), In_c(15, ball, bucket),$

$Walk(15 : \overline{18}, dudley, bucket\_loc : nextroom), At_c(18, dudley, nextroom)\})$

[2] with suitable knowledge about the domain, our mechanism will also allow us to deal with actions in the self-defeating category [GS87]. However these problems involve the characterization of the applicable physical laws, and tend to become quite complex in any framework.

Proj$(5, p, \{Filled(1:\infty, bucket), Have(1:\infty, ball, dudley),$
$At(1:17, dudley, bucket\_loc), At(18:\infty, dudley, nextroom),$
$Dry(1:\infty, floor), In(16:\infty, ball, bucket)\})$

The above step in reasoning shows a plan to achieve the goal, and shows its context set at step 5, soon after plan formulation. In steps 6, 7, and 8 to follow, by applications of RMP, $Spill(\overline{15}, oil)$, $\neg Dry(16, floor)$ and $Hinders(Walk(15:\overline{18}, dudley, bucket\_loc : nextroom))$ will be members of the *Context_List*. Once 'Hinders' enters the context of plan p, Dudley must take the required replanning steps to remedy the situation. One possible remedy could be to wait until the floor dries, but the agent must evaluate this plan against the possibility of overshooting the deadline for his goal. Thus in deadline situations, it is not sufficient for the situated agent to detect the ramifications, it also needs to have the ability to react to them in a timely fashion. This example illustrates a real-time treatment of the ramification problem.

5.3 Summary and related work

We have presented a framework that handles several real-time versions of the YSP (including several aspects of the frame problem) and performs fully deadline-coupled planning and reasoning. As such, it is difficult to compare it with isolated brittle solutions that handle particular temporal aspects of one of the above mentioned problems. Moreover, our inference engine was not designed to solve the YSP or to tackle particular aspects of the frame problem. Instead, we consider these problems as independent benchmarks against which to test our flexible and yet robust real-time reasoning mechanism.

What distinguishes this work is the capacity of an agent to perform the reasoning in a *changing* environment, accepting new observations, and incorporating them into

current (and changing) beliefs. The agent dynamically forms a model to fit his/her evolving knowledge base by using a combination of procedural and declarative methods. There is no meta-reasoner to step in and perform the difficult tasks, neither are "undecidable" methods invoked to analyze all possible models. The agent effectively computes *a* plausible model, and has the capacity to revise it as time goes on, resolving contradictions as and when they arise in the process. The mechanism functions for the agent whether he/she is the one performing the actions, or observing them.

We have compared our temporal projection approach with chronological minimization [Sho88, Kau86], causal minimization [Hau87, Lif87b], the model approach [MS88, Ams91]. and the time map approach [DM87] in Chapter 4.

Here, we compare our solution with two other recent works that address the YSP. Baker [Bak89] provides a solution to the YSP using situation calculus and circumscription, with an 'Ab' predicate in which the result is varied. It is necessary to add "existence" axioms for situations, but this is done elegantly in this framework. Baker's approach can solve most versions of the YSP (though not in real-time). It is not clear who does the reasoning, the (undecidable) circumscription provides the likely models. Since the agent's role is not obvious, it is not possible to integrate it within a planning framework, hence it can not be easily extended to the "killer" scenario. It can not solve the 'stolen car' problem,[3] which we can address. Our approach can handle all the other problems mentioned here, although our detective scenario is closer to that in [MS88] and different from the murder mystery described. Our approach can also offer a treatment for all the ramification problems mentioned in [Bak89].

[3]The 'stolen car' problem, is a problem with chronological minimization. Upon seeing his car stolen, the agent is forced to conclude that the car was stolen only at the last possible moment before it was found missing. Our mechanism does not run into a problem with the 'stolen car'. Suppose it is known that $\neg Missing(0, car)$ and then, at time 10, Dudley observes $Missing(10, car)$, the TP rule does not project $\neg Missing$ up until time 9, unless $Missing_c(10)$ is known. We do not postpone changes until the last possible moment unless a potential cause for the change is known.

[Ams91] is the only other work other than ours that attempts to discuss the issue of who the reasoner is in a given scenario, which we have addressed at length. Amsterdam highlights the advantage which a reasoner has when he/she is at the site of the action, namely, that he/she can observe an action whenever it happens. This allows the agent to utilize the closure property – "if an action is not mentioned then it did not happen". However, here again, there is a meta-reasoner doing the inference. That theory's greatest limitation (and this is said in [Ams91]) is the rigidity of the above mentioned closure principle. There are scenarios where actions can be derived from propositions and hence do not have to be explicitly specified.

In this chapter we developed various real-time versions of the Yale Shooting scenario. Using them we demonstrated Dudley's time-situated handling of the three frame problems, namely projection, qualification and ramification.

Chapter 6

Time-situated Planning

Recently there has been a surge of interest in systems capable of intelligent behavior in dynamic and unpredictable environments [AC87, Bro91, Kae88, HR90]. Resource limitations, including the treatment of time as a consumable resource has received attention in systems that perform real-time planning. In chapter 1 we examined some of the relevant literature in traditional AI planning, and discussed recent trends in the field. A comprehensive survey of research in deliberative real-time AI can be found in [GL94]. In this section we describe the extent of our contribution to AI planning. While our research tackles issues involving time and deadlines, it deals only modestly with optimality issues in planning.

This chapter describes the details of the planning mechanism. The temporal reasoning rules described in Chapter 4 provide the beliefs regarding the context sets and the temporal projections which are used to decide the structure of the partial plans. Real-time truth maintenance ensures that all the contexts are kept current.

6.0.1 Issues in real-time planning

Real-time systems research from the AI perspective has regard for two major problems as described in [BH91]:

1. Response time constraints

2. Dynamic situations

Response time constraints for a task pose a limitation on the total time (the time to plan and the time to execute) of a solution. Response time constraints further factor into the problem of meeting deadlines and the problem of providing optimal response times. In our commonsense formalism we have addressed the first aspect of response time constraints, namely, deadlines. Total response time includes both planning and execution time. Optimal response time is not necessarily the result of planning for an optimal solution (with respect to cost factors other than time), since the planning cost for it may be extraordinarily high. Typically, planning time and solution quality have a trade-off between them. This has been extensively studied in several works based on decision theory [Hor88, DB88, RW91]. They tackle the response time optimization problem, which is aimed at finding the best possible solution within the least amount of time. However, these works do not have the means to account for *all* the time spent in their reasoning, nor is it their main concern.

Dynamic situations are those where the world changes in the course of planning. Planning completely before the execution phase can lead to obsolete solutions. Typically, cycles of planning followed by execution are suggested as a solution to this problem [Kor90, BH91]. However, the time required to decide how much planning must be done before execution begins is not often taken into account.

6.0.2 Where the active-logic work fits in

Our work overlaps with the above concerns in real-time planning. We explore two particular aspects: 1. Real-time embeddedness 2. Deadline-coupled reasoning. The main contributions of this dissertation to AI planning are in these two areas. The first addresses the fact that the agent is embedded in a dynamic environment, and furthermore, ensures that all the time spent in the reasoning and acting is in the real

time-frame. The second addresses the issue of response time.

The decision theoretic works hinge on having a priori and complex knowledge of "utilities" associated with procedures, and on analyzing how differing time costs can affect the "optimal" choice of a decision procedure. Performance profiles graph the expected value of parameters returned by a given procedure as a function of time. Quoting from [BD94]: "knowing expected value is also in general insufficient; there is need for the complete probability distribution." This type of complex knowledge, sometimes represented by surfaces in three-space needs to be analyzed so that these techniques can yield significant incremental benefits due to deliberation.

We have not attempted to model complex deliberation that assumes the prior availability of detailed characteristics of the decision procedures. *In fact, we have (with the exception of some heuristic inference rules) not attempted to perform optimization. A plan that takes longer or consumes more material resources is as acceptable as one that takes less time or is more efficient, as long as it is feasible.*[1] However, as noted before, we meet important time-related aspects that these works only partially seem to: we account for all the time spent in the deliberation. This includes the time for reasoning, as well as the time spent in any meta level reasoning. Furthermore, we do so uniformly, without invoking any additional procedures or demons for the meta level thinking.

We propose a generalized mechanism for planning *in* time. The specific inference rules for the planning tasks themselves and algorithms developed here are not very sophisticated. But, for someone interested, there is nothing that prevents one from encoding richer planning structures into our framework while it still accounts for all the

[1] We remark here, that it would be possible to model detailed decision theoretic reasoning in the active-logic framework if the desire is to model an "expert" agent who is endowed with extensive prior information about the domain and about the decision algorithm characteristics. We have not made many attempts in this direction.

time spent in planning and acting. Moreover, apart from its capacity to integrate and interleave planning with execution, the planner has the ability to incorporate changes around it into its reasoning. The active-logic mechanism includes observations from the world into the belief set at the very moment that the agent takes note of them, and they continue to affect the ongoing theory being developed by the agent in real-time. Observations can be used to provide crucial feedback during the monitoring of an action and act as a valuable aid in error detection and recovery. We have not addressed the issue of plan failure and recovery in this work. But a time-bounded treatment of it using deadline-coupled inferencing would be a direct extension to our current work. Observations also serve to inform the agent of crucial events that will affect the future of its planned actions. Sometimes, observations can render the planning process unnecessary if the (sub)goal is observed to be already solved. At other times observations may completely upset the assumptions on which the plans were based, calling for much more planning effort.

6.1 Inference rules for fully deadline-coupled planning

This section contains a sample subset of domain-independent inference rules for the active logic for deadline coupled planning.

1. The agent makes an observation

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

An observation is incorporated into the context set of every plan p being processed. In particular the null plan maintains a context consisting of all observations and the theorems that come to be proven in this context. Note that this

rule is a modified form of the original step-logic observation rule which did not have to make the distinction between contexts. In the absence of any planning or temporal projection, the context of the null plan bears close resemblance to the belief set of the SL_7 logics of Elgot-Drapkin.

2. Forms the first partial plan(s) by finding a triplet for the goal

$$\frac{\begin{array}{l} i: \quad \mathbf{Now}(i), \mathbf{Goal}(i, G(S:F,\ldots), Deadline\_G), \mathbf{Unsolved}(i, G), \\ \qquad \mathbf{CS}(i, null, \{\ldots, Result(A_k, R_{A_k}), Condition(A_k, C_{A_k}), A_k \rightarrow G, \ldots\})\ldots \end{array}}{\begin{array}{rl} & \mathbf{Ppl}(i+1, p_k, \left\{ \left[\begin{array}{c} C_{A_k} \\ A_k \\ R_{A_k}(S_k : F_k \bullet\!\!\rightarrow F, \ldots) \end{array} \right] \right\}), \\ i+1: & \mathbf{CS}(i+1, p_k, \mathbf{CS}_{i,null})\mathbf{Proj}(i+1, p_k, \{\}) \\ & \mathbf{Feasible}(i+1, p_k)\mathbf{WET}(i+1, p_k, 0)\ldots \end{array}}$$

if $G \notin \mathbf{Proj}_{i,null} \cup \mathbf{CS}_{i,null}$.

When Dudley has a goal that is not currently being planned for, he develops the first partial plan(s) for solving it. For every available action A_k (or conjunction of actions) that solves the goal he generates a new plan and calls it by a name p_k. In short, corresponding to every axiom with the consequent G he performs backward reasoning to deduce the actions that must be done to achieve G. The time of the action is is linked the deadline by the "$\bullet\!\!\rightarrow$" symbol which denotes that the result of the action must be protected until the deadline. We give a simple example to illustrate this rule.

$\mathbf{5}: \mathbf{Now}(5), \mathbf{Goal}(5, At(10, dudley, home), 10),$

$\mathbf{Unsolved}(5, At(10, dudley, home)),$

$\mathbf{CS}(5, null, \{\ldots, Result(Walk(T1:T2, dudley, garden:home, Walk\_speed),$

$\{At(T+3, dudley, home)\}),$

$Condition(Walk(T:T2, dudley, garden:home, Walk\_speed),$

$\{At(T, dudley, garden)\}),$

$Walk(T : T2, dudley, garden : home, Walk\_speed) \rightarrow At(T2, dudley, home), \ldots\})$

$$\mathbf{6} : \mathbf{Ppl}(6, walk\_it, \left\{ \left[\begin{array}{c} At(T, dudley, garden) \\ Walk(T : T2, dudley, garden : home, Walk\_speed) \\ At(T2, dudley, home) \end{array} \right] \right\}),$$

3. Adds an action to the plan to satisfy a condition

$$\frac{i : \mathbf{Ppl}(i, p, \left\{ \ldots \left[\begin{array}{c} \{\ldots, C\} \\ A \\ R_A \end{array} \right] \ldots \right\}), \mathbf{CS}(i, p, \{\ldots, Q \rightarrow C\})}{i+1 : \mathbf{Ppl}(i+1, p, \left\{ \ldots \left[\begin{array}{c} C_Q \\ Q \\ R_Q \end{array} \right] \left[\begin{array}{c} \{\ldots, C\} \\ A \\ R_A \end{array} \right] \ldots \right\})}$$

if $C \notin \mathbf{Proj}_{i,p} \cup \mathbf{CS}_{i,p}$

For every condition C in the condition list of an action that is not projected to be true, if there is an axiom for satisfying it, Dudley adds the corresponding action to the plan. If there is more than one axiom for satisfying the same condition, Dudley formulates a plan for each possibility, and indexes the name of the partial plan with a new suffix to distinguish the new plans.

4. Refines a non-primitive action

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

provided every condition in $C_A \in \mathbf{CS}_{i,p} \cup \mathbf{Proj}_{i,p}$.

The active logic planner is hierarchical. Abstraction is embodied in the way the axioms encode the knowledge about actions. Skeleton plans at upper levels first synthesized by using higher level actions. These are then broken into more primitive actions by rules such as the action refinement rule described above. As the refinement progresses, better estimates of the execution time of the plan become available. The context set maintains the actions reasoned about at all levels. Further, these actions are used to annotate any reasoning based on them. Lower level actions are annotated by the higher level action that they refine (see the context set rule from Chapter 4). In the event replanning becomes necessary, this provides the mechanism to revise a plan by substituting an action and all the actions below it in the hierarchy when required. Our design allows for the concurrent processing of levels, and for concurrent refinement of multiple partial plans[2].

5. Includes a *Conditional* action in the plan

[2] More on restricting parallelism in Chapter 8.

$$\frac{i : \mathbf{Ppl}(i, p, \left\{ \ldots \left[\begin{array}{c} \{\ldots, C_{A_k}\} \\ A \\ R_A \end{array} \right] \ldots \right\}), \mathbf{CS}(i, p, \{\ldots, C \wedge Q \rightarrow C_{A_k}\})}{i+1 : \mathbf{Ppl}(i+1, p, \left\{ \ldots \left[\begin{array}{c} \{C\} \cup C_Q \\ Q \\ R_Q \end{array} \right] \left[\begin{array}{c} \{\ldots, C_{A_k}\} \\ A \\ R_A \end{array} \right] \ldots \right\})}$$

if $C \notin \mathbf{Proj}_{i,p} \cup \mathbf{CS}_{i,p}$

When an axiom $C \wedge Q \rightarrow C_{A_k}$ is found to be a way to satisfy a (sub)goal C_{A_k}, the action Q itself is not sufficient for C_{A_k}, C has to be true in the projection as well. This is taken care of by adding C in addition to the conditions for Q in the action triplet introduced in the plan. Later in Chapter 7 we elaborate on the planning options that depend upon the category of C.

6. Executes an action

$$\frac{i : \mathbf{Ppl}(i, p, \left\{ \left[\begin{array}{c} C_A \\ A(i : T, \ldots) \\ R_B \end{array} \right]_1 \ldots \right\}), \mathbf{CS}(i, p, \{\ldots, C_A\})}{i+1 : \mathbf{Ppl}(i+1, p, \{\ldots\})}$$

This inference rule executes an action when its start time has been bound to the current Now by the agent. The time for some actions is decided right when they are inserted into the plan, for others it must be decided by a specific inference rule. In our present implementation, we execute a primitive action as soon as its conditions are satisfied.

7. Includes a *Repeat-until* action in the plan with *signaling-condition* SC_A

$$\frac{i: \mathbf{Ppl}(i,p,\{\ldots \left[\begin{array}{c} C_B \\ B \\ R_B \end{array}\right]_j \ldots\}),\ \mathbf{CS}(i,p,\{\ldots, \mathit{Repeat\_until}(S:T,A,SC_A,\ldots) \rightarrow C_B, \mathbf{condition}(A,C_A)\})}{i+1: \mathbf{Ppl}(i+1,p,\{\ldots \left[\begin{array}{c} C_A(S:T,\ldots) \\ \mathit{Repeat\_until}(S:T,A,SC_A,\ldots) \\ C_B \end{array}\right]_{j-1} \ldots\})}$$

The above inference rule adds a repeat-until type of action to the plan. The condition of the repeat-until action is the condition for the repeated part, but maintained over the entire duration of the outer loop. More detailed discussion on this category of actions will follow in Chapter 7.

8. Executes of a *Repeat-until* action in the plan

$$\frac{i: \mathbf{Ppl}(i,p,\{\ldots \left[\begin{array}{c} C_A \\ \mathit{Repeat\_until}(i:T,A,SC_A,\ldots) \\ C_B \end{array}\right]_1 \ldots\})}{i+1: \mathbf{Ppl}(i+1,p,\{\left[\begin{array}{c} C_A \\ \mathit{Repeat\_until}(i+1:T,A,SC_A,\ldots) \\ C_B \end{array}\right]_1 \ldots\})}$$

if $SC_A \notin \mathbf{CS}_{i,p} \cup \mathbf{Proj}_{i,p}$

9. Completes execution of a *Repeat-until* action when a *signaling-condition* appears

$$\frac{i: \mathbf{Ppl}(i,p,\{\left[\begin{array}{c} C_A \\ \mathit{Repeat\_until}(S:T,A,SC_A,\ldots) \\ C_B \end{array}\right]_1 \ldots\}), \mathbf{CS}(i,p,\{\ldots,SC_A\})}{i+1: \mathbf{Ppl}(i+1,p,\{\ldots\})}$$

10. Spawns the generation of multiple plans on encountering a compound condition[3]

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

This inference rule encodes the linear planning strategy of our planner. Clearly, a total ordering such as this will cause the generation of non-optimal and sometimes even redundant plans. We have some heuristic inference rules that identify some obvious redundancies in a planner and identify the presence of loops. In general though, we have not focussed on plan optimization. One heuristic rule that identifies a redundant plan and rejects in favor of a better one is described below:

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

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

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

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

6.2 Examples of active-logic scenarios

Appendix A gives examples of some sample scenarios which are variations of Dudley's railroad track predicament. They bring out some of the interesting aspects of time-situated planning that we have discussed in this chapter.

6.3 Summary and related work

In this chapter we have described the inference rules related to plan formulation. In Chapter 7 we discuss the inference rules related to time estimation and feasibility analysis.

In this section we briefly describe the handling of interactions and dependencies in some well known planners. Mapped along an axis representing the amount of deliberation, reactive systems occupy the low end, while the works that we mention at the end of Chapter 7, which perform extensive deliberations and meta-reasoning are on the high end. In this section we also mention some reactive planning research as well as those planners that perform a modest amount of meta-reasoning but do not reason about the deliberations themselves.

6.3.1 Plan interactions and dependencies

The Sussman anomaly[Sus73] showed that certain planning situations are intrinsically non-linear. Waldinger [Wal75] first suggested the technique of "goal regression" to tackle the problem of conjunctive goals. With INTERPLAN [Tat75], Tate suggested recording a link between the effect of one action and the condition of another. (We have used a similar idea by using the "$\bullet\!\!\rightarrow$" symbol to record the need to preserve a certain effect of an action until a later time). NOAH's procedural nets, and SOUP (Semantics of User's Problem) [Sac75] used critics which are outside advisors that perform decision making regarding non-linearity and plan optimization. It was the

first partial-order planner. We have a total-order planner, simply because it turned out to be the simplest kind to build while we concentrated on the time-related aspects. We commit to a sequence of actions, but the actual times at which the action must be executed is bound to the *Now* only at the time of acting. NONLIN [Tat77b] could detect interactions and take the necessary corrective action. DEVISER [Ver83] went a step further and handled time limits while performing partial order planning. Planning with conditional operators and iterators has been dealt with in NOAH [Sac75], SIPE [Wil83b] among others.

There have been numerous efforts to improve planning by recognizing goal interactions and dependencies during the planning process, and better representations of actions and plans; we refer to [AHT90] for a collection of these. These researchers recognize the need to use features of the plan to reason about improving the plan. But this is not done by the planner itself. In our work, although we do not make any attempts to optimize plans, we perform domain-independent meta-level reasoning within the same framework as the object-level planning; and unlike the "critics" [Sac75], the meta-reasoning is an intrinsic part of the planning that also consumes time.

6.3.2 Approaches to handling meta-planning tasks

One way to implement meta-level decision making is to design two distinct component systems, one for object-level and one for meta-level reasoning. The other way is to design a uniform meta-level architecture where the meta-level problems are formulated using the same language and structures as the base-level problems, using similar methods. This introduces flexible systems, but along with it also introduces the possibility of infinite regress. This is the metareasoning challenge.

Reactive systems eschew meta-level planning, and even planning. by considering all contingencies at design time. Typical of this group is the work of Brooks [Bro91,

Bro86]. Other efforts that obviate the need for explicit reasoning at execution time are [AC87] and [RK86] and [Kae87]. This is an interesting approach. In the problems that we have dealt with in this dissertation, which fall in the "unforeseen" category, the fully reactive approach is often believed to lead to brittle and inflexible systems if no real-time deliberation is performed [DF89, Doy88, PR90, IG90].[4]

At middle of the deliberation spectrum, many researchers agree that some form of deliberation is necessary in planning. We mention a few of these here. (A detailed relationship with several approaches to associating a cost with the deliberation process is discussed at the end of Chapter 7.) SIPE [Wil83b] separated execution and generation by allowing the user to guide the planning process (perform the meta reasoning) during execution. The PRS system [GL88] uses metareasoning to recognize the need for additional planning. More recently [IG90] proposed a situated architecture for real-time reasoning. Based on PRS, it provides management representation of metareasoning strategies in the form of metalevel plans, and describes an interpreter that selects and executes them. The architecture is not *fully* embedded in real-time though, since the time of this interpreter is not accounted for.

Our *fully* deadline-coupled planner has an important qualification that these efforts fail to meet: in addition to performing metareasoning for determining the current time, estimating the expected execution time of partially completed plans and being able to discard alternatives that are deadline-infeasible, it also has 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! Active logics do this without a vicious circle of "meta-meta-meta..." hierarchies.

---

[4] Partial reactivity to the environment is achieved in our formalism by taking timely note of changes in the environment through observations. However, we simply incorporate the observations into the ongoing deliberative process of reasoning; they do not trigger any special reactive components.

Chapter 7

How long will it take?

7.1 Estimating the WET of a plan

A truly time-situated planner must be able to keep track of every unit of time spent, whether it is spent in inferential or physical activity. This also includes the time spent in making estimates of how much time will be spent. Thus, first of all, we need a time-situated estimation mechanism. The WET (working estimate of time) of a plan is a rough estimate of the total time that the plan will consume. It consists of two parts. The PET (*planning estimate of time*) is the (estimated) time spent in reasoning about the plan. This includes plan formulation, refinement, temporal projection and context-based reasoning. The EET (*execution estimate of time*) of the plan is the (estimated) time required to actually execute the actions that have been identified in the plan. Thus, WET = EET + PET.

The WET estimation consists of EET estimation and PET estimation. Our approach is to concentrate on the EET portion. We estimate the PET only by a small factor that is an estimate of how long it will take to refine the current plan to the level of primitive actions. Basically, for each action that is non-primitive, we add a constant number of time steps (default is two) that are required to firstly refine the action, and secondly to bind the variables for actual execution of the action. Thus, we

add at least $2n$ to the EET of the plan if there are n non-primitive actions currently in the plan. If a measure of the level of abstraction of an action is available (such as in the representation in the ABSTRIPS planner [Sac73], that number would reflect the number of steps required to refine the action into primitive level actions, and could be substituted as an estimate in place of the constant 2.

In our design, the decision to not include an estimate of the (future) meta-planning time[1] into the WET was taken to avoid recursion of meta-meta-meta ... levels of estimation. Time must be spent to choose between alternatives or to adjust the plan to ensure that it does not violate other goals [Wil83a]. Inferencing such as this constitutes the meta-planning that Dudley performs. We do account for the time spent in making these inferences, as they are made. But the WET is restricted to actions in the plan, namely object level actions. This is not a serious disadvantage. We have a uniform approach to treating planning and meta-planning. A meta-level plan will eventually be translated into an object level plan that satisfies the meta-goal. Once at this level, the WET will accommodate the execution time of the meta-plan into the new WET. We give a brief example to illustrate this.[2]

Suppose Dudley has a plan to go out and fetch the newspaper in the morning. However, on a particular morning, it is raining outside. The plan being developed to fetch the newspaper has the ramification that it will cause Dudley to get soaked, and violate the sustenance goal[3] to keep himself dry. He must then (meta) plan to try and still keep himself dry. The meta-reasoning results in an object level plan to wear a raincoat, which must be merged with the plan to fetch the newspaper. The new WET will continue to reflect only the execution time of the plan to walk outside

[1] Current and past meta-planning time is fully accounted in the sliding **Now** predicate and is factored already into the feasibility analysis.

[2] This example is mentioned in [Wil83a].

[3] A sustenance goal is one which must be preserved during the entire planning process.

and fetch the newspaper while the meta-planning proceeds in time. But once the object level plan to wear the raincoat begins to be synthesized, the WET reflects this additional time to look for a raincoat and put it on. As the meta-planning proceeds, time is consumed and is accounted for by the sliding *Now*. In this sense, we have a commonsense formalism for a fully deadline-coupled estimation of the WET. One may argue that the planning time may be too high, and if the WET can not factor that into the computation the estimates will be too low to be useful! That may very well happen if planning continues to introduce only inferential actions into the plan for which no object level estimation is available. With human reasoners, in case the problem involves very complex deliberations that are all inferential, they do not have any idea of how long their reasoning may take. So long as the agent can switch to reasoning about object level actions after a certain amount of thinking, the estimates will not be too low. Between these estimates and the accounting for how much *Now* has changed while making them, we feel that we have a reasonable estimation method for the WET. It has the advantage that it is a simple computation that does not require too much prior knowledge or tedious processing.

Note that the WET is only a rough estimate and hence feasibility conjectures based on it are at best approximate. The agent often tries to estimate an upper bound on the WET, so as to make sure deadlines are met. Deadlines may still be missed because: (1) The WET estimate was not accurate, individual components took longer to execute than expected. (2) The agent experienced sudden unexpected changes that rendered the planning obsolete. (3) Actions in the plan took their estimated time too execute, but, these actions *failed* and did not yield the expected results.

The following two rules compute the WET of a plan and check for its feasibility at *every* step in the reasoning.

- Computes the WET

$$\frac{i : \mathbf{Ppl}(i, p, \left\{ \begin{bmatrix} C_{A_1} \\ A_1(s_1 : f_1, \ldots) \\ R_{A_1} \end{bmatrix} \cdots \begin{bmatrix} C_{A_k} \\ A_k(s_k : f_k, \ldots) \\ R_{A_k} \end{bmatrix} \right\}) \quad \mathbf{CS}(\ldots, estimate(A_k, E_{A_k})}{i+1 : \mathbf{WET}(i, p, \sum_{j=1}^{k} E_{A_j})}$$

the summation is limited to those j values for which E_{A_j}[4] is known.

The WET is Dudley's calculation of how long his partial plan (formed as of the previous step) will take to refine and 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 the WET + Now is within the deadline, he declares it **Feasible**, and continues refining and/or putting the partial plan into execution.

- Keeps track of feasibility

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

if $n + i \leq Ddl$.

7.2 Categories of actions and time estimates for plans

As Dudley develops a partial plan to save Nell, he continuously refines his estimate of the time needed to carry the plan to completion.[5] In the begining phase of plan generation, actions are more complex and abstract. Estimates for the execution time

[4] E_{A_j} is the time estimate A_j. It includes the two steps required to refine A_j if A_j is non-primitive.

[5] The WET estimation 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.

for these actions are based on prior experiences with the action or with similar actions and are held as axioms of the form $estimate(< action >, < time\_to\_complete\_action >$). The estimates may also be acquired as a result of observation. As **Now** changes, and time is spent in planning, the agent substitutes lower level actions into the plan for which closer estimates may be available, up until the level of primitive actions where the estimate is simply the anticipated interval for executing the low level task. In general, the estimate may or may not be separable into individual constituents.

Dudley's database of axioms and rules contains knowledge about actions and their (intended) effects. However, not all actions are the same from the perspective of planning. Especially with regard to planning under time pressure, Dudley may have to estimate differently the time interval for the duration of each action in the plan, depending on the type of the action. We attempt here to formalize some categories of commonly encountered actions from the standpoint of planning.

- **The *Repeat_until* category**

  ($Repeat < action > until < signaling\_condition >$) is the form of an action that needs to be performed in a loop. In order to achieve a particular goal, the only known procedure may be to repeat a certain action or sequence of actions. Very often, there is an observation that signals the successful completion of the task. This observation (*signaling_condition*) may or may not coincide with the goal. For example, dialing the telephone repeatedly until a connection is obtained, is a means for establishing contact with someone. Similarly, beating egg whites until stiff peaks appear [Ger90] is a means for beating eggs to the right consistency. An agent must incorporate repeated actions into plans in many day to day situations. The *signaling_condition* is often known to the agent. The inference rules for plan formulation enable the agent to formulate a plan with a *Repeat_until* action, and guide the actual execution of this type of action.

A primitive action can be directly acted upon, and is removed from the plan upon its execution. A *Repeat_until* action is a non-primitive action which can be executed, i.e. the repetitive part of it can be executed, but is not removed from the plan unless the *signaling_condition* is observed.

For most actions that fall in this category, the agent may have an estimate of how long it may take until the *signaling_condition* typically appears. The agent knows what sequence to repeat, but does not know the exact number of times that it must be carried out. For example, in the case of beating egg whites, Dudley may know that this typically takes 3 minutes. If 6 minutes go by and stiff peaks do not appear, it signals a possibility of failure of the *Repeat_until* action[6]. The planning process as well as execution is incremental; the actual number of times the repeat is executed is determined in real-time through execution combined with observation.

There is a difference in the two examples of *Repeat_until* actions described above. In the case of the egg whites, it is necessary to repeat the action, not because it fails to give the intended result, but more because it is part of a sequence of actions that must be performed in order to achieve the goal. Here, the cumulative effect of the repeats marks the end of the loop. In the example of dialing until a connection is obtained, the agent keeps redialing because the earlier dialings fail. The first successful action marks the end of the loop. However, we will omit this distinction here, and put both examples in the *Repeat_until*[7] category,

[6]This paper does not offer a formal treatment of plan failure and recovery. An alarm mechanism can be built that signals a potential failure to Dudley in the event that he overshoots his estimate for a *Repeat_until* action by a substantial margin.

[7]Another real-time scenario which illustrates the use of a *Repeat_until* action, is one in which Dudley chases the bad guy in real-time. As the target object moves, Dudley must perform the action of taking one pace in the direction of the current position of the target. At every step 'now' and 'here'

since, from the planning agent's point of view the plan has the same structure and monitoring must be done to watch out for the *signaling_condition*.

- **The *Conditional_effect* type actions**

  An axiom for this type of action is often of the form:

  $C \wedge A \rightarrow R$

  $(if < condition(s) > and < action > then < result >)$,where R is a (sub)goal, and is the result of performing A, given that condition C is satisfied. The condition C must be held true in addition to the list of conditions C_A which are seen as necessary by the agent in order to be able to perform A.

  Within this category, there are two possibilities:

  (a) C is *do-able* under the agent's domain of control, i.e. the agent has an axiom of the form $B \rightarrow C$. The inference rule for planning for this type of axiom is as follows: To make a plan to achieve R, insert A into the plan, and add C along with C_A as the conditions in the triplet corresponding to result R. In our formalism, once inserted, C will also be continuously checked at each step, so that the plan may be altered in the event that C ceases to hold. Further,

  (i) If C is already true in the context of the plan, the agent does not have to plan additionally for it. The estimate of the time for achieving R in this case is not affected by the presence of C.

  (ii) However, if C is not already in the context of the plan, it will be necessary to add action B to the plan to first achieve C and then proceed with A. In this case the estimate should include along with the estimate of A, at least

---

must be used as parameters to create a new instance of a pace in the dynamic plan. Dudley may be able to estimate the time required to reach the bad guy from the differential in their respective speeds, and can use it to tailor his plan vis-a-vis the approaching deadline; but the actual number and specification of the paces must match the uncertainties in the changing environment.

two additional time steps: one is estimated for the addition of B to the plan, and at least one other for executing it. In this case, the estimate is $E+2$, where $estimate(A, E)$. In subsequent steps, as B is added and refined, a more accurate estimate can be obtained.

As an example of this category, consider this axiom from the Yale Shooting Problem:
$Loaded(T) \wedge shoot(T) \rightarrow \neg alive(T+1)$. If the goal is to kill Fred and $Load(T) \rightarrow Loaded(T+1)$ is known, a plan for killing must include along with the conditions for shoot, the condition that the gun must be loaded. Further, in the event that the gun is not already loaded, the $Load(T) \rightarrow Loaded(T+1)$ axiom suggests additional planning to this end.

(b) C is not *do-able*, but is merely *observable*, i.e. it is not under the control of the agent. Here there are two cases to consider:

(i) If C has already been observed and is projected to remain so, planning and time estimation can proceed as in case 2(a)(i).

(ii) C is an observable condition, but it is not known 'now' whether or not C is true. Then, Dudley must insert into his plan an action to observe whether C is true, and depending on the conclusion, insert A into the plan, in the event that C is indeed true. If several alternatives exist, based on several observable conditions, each suggesting different actions to be undertaken, he must postpone deciding between them for the present. However, he can use the possible list of alternatives to obtain a bound on the estimate, taking the alternative that has the maximum estimate into account.

As an example of 2(b)(ii) consider: Dudley can see that Nell is tied to the tracks, but can not tell from the distance what kinds of knots the bad guy has used in tying the ropes. He knows of the following common kinds of knots from his boy

scout days and of particular procedures employed in tying and untying them.

$Clove\_hitch \wedge Untie\_clove\_hitch(T : T+2) \rightarrow \neg Tied(T+2)$

$Timber\_hitch \wedge Untie\_timber\_hitch(T : T+2) \rightarrow \neg Tied(T+2)$

$Unknown\_knot \wedge Cut\_ropes(T : T+8) \rightarrow \neg Tied(T+8)$

$estimate(Untie\_clove\_hitch, 2)$,

$estimate(Untie\_timber\_hitch, 2)$,

$estimate(Cut\_ropes, 8)$

Since Dudley has no control over which knot he will encounter upon arriving at the tracks, he must plan for all contingencies. The decision regarding which action to insert into the plan must wait until the appropriate observation. He could plan to cut the ropes regardless, but that will take the longest time. Thus, by postponing his decision until run-time, Dudley may save on time. He must, however, make sure that the conditions corresponding to all the above alternatives will be satisfied at that point in time, if he wishes to keep the choices to the very end. He must therefore bring a knife to the tracks in case he will have to resort to cutting. He thus creates at this point a kind of pseudo-action (or meta-action) to insert into the plan. This action is the disjunct of all the alternatives. To supplement the plan to take that decision, Dudley inserts an *Observe* action into the plan,[8] which will itself take a time step. The estimate of the pseudo-action is taken to be the maximum of all its disjuncts. The result of the *Observe* action is unknown at the time of planning. At execution time, more will be known, and the pseudo-action can be substituted for the actual action applicable in that context.[9]

[8]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 ([KP89], [Dav88]) on ignorance and perception.

[9]This is also linked to the notion of plan commitment. This is a strategy to delay commitment

- **Actions with a simple formula for an estimate**

  These are the kind of actions for which the agent can determine a time estimate instantly, if an estimate for the *rate* of the action is known, and if an estimate for the *amount* of work to be done is also known. The *Run* action is in this category. Knowing the distance to Nell and his speed of running, Dudley can estimate how long his *Run* will take. But, it is possible that one or both, the distance or the speed may not be known. Dudley must carry out a deliberate observation or calibration to obtain these. E.g., Dudley looks out of the window and sees Nell tied to the rail tracks. He may not know the distance between his house and the tracks. Either he must look it up, ask someone, or do some calibration, such as simple trigonometry. His own running speed, he may know from past experience, or he may need to figure that out too, by running the length of his living room and timing himself as he does so. Such methods for obtaining estimates for actions in this category may be undertaken in circumstances where it is very crucial to obtain these estimates, and further decision making hinges on them. The cost of the more refined methods is obviously the time spent in obtaining them. In our simplified scenario, Dudley knows the distance to the rail track and his speed of running.

- **An action with a fixed interval between its start and finish times**

  This is the simplest category, and includes the kinds of actions which are relatively simple. These actions have a fixed duration. The estimate for an action in this category is the difference between its start and finish times where known. An example of this category of action is 'pull'. It takes one step to pull Nell from the tracks once she is untied.

---

until the last possible instant to allow for more flexibility in planning, of course at the cost of planning for all contingencies and allowing for the time in the on-the-spot decision-making.

For each category of actions described , we have described the inference rules for planning for the actions and for the estimation of the time required to carry them to completion. In most cases some form of a time estimate is either known from prior experience, or acquired from observation. In those cases where no known estimates are available to the agent, the unknown estimates are a measure of how much knowledge the agent has regarding the WET of the plan. The currently unknown estimates may potentially be a big drain on time. Dudley keeps a count of how many such unknown estimates exist in each plan. In ongoing work on this front, we are looking at 'agent attitudes' to characterize agents who can use this and other uncertainty information along with perhaps some on-line utility computation, to perform a primitive risk analysis as a basis of choosing between plans which have the same time estimates. An agent who is *risk averse* may choose to go with a plan that is better known even if it has a large WET.

7.3 A discussion on other meta-level reasoning tasks

In the previous section we described the WET computation which is the component of the inferencing crucial to deadline-coupled planning. Several other meta-level decisions must be taken under time-pressure. We have addressed these to a very limited extent. We describe them briefly.

7.4 Goal interactions

Dudley may concurrently have several goals and subgoals. It is likely that the actions in one plan may contradict other goals. Since our reasoning is confined to the context of the plan, we do not currently have a way to handle complex goal interactions between several plans. Within a plan, the effect that the goals have on each other is recorded in the context set and reasoned about. Goal interactions must be reasoned about using

meta-plans. Some of the agents goals are sustenance goals [Wil83a] and therefore beliefs regarding them appear in the context of every plan. We can reason about sustenance goal conflicts without significantly enhancing our framework. Whenever an action in a plan contradicts a sustenance goal, Dudley tries to construct a meta-plan to protect that goal. Which in turn causes the generation of an object level plan which must be carefully merged with the current plan. In the example we mentioned before, the sustenance goal to keep himself dry causes the generation of the object level plan to wear a raincoat, which must be merged with the plan to go out and fetch the newspaper.

7.5 Choosing between plans

When more than one partial plans are being formulated, a meta-level task is to decide which of them to work on. Since the active logic allows simultaneous generation of plans, the question of which one to continue to *reason about* is not relevant except when we have a limited memory. We will discuss heuristics to guide this decision later in Chapter 8. With the active logic we have described thus far, the dilemma of choosing arises when one or more plans are ripe for action (contain primitive actions whose conditions are satisfied). Actions are interval formulas. They may have specific times of execution such as "Wake up at 7:00 am". This action will be performed at 7:00 am and is linked to the *Now* explicitly in its definition. Other actions may have variable times that have to be bound to the current time when the agent decides to act.

The agent's choice of acting on an action in one plan v/s the other is guided by several criteria:

- Does performing one action alter the physical state of the world so that the other one can no longer be executed? (Are they mutually exclusive?)

- Is it possible to perform both actions concurrently?
- Is the action reversible? If it becomes necessary to backtrack on the plan, can it be undone?

  Although not a part of this work, an obvious but non-trivial extension to the active logic would be to perform commonsense reasoning of the above nature before choosing one of many plans that are currently ripe for action.

7.5.1 To act now or to plan further

A related issue in meta-planning is to take the decision regarding "acting now" or delaying action until more reasoning about the plan gives better measures of success. This is an issue that has received extensive attention in works such as [Hor88, BD94]. Often this involves reasoning about the probability of success and a numerical measure of the utility of various actions. One can implement algorithms such as these in a time-situated active-logic framework. We currently adopt the simple strategy that was adopted very early on in one of the first works that attempted to interleave planning and acting [McD78]. Whenever Dudley is ready to act, he performs the action.

Issues to think about in this regard are:

- Will there be lost opportunities? Will it be difficult to perform the same action later?
- Will performing this action result in more informed planning? Will it help to choose alternatives and to get better estimates?
- Is the action reversible?

Taking an action often reduces unknowns. Besides, as is clear from our detailed discussion of the frame problem, it is the only way to test whether it will succeed.

7.6 Summary and related work

This chapter describes some important meta-level tasks. The estimation of WET and checking of feasibility involve inferences that are drawn at every step in deadline situations. We showed how knowledge of action categories can help in estimating EET of the plan. We briefly discussed some issues in more elaborate meta-level decision making.

An excellent survey of research in deliberative real-time AI is available in [GL94]. They categorize real-time systems into *purely reactive* (those that hardwire reactions completely), *combined response* systems (those that have distinct asynchronous components that handle deliberation and reaction) and *integrated* systems (those that have a single architecture that is capable of a wide range of timely responses depending upon the time criticality requirements).

Those in the last category put the time that is available in the best use. These approaches have been collectively characterized by terms such as *flexible computation* [HR91], *deliberation scheduling* [BD94], and *anytime algorithms* [DB88, ZR92]. They spend the resources available to the agent in deciding whether to act, how to act, and when to act. The main differences between our approach and these is the following: (1) they do not account for the time-cost of the *deliberation scheduling algorithms* themselves, only for the cost of deliberation that they consider; while our mechanism is completely situated in time; (2) they require prior complex (meta) knowledge about their reasoning algorithms or procedures themselves, and their characteristics with respect to time; they also require a great deal of knowledge about the domain in the form of probabilities of events and expected utilities of actions that the agent must be aware of; (3) they usually attempt to solve an *optimization* problem in a specific domain, whereas our approach is to come up with a formalism that accounts for all the time spent between Now and the deadline while attempting to reason about the

feasibility of a solution, not to find an optimal solution. Thus, we note that these approaches are not *alternatives* to our time-situated reasoning approach using active logics, but rather that they are suited for a different range of more informed problem solving.

Below, we give a more detailed account of a few works in this deliberative category.

7.6.1 Anytime algorithms

This now widely used term was first coined by Dean and Boddy [DB88]. It represents a class of algorithms which have the following characteristics:

(1) They can be interrupted at any time and will produce *some* solution to the problem; (2) given more time they will produce better solutions; and (3) the user of the algorithm has some explicit characterization of the tradeoff between the algorithm's performance and the amount of time that is available to compute a solution.

Anytime algorithms are similar in spirit to the notion of "imprecise computation" commonly used in research in operating systems which divides the task into a *mandatory* part that gives a solution and an *optional* part that refines this solution. The problems that have been attempted using the anytime technique have the flavor of more traditional optimization problems, which by themselves cannot cover the space of planning problems. The feature of "interruptibility" of anytime algorithms is not particularly one of great value in deadline-coupled planning in commonsense scenarios involving hard and non-extensible deadlines. We want Dudley to come up with a feasible solution by the deadline (if possible). We do not care if at any point of time he has found an approximate solution of an inferior quality. Having that assurance is not crucial.

7.6.2 Flexible computation

Work by Horvitz et al [Hor88, Hor89, HR91] attacks problems that may be classified as "high stakes decision problems". A typical example is in the medical domain where the decision making is complex, but highly informed. Most of the options and quantified information regarding relationships among decisions and propositions is available in the form of *influence diagrams*. Horvitz et al have also addressed the problem of dividing computational resources between meta-reasoning and object-level problem solving, particularly in the case when both are being solved using anytime algorithms [HB90]. By the use of mathematical functions which assume particular forms for the various utilities, they manage to keep the meta-reasoning cost quite small or constant.

Lesser et al [LPD88] provide a *design-to-time* approach which combines existing multiple methods for various tasks to maximize the quality of the combined solution within the available time.

In an approach that is inspired by economics, Etzioni [Etz91] addresses the problem for a time-constrained agent using special terms commonly used in economics. When a particular resource is available in limited quantity, it renders competing actions mutually exclusive. He defines an opportunity cost for each action, which is the maximum of the utilities of the other contending actions. He suggests a heuristic to choose the action with the highest marginal utility, without assuming prior knowledge of the utilities. There is a learning mechanism that calculates them through repeated executions. It seems that it would be possible in principle to implement Etzioni's methods within our active logic framework.

Lastly we mention work in the direction of building systems and architectures that exhibit desirable real-time behaviors, although not all components of these systems function in the real-time domain: Guardian [HRWA$^+$92] Phoenix [HHC90] and PRS [IG90]. FORBIN [DFM88] which is a planning architecture that supports hierarchical planning involving reasoning *about* deadlines, travel time, and resources are some

examples of such systems. TILEWORLD [PR90] is a simulated dynamic and unpredictable parametrized agent and environment. It is possible to experiment with the behavior of the agent and various meta-level strategies by tuning parameters of the TILEWORLD system. Once again, although all the reasoning here is not performed in real-time, many of their observations, especially regarding the manifestation of agent attitudes through the tuning of parameters could be of use in the development of an active logic where the active logic can self-adjust its parameters to the environment to decide the level of risk or deliberation it can perform.

Chapter 8

Towards realism: Limited space and computation capacity

8.1 Resource limitations

We have thus far described our design using active logics to tackle the fully deadline-coupled reasoning problem. It has also been described in [KNP90, NKP91]. The original step-logic work suffered from the *litterbug* problem. In a step-logic with the inheritance property, the agent's finite set of beliefs increases steadily as time goes by. Space is the other crucial resource that the agent must carefully reason with, while it reasons about time in a time-situated manner. Another problem that has its beginings in step-logics is the issue of *unbounded parallelism*. If unlimited computation power can not be assumed, it must be reasoned about and controlled. In this chapter we describe these shortcomings and provide meaningful solutions that brings the formalism closer to realism.

Studies in cognitive science that suggest a break up of human memory into two memory stores: a short-term memory (STM) and a long-term memory (LTM) dates back to William James (1842–1910) who first introduced the dual memory concept. This organization is based on the assumption that processing of information is treated

in a short-term store which operates independently, but is in constant contact with the knowledge stored in the long-term store. Also, new incoming information is used to alter and enrich the content of the long-term store. The amount of information stored in the STM is small compared to that in the LTM. Sir William Hamilton was the first to have recorded evidence of the limited size of the STM. Amazingly, the capacity of the STM seems to be independent of the type of data involved, or the length pf the actual strings. Miller [Mil56] offered an explanation. He postulated *chunking* of data into units which could occupy a slot in the STM. This increased the capacity of the STM in what we would today call the physical capacity measured in bits and bytes.

We suggest a memory model that is inspired by cognitive psychology to solve the space problem faced by Dudley as he reasons with active logics. A beliefs can be roughly thought of as a chunk.

8.2 Litterbugs and parallelism

The space problem: As time advances, more knowledge is gathered as a result of observations from the agent's environment and as a result of the deduction processes within. The knowledge base which is continuously expanding could potentially become so formidable that it would be completely unrealistic to assume that the agent could possibly apply all the inferences to this complete knowledge base. Usually, most of this information is not directly relevant either to the development of the agent's current thread of reasoning. Active logics and our treatment of deadline-coupled planning so far have disregarded the space problem in preference to dealing squarely with time-related issues. The space issue deserves serious attention where the original number of beliefs of the agent is large, and where very many new beliefs are added to the agent's knowledge base over time.

Unrealistic parallelism: A step is defined as the time required by the agent to

perform one inference or one primitive physical action in the world. Actions can be carried out in parallel if the sensors and effectors permit. For example, an agent can walk and eat simultaneously. Active-logics planners treat 'think' actions within the agent in the same spirit as physical actions and recognize that they sap precious time resources. The original step-logic inference system assumed that during a given step i the agent can apply *all* applicable inference rules in parallel, to the beliefs at step $i-1$. There are two problems with this. One is the unrealistic amount of parallelism that potentially allows the agent to draw so many inferences in one time step that the meaning of what constitutes a step begins to blur. Secondly, it is unreasonable to expect that all inference rules would have the same time granularity. For example, it is unlikely that a simple application of Modus Ponens will take just as long to fire as an inference rule to refine a plan, to perform temporal projection using the suggested algorithm or check for plan feasibility, especially as plans become very large. While the representation is uniformly declarative, some rules have more procedural flavor than others, and can be imagined to take more time. Just as there is a limit on the physical capabilities of the agent as to how many physical actions can be done in parallel in the same time step, there must be a limit to the parallel capacity of the inference engine as well.

8.3 The step-logic meaning of a step and realism

In Elgot-Drapkin's work [ED88a] the issue of space and computation bounds was recognized partially while defining a step. She says "because there are a growing number of inferences in later steps, there would not be a one-to-one correspondence between steps and actual time elapsed in an implementation; the length of time taken to make all deductions for a given step would in general grow in later steps." A possible way to deal with the space problem is to vary the duration of each step, as suggested

above. There are several potential problems with this. All estimates in the framework are made in terms of steps. If the step duration increases arbitrarily, these estimates would go haywire. There would be no relation with the deadline which is an absolute parameter in the framework. Secondly, the logic uses time parameters that quantify the history in terms of "one step later". If the step duration itself is variable, this quantification in terms of incremental steps will be inaccurate, if the "plus one" has a different meaning depending upon what step number the agent is reasoning with.

We suggest a solution that does not change the mapping between a step and real-time (absolute time) as the reasoning progresses. Instead we follow another suggestion made in [EDMP87] to keep the total number of beliefs in each step under check. We also propose a means of addressing the parallelism and granularity of the inference process.

A claim towards fully deadline-coupled reasoning would be a tall one if the model depicts an agent with an infinite attention span and infinite think capacity. In this chapter we propose an active logic extension of the original step-logic formalism to take into consideration space and computation constraints. We revisit the fully deadline-coupled planning problem in the light of this new framework.

8.4 A limited span of attention

We propose a solution to the space problem partially based on [EDMP87] as follows. The agent's current focus of attention is limited to a small fixed number of beliefs forming the STM (short term memory), while the complete belief set is archived away in a bigger associative store, namely, the LTM (long term memory). In addition, we use a QTM which is a technical device to hold the conclusions that result in each step. This is a temporary buffer that holds them since further inferencing with these must

wait until the next time step. The size of the STM is a fixed number K[1].

In the most simplistic model, the STM could be represented as a queue, in which case the inference/retrieval algorithm reduces to a simple depth first or breadth first strategy depending upon whether new observations and deductions are added to the head or tail of the queue, respectively. It seems that choosing the STM elements without focus consideration may lead the reasoning astray quite easily, and often lead to incomplete threads of reasoning due to thrashing. We propose to maintain a predicate called **Focus**(...) which keeps track of the current line of reasoning. This is dynamically changed by the agent's inference mechanism and is responsible for steering the reasoning back to a particular thread even when a large number of seemingly irrelevant inferences are drawn. Among the agent's inference rules is a set of *focus changing (FC)* rules, which when fired alter the focus. Those K beliefs from the associative LTM which are most[2] relevant to the current focus are highlighted to form the STM.

In short, the framework can be described as follows. The $QTM_{i/i+1}$ is an intermediate store of formulae that holds theorems derived through the application of inference rules to the formulae in STM_i (the STM at step i). They are candidates for the STM at step $i+1$, although only K among them will be selected. Thus the results of the inference rules, can be imagined to fall into $QTM_{i/i+1}$ and are available for selection to form the STM at the next step[3]. The $focus$ and Now which are crucial to time-situated reasoning are always accessible to the agent.

---

[1]What is a realistic K for a commonsense reasoner? There is psychological basis that suggests that human short-term memory holds seven-plus-or-minus-two 'chunks' of data at one time [Mil56].

[2]There is a ranking among the relevant formulae and the K at the top of the list are picked. In our implementation, we select the K formulae at random from the candidate formulae.

[3]This has the feature that all thinking does not pass through the STM unless it is relevant to the focus.

FRAMEWORK:

$$\frac{\mathbf{i} : \mathbf{STM_i}\{...\}, \mathbf{Now}(i), \mathbf{Focus}(i, ...), \mathbf{LTM_i}\{...\}}{\mathbf{i+1} : \mathbf{STM_{i+1}}\{...\}, \mathbf{Now}(i+1), \mathbf{Focus}(i+1, ...), \mathbf{LTM_{i+1}}\{...\}}$$

$QTM_{i/i+1}$ holds β if β is an i-theorem. It includes relevant formulae which are retrieved from the LTM using the retrieval rule. Step i concludes by selecting K formulae from $QTM_{i/i+1}$ which are relevant to $Focus_i$ to form STM_{i+1}. LTM_{i+1} is LTM_i appended with $QTM_{i/i+1}$.

The main problem in limiting the space of reasoning is to decide what should be in the focus. Within our planning framework, we have developed a mechanism that is at work to limit the focus to a single feasible plan at a given time step. A list of actions, conditions and results from the plan that need further processing (we call it the active list), form a list of keywords in the focus. We describe some details of this mechanism in section 8.6. Heuristic rules are proposed to maximize the probability of finding a solution within the deadline. This would correspond to a kind of best first strategy or a beam search of width K in the general framework. Although these heuristic rules are independent of the instance of the problem in question, they are likely to be different depending upon the category of the problem being solved. A deadline-coupled actor-planner is likely to maintain a much narrower focus than a long-range 'armchair' planner. In section 8.6, we outline some of the specific heuristic strategies employed for the tightly time-constrained planner.

8.4.1 A limited think capacity

Next, we address the bounded computation resource problem. An intelligent agent can be expected to have a sizable reservoir of inference rules acquired during its lifetime. Firing of an inference rule corresponds to a 'think' action. Without a bound on its inferencing power, the agent could fire all the inference rules applicable (termed in conventional production systems as the conflict set) simultaneously during a time

step. We limit the inference capacity of the engine to I. Each inference rule j is assigned a drain factor d_j. This is a measure of the drain incurred by the inference engine while firing an instance of this rule. For instance, Modus Ponens and the more elaborate inference rule for plan refinement, would be given different drain factors to reflect this difference in granularity [4].

Our limited-capacity inference engine fires only a subset of the applicable rules in each time step. Among the various alternatives, it is possible to pick the inference rules either completely nondeterministically up to the engine capacity I, or one could again apply some heuristics to improve the agent's chances. Several parameters, such as agent attitudes, the uncertainty of the environment, or the urgency to act could dictate this choice.

Thus, in effect, during each step, K beliefs are highlighted from the knowledge base (LTM) to constitute the STM. From among the rules applicable to these K beliefs, a subset of rules is chosen such that sum of the drain factors does not exceed the engine's inference capacity I. The results of the inferencing are put in the QTM. Finally, the contents of the QTM are copied to the LTM.

8.5 On the adequacy of the limited memory model

Let $SL(OBS, INF)$ denote a step-logic with an inference function INF, an observation function OBS and unlimited memory described in the original framework. Let $SL_K^{FET}(OBS, INF)$ denote the corresponding active logic with a limited short-term memory of size K and an algorithm FET describing the strategy for fetching elements

[4] How to calibrate the inference rules for the assignment of these drain factors is a separate and interesting issue, but we will not address it presently. Also, how thinking actions compare with physical actions is a technical issue that could be resolved by trying to calibrate the system to check on the relative speed of its inference cycle with that of its sensors and motors. We skip this implementation sensitive issue.

into the STM.

Theorem 8.1 If all the inference rules in INF are monotonic then it is possible to describe a simple algorithm FET such that any theorem of $SL(OBS, INF)$ will eventually appear as a theorem of $SL_K^{FET}(OBS, INF)$. I.e. if $\vdash_i \alpha$ in SL (α was proven at step i) then $\exists j$ such that $\vdash_j \alpha$ in $SL_K^{FET}(OBS, INF)$.

Note: the requirement of monotonicity in particular entails that the "clock"-rule for Now is left out. Thus the result applies only to Now-free inferences. We also assume that new observations are consistent with previous facts and derivations.

Proof We begin by showing that the following dovetailing transformation on INF into $Dove[INF]$ yields an equivalent step-logic with unbounded memory in terms of the final theorem set. We then show that $SL(Dove[INF], OBS)$ has the property that $SL_K^{FET}(Dove[INF], OBS)$ has the same final theorem set as $SL(Dove[INF], OBS)$ for the algorithm FET described below.

Let all the rules in $Dove[INF]$ have at most two antecedent formulae. This is achieved by transforming every rule in INF which is of the form:

$$\frac{A_1 \wedge A_2 \wedge \ldots \wedge A_n}{W}$$

into n rules of the form:

$$\frac{A_1 \wedge A_2}{P_3}$$

$$\frac{A_3 \wedge P_3}{P_4}$$

$$\vdots$$

$$\frac{A_n \wedge P_n}{W}$$

This can make the number of rules very large, but it allows us to have a very simple algorithm FET to show that there is no net loss of theorems on account of limiting the size of the STM. Let $InfCh_\alpha$ denote the inference chain used to derive a theorem α. The algorithm FET proposed uses dovetailing to ensure that STM cycles through all possible combinations of beliefs, so that eventually whatever formulas are used in $InfCh_\alpha$ also occur in STM.

We want, first, to ensure that the algorithm has access to all logical axioms, so we feed them in lexicographically. At each step, some more (finitely many) logical axioms are added to the LTM; we feed in all formulas of length $\leq i$ at step i, that use only the first i symbols of the language. FET forces STM to cycle through all combinations of beliefs in LTM, all combinations of two at a time, to allow every rule that could fire to actually fire.

As time goes on, more and more logical axioms are fed into LTM, and also new inference results are being produced and going into LTM. FET is an algorithm that gets every combination of two formulas, including new ones that come in by either inference or feeding. We can conceptualize all formulas (in the entire language) to be already in LTM but only those that occur in $InfCh_\alpha$ to be marked in red. As time goes on, more and more become red, due to inference and feeding. We also mark each formulas with an index (a unique natural number), and bring into STM two at a time, but only red ones, and never repeat a pair already brought in; we can imagine each pair of formulas has a link that become blue when it is brought into STM; so we never bring a blue-linked pair in again. At each step we bring in a non-blue pair and apply all applicable rules to it. One could either bring in the pair with the smallest index-sum, the pair with the largest index sum or pick a pair at random, among many alternatives.

This simple algorithm FET will perform all possible inferences including those in $InfCh_\alpha$ eventually deriving α, although after many more time steps. □

Remark 8.2 The inference rules used for deadline-coupled planning are nonmonotonic. The rules to calculate WET, to refine a plan, to revise the context of the plan and to project the context set of the plan are the main nonmonotonic inferences drawn in this system. In checking if a condition c already appears in the context or the projection of a plan before the plan is refined, the rule for plan refinement is nonmonotonic since c may be absent when checked but present later. The nonmonotonicity of the projections stems from applying the default of temporal persistence. The context set revision rule is nonmonotonic since it withdraws from the context set formulae whose basis is no longer true in the current projection. In $SL(OBS, INF)$ where there is no limit on the memory, partial plans get refined whenever possible, and the context set is revised at every time step, also, the projection in the latest context is recomputed at every time step. With $SL_K^{FET}(OBS, INF)$ however, the order in which these rules will get fired depends upon the simultaneous occurrence of the matching formulae in the STM. If the refinement of the partial plan proceeds before the context set revision, it is possible that redundant plans will be developed before the context set and the projection will catch up to let the planner know that something is already true and does not need to be planned for. As an example, consider a plan in which a condition for a certain action requires that a certain high-rise building be pink. Dudley may have an axiom which says that all high-rises are pink, but has not had a chance to apply it to the context set in question to conclude that the high-rise building in question is pink. Hence he formulates a (redundant) plan to paint the high-rise pink. Subsequently, as the context set is revised this condition is already true in the projection and an inference rule needs to be fired to identify and eliminate this portion of the plan.

We note here that redundant plans of this nature may be generated even in the case of unlimited memory, if there is a long inference chain based on the facts required to derive the condition in question. For example, if Dudley does not directly know that

all high-rises are pink, but infers it from the fact that all high-rises are tall structures, that all tall structures are made of concrete, that anything made of concrete is pink. Since bringing in all these axioms and revising the context set may take quite a few steps, it is likely that redundant plans are generated even in the case of unlimited memory. A rule that corrects this situation is useful in both cases.

Remark 8.3 If the inference rules for plan refinement, for context set revision and for computing projection compute all possible instances of **PPl**, **CS** and **Proj** instead of working on the *latest* instances alone, then all the partial plans generated by the unlimited step-logic will also be generated by the bounded active logic.

When each instance of the partial plan, context set and projection is kept active, the same combinations that occur in the unlimited step-logic will eventually cycle through the STM, giving the same partial plans, in addition to several other plans generated.

Remark 8.4 In some cases, where context-set revision precedes planning, in fact efficient plans may be generated since the planner is in fact more informed about extended effects and side effects prior to the planning[5].

[5]In the FET algorithm, bringing the lowest sum of indices corresponds to a breadth-first strategy. Using highest sum of indices would correspond to a depth-first strategy. The former will ensure that partial plan refinement will not get too far ahead of the context set revision. Once the partial plan is refined, a context set revision rule must fire since its antecedents were already present in the LTM. In the depth-first method, you will refine a plan as far as possible, then revise CS as much as possible, then project as much as possible, and then alternate. It is interesting to explore how these will interact. In a random strategy that exhaustively brings in all pairs, arbitrary speeds of the three chains need to be considered to see its effects on the planning.

8.6 Heuristic strategies for deadline-coupled planning

In the previous section we presented some formal results on the adequacy of an active-logic to generate the required plans even when there is a bound on the size of the STM. The algorithm *FET* was a simple breadth-first strategy used to demonstrate this. In this section we present heuristic algorithms to be used in place of FET to improve the chances that the formulae used in the derivation of the plan will appear sooner in the STM.

8.6.1 Focus and keywords

As a general approach to limiting space, we proposed that beliefs be organized in the LTM by association with some topics or keywords. When one or more of these topics are in the focus, the related beliefs become candidates for retrieval into the STM, as a result of a retrieval rule. Formulae in the STM are not automatically inherited from one step to the next. Only when they are still relevant to the current focus do they become candidates and must compete with other relevant formulae to fit into the limited size STM.

The focus holds the keywords of current interest. It has similarities with the RTM proposed in [EDMP87]. We imagine that in a more general framework the focus would contain keywords arranged in a partial order according to priorities.[6] Beliefs related to high priority topics are given preference for being brought into the STM. As mentioned before, for our actor-planner Dudley we restrict the focus to equal priority keywords related to a single plan at a given time step. Non-primitive actions that appear in the triplets of a given plan, that still need to be refined are appropriate keywords for goal-directed retrieval. Conditions of actions which are not yet satisfied, as well as, the

[6]The question is how to choose the "keywords" that are in the focus at a given time, and how to assign priorities to them. Our ideas presented here are aimed at a commonsense agent engaged in deadline-coupled planning.

results that appear in the triplets of the plan serve as keywords to deduce the effects of the plan. These are kept in the focus as the formula $plan\_in\_focus(p, PKWL)$ where p is the name of the partial plan and $PKWL$ is the list of keywords for p.

Observations are put into the current focus at least for a few time steps, since it is possible that they may be important, and may trigger some new threads of reasoning[7]. Current observations are kept in the focus as the formula $obs\_in\_focus(OBL)$ where OBL is the list of observations that serve as keywords. Together, the focus is a predicate $\mathbf{Focus}(i, plan\_in\_focus(p, PKWL), obs\_in\_focus(OBL))$ at step i.

When there are multiple options in the STM for achieving a goal, more than one partial plan is spawned. All plans for achieving a certain goal may be given equal priority at first, thus continuing to develop them in a time shared manner and bringing them into focus sequentially. However, in a deadline situation, it may be advisable to commit to a plan (to put it in focus and the others in a background queue for backtracking) and continue with it unless it seems infeasible.

The development of an appropriate plan depends on three aspects: satisfying (pre)conditions, refinement of general actions into more detailed ones, and using the result of the plan for finding its effects.[8] To illustrate, given a triplet in the $\mathbf{Ppl}[C_A, A, R_A]$, the following axioms are used: axioms that produce C_A (i.e. C_A appears in their conclusions), axioms that are used for refinement of A (again A appears in their conclusions) and axioms in which R_A appears in their antecedents (i.e.

[7] How to in fact select some crucial observations from all the stray input to the sensors remain unaddressed, but it is not among the problems we will solve at present. A tutor or a human hint to the automated agent that some observations are worthy of more consideration. In our example, Dudley may first start to think about running to Nell to rescue her, when he suddenly sees a telephone. This brings 'calling', and subsequently the related axiom of calling the driver to stop the train into focus. This spawns the generation of a second plan.

[8] [WHR89] have discussed the relative merits of refinement into lower levels v/s searching to greater depth to find the right action.

$R_A \rightarrow e$), to compute the extended effects. Therefore, if we make sure these types of axioms are brought into the STM, we will increase the probability that a plan will be found. Note, that other axioms should be used for finding indirect effects (i.e., $e \rightarrow e1, e1 \rightarrow e2...$) which can't be obtained by the agent's action. Our heuristic brings such axioms to the STM too.

8.6.2 Some inference rules for resource limited reasoning

At each step, the agent reflects on its long term memory reservoir to pick out formulae that are relevant to its current focus of reasoning using a retrieval rule. The LTM is an associative store and hence this retrieval is fast.[9]

Focus directed retrieval rule (FDRR):

$i : ..., \mathbf{LTM}\{..., \beta, ...\},$

$$\frac{\mathbf{Focus}(i, plan\_in\_focus(p, PKWL), obs\_in\_focus(OBL)), ..}{\mathbf{QTM_{i/i+1}}\{..., \beta, ...\}}$$

if β is relevant to either p or a keyword in $PKWL$ or OBL.

In our work on planning, the **Focus** includes keywords related to a *feasible* plan. A (partial) plan is *feasible* if the sum of Now and the plan's working estimate of time is still within the deadline. A list of feasible partial plans is maintained. From among these a subset of plans is selected to work on and is called the interleaving list (IL). Dudley works on each plan in the interleaving list for a *period* number of steps, then goes on to the next plan in the IL in round robin fashion. The interleaving rule (ILR) serves this purpose by periodically selecting the next plan in the IL to put into the focus. This is one of the focus changing (FC) rules in Dudley's inference engine[10].

[9] The retrieval rule is a weak parallel of the inheritance rule in Elgot-Drapkin's step logics, in the sense that formulae in the STM at the previous step reappear in the STM at the current step provided they are *still* relevant.

[10] Probably, other scheduling procedures that were developed by operating Systems researchers can

This rule time-shares between plans and always fires. A separate rule controls the contents of IL.

Interleaving Rule (ILR):

$$\frac{\mathbf{i} : \mathbf{Now}(i), \mathbf{IL}([p_{j_1}, ..., p_{j_n}]), ...}{\mathbf{i+1} : \mathbf{Focus}(i+1, plan\_in\_focus(p_{j_1}, ...), ...), \mathbf{IL}([p_{j_2}, ..., p_{j_n, p_{j_1}}])}$$
$$\text{if } \mathbf{i} \bmod \text{period} = 0$$

When there are two or more plans in the **IL**, and when it is time to choose between them, a rule fires to narrow the focus to only one plan. We stipulate that the difficult problem of 'when to decide to choose' depends on mental states and attitudes of agents [Sho91]. A more 'cautious' type of agent will skeptically continue to process two alternatives, perhaps risking overshooting the deadline, but a more 'dashing' type of agent will take the risk to pursue just one plan. We have developed a heuristic rule under the following commonsense observation: An agent can continue to work on several plans provided there is *ample* time ahead to try and pursue them one after another in the interest of fault tolerance. For example, even after calling the driver to stop the train, Dudley may want to run to the railroad track and attempt the rescue Nell nevertheless, if there is enough residual time. An agent may do so as a guard against possible failure of his own or other agents' plans, or perhaps as an extra precaution when the plans are not recognized to be mutually exclusive. We look then at the sum of the WET's of all the plans in the IL as a measure of the overhead planning time. When the sum of the WET's and Now exceeds the deadline, he drops a plan from the IL. We currently have the simple heuristic of dropping the plan with the highest WET, but recognize that this may very well be the most refined plan as well[11]. Additional bookkeeping is necessary to ensure that two rules do not alter the IL

be used here,but it is beyond the scope of our paper. We only demonstrate how such procedures can be used *in* time.

[11]If one can find a way to include good estimation of the planning time (and probably decision time)

or the focus simultaneously. We skip these implementation details in this description. Reduce IL rule (RILR):

$$\frac{\mathbf{i} : \mathbf{Now}(i), \mathbf{IL}(L), \mathbf{wet\_ordering}([p_{j_k}, ...]), ...}{\mathbf{i+1} : \mathbf{IL}(L - p_{j_k})}$$

if $\sum_{l \in L} \mathbf{WET}_{p_{j_l}} + \mathbf{Now} > Deadline$.

An agent may be forced into a decision if two or more plans are ripe for action and the actions are mutually exclusive. The agent must evaluate the relative merits of the plans before making the decision, if acting on one will commit the agent to one plan. Although we do allow planning and acting to be interleaved, we allow the agent to act on a plan if it is the only one in IL. This is to avoid the complex interactions between plans as the result of the changed state of the world following the execution of one plan. We continue to examine this issue in ongoing work.

8.6.3 Capacity of the inference engine

As mentioned earlier, we suggested a limited capacity inference engine that would fire a cumulative set of inference rules to not exceed its inference capacity in each time step. In the simplistic examples that we present, there is a very limited number of rules firing at each step. Furthermore, if the plan length is within a reasonable bound, drain factors of the rules are also quite small and as a first approximation we postulate them to each take roughly the same time and fire in parallel in a single step whenever applicable. It should be noted that the meta rules for resource limited reasoning which were described above fire alongside the other object level inferencing at each step as

into the WET it seems that more refined plans will have less planning time than other plans. Maybe, the three parts of the WET should not be combined and the decision whether to knock out a plan from the IL should be made using some sort of multi attribute decision rule (i.e., based on executing time, planning time and decision time).

part of a uniform framework. If we limit the capacity of the engine, the meta rules that are fired will limit the number of planning rules that are fired in each step.

8.6.4 Some illustrations from two plans

Dudley begins to formulate a plan *save* to get Nell *Out_of_danger*. Initially, the focus consists of $\mathbf{Focus}(j, plan\_in\_focus(save, [Out\_of\_danger(...)]), ...)$, and the interleaving list is $IL([save])$. Here, *save* is the name of the partial plan and is used to retrieve formulae related to the plan such as its WET, its context set, projection etc. The list of keywords for this plan contains Out_of_danger. It is used to retrieve axioms from the LTM whose right hand side matches the keyword. At step i, the plan *save* bifurcates into $save_1$ and $save_2$ based on the following axioms which are retrieved from the LTM:

$Pull(T : \overline{T+1}, Y, X, L) \rightarrow Out\_of\_danger(T+1, X, L)$

$Stop\_train(T : \overline{T+2}, driver) \rightarrow Out\_of\_danger(T+2, nell, r)$

Plan 1: Pull her away from the tracks

$$\mathbf{Ppl}(i, save1, \left\{ \begin{bmatrix} \neg Tied(t_1, n, r) \\ Pull(t_1 : \overline{t}_2, d, n, r) \\ Out\_of\_danger(t_2 \bullet\!\!\rightarrow Deadline, n, r) \end{bmatrix} \right\}),$$

$$\mathbf{CS}(1, save1, \{\ldots, t_2 \leq Deadline, t_1 = t_2 - 1\})$$

Plan 2: Stop the train

$$\mathbf{Ppl}(i, save_2, \left\{ \begin{bmatrix} Knows\_about(\tau_1, n, dr) \\ Stop\_train(\tau_1 : \tau_2, dr) \\ Out\_of\_danger(\tau_2 \bullet\!\!\rightarrow Deadline, n, r) \end{bmatrix} \right\}),$$

$$\mathbf{CS}(i, save2, \{\ldots, \tau_2 \leq Deadline, \tau_1 = \tau_2 - 2\})$$

The interleaving list is expanded to contain both $save_1$ and $save_2$ and Dudley continues to work on both feasible plans in a time-shared fashion. The focus thus contains $save_1$ for an interleaving period during which axioms for untying Nell and running to her are progressively retrieved from the LTM. Other facts of no relevance to the plan

such as $color\_of\_eyes(\ldots)$ or that are relevant to the other plan such as the axioms about dialing to get a connection are left alone in the LTM. After the period expires, $save_2$ is brought into focus and worked on in a similar fashion. It is later, at step j, that Dudley realizes that the sum of the WET's of both plans and **Now** is going to overshoot the deadline, and he must restrict the IL using the RILR rule. We show a snapshot of the two plans when this happens:

$$\mathbf{Ppl}(j, save_1, \left\{ \begin{array}{c} \left[\begin{array}{c} At(t_6, d, home) \\ Run(t_6 : \overline{t}_7, d, home : r) \\ At(t_7 \bullet\!\!\rightarrow t_4, d, r) \end{array} \right] \\ \left[\begin{array}{c} At(t_3 : t_4, d, r) \\ Release(t_3 : \overline{t}_4, d, n, r) \\ \neg Tied(t_4 \bullet\!\!\rightarrow t_1, n, r) \end{array} \right] \\ \left[\begin{array}{c} \neg Tied(t_1, n, r) \\ Pull(t_1 : \overline{t}_2, d, n, r) \\ Out\_of\_danger(t_2 \bullet\!\!\rightarrow Deadline, n, r) \end{array} \right] \end{array} \right\})$$

$$\mathbf{Ppl}(j, save_2, \left\{ \begin{array}{c} \left[\begin{array}{c} At(\tau_9, d, nh) \\ Run(\tau_9 : \overline{\tau}_8, d, nh : r) \\ At(\tau_8 \bullet\!\!\rightarrow \tau_7, d, nh) \end{array} \right] \\ \left[\begin{array}{c} At(\tau_6 : \tau_7, d, nh) \\ Dial(\tau_6 : \overline{\tau}_7, d, dr) \\ In\_contact(\tau_7 \bullet\!\!\rightarrow \tau_4, d, dr) \end{array} \right] \\ \left[\begin{array}{c} In\_contact(\tau_3 : \overline{\tau}_4, d, dr) \\ Warn(\tau_3 : \overline{\tau}_4, d, dr, n) \\ Knows\_about(\tau_4 \bullet\!\!\rightarrow \tau_1, n, dr) \end{array} \right] \end{array} \right\})$$

Dudley develops two alternative plans in a time-shared fashion. Suppose the parameters in the problem (namely, the deadline, Dudley's speed of running, the distance to the railroad track and to the neighbor's house, etc.) are such, that at step j shown above, the sum of the WET's of the two plans is no longer within the deadline. Dudley

exercises a choice through the rule RILR which reduces the interleaving list to the plan to call the driver of the train. (Abbreviations used are: n=nell, d=dudley, r=railroad track, nh=neighbor's house, and dr=driver of the train.)

Suppose *save1* is the plan with the higher WET. Using this heuristic, Dudley gives up the plan to run to Nell, and executes the plan to go to the neighbor's house to call the driver to stop the train. With the sum of the WET's exceeding the Deadline, Dudley starts to run in the direction of the neighbor's house and removes $save_1$ from the IL, still retaining it in the list of feasible plans to be available in case of unanticipated run-time failure.

8.7 Summary

In this chapter we have presented a revised formalism for planning with active logics that incorporates concerns of limited time, space, and computation. We have shown the adequacy of this model in deriving the same set of theorems as the model without space limitations. We have developed some heuristics for the deadline-coupled planner that will allow it to function within tight resource limitations.

Chapter 9

A modal semantics for active logics

Most formal approaches do not have an appropriate representational framework to tackle time-situated reasoning problems such as the ones we have mentioned in this dissertation. They assume that an agent is able to reason forever in a timeless present as if the world had stopped for the agent's benefit. Resource limitations have been of some concern in formal work. In particular, the problem of *logical omniscience* has received attention in the epistemic logic literature. It concerns the difficulty with the classical Hintikka possible world semantics [Hin62] that the agent always knows the logical consequences of her beliefs. However, no existing works provide a semantics addressing the issue of how the reasoning progresses vis a vis the passage of time. Although work in temporal logic involves reasoning *about* time (e.g., [All84, McD82, MS87]), time is not treated as a crucial resource that must be carefully rationed by the agent, as it is spent in every step of reasoning.

Step-logics [EDP90, PEDM] as a formal apparatus to model an agent's ongoing process of reasoning were described in Chapter 2. We have thus far described an active logic based *fully* deadline-coupled planning and reasoning mechanism which is a combination of declarative and procedural approaches. Although active logics have been characterized and implemented, only limited attempts have been made to give a formal semantics for the step-like reasoning process.

This chapter is intended to bridge the gap between previous modal approaches to knowledge and belief and time-situated frameworks such as step-logics which have a means for attributing time to the reasoning process. We discuss the various aspects of logical omniscience and their treatment in section 9.1. In section 9.3 we present a modal active-logic that is step-like in spirit and motivated by the work on step-logics, but for which, unlike the former, we can provide a sound and complete modal semantics in section 9.4. In section 9.5 we examine how our approach addresses the logical omniscience problem and summarize our contribution.

9.1 The various aspects of omniscience and its treatment

Fagin and Halpern [FH88] have analyzed what is meant by the notion of *logical omniscience*. They define an agent to be logically omniscient if whenever he believes formulas in a set Σ, and Σ logically implies the formula ϕ, then the agent also believes ϕ. They further identify three cases of special interest: (a) *closure under implication*, namely, whenever both ϕ and $\phi \rightarrow \psi$ are believed then ψ is believed, (b) *closure under valid implication*, namely, if $\phi \rightarrow \psi$ is valid and ϕ is believed then ψ is believed and (c) *belief of valid formulas*, namely, if ϕ is valid, then ϕ is believed.

The agent in the classical model of knowledge [Hin62] has all the undesirable properties (a), (b) and (c) above. Several improvements have been suggested, and they have been broadly classified as "syntactic" and "semantic" approaches. In the syntactic approach e.g. [Ebe74, MH79], what the agent knows is represented by a set of formulas and hence is not constrained under consequence. But such approaches are difficult to analyze, since they are not guided by knowledge-based principles. A commendable syntactic approach is presented by Konolige in his *deduction model* [Kon86a] which gives a formal characterization of explicit beliefs and captures how agents syntactically

derive new beliefs, possibly with an incomplete set of inference rules.

In contrast, semantic approaches attempt to give semantics similar in most cases to the possible world semantics, but with "fixes". Levesque [Lev84] gives a semantic account of *implicit* and *explicit* belief where implicit beliefs are the logical consequences of explicit belief. A solution to (a) and the possibility of having contradictory beliefs is achieved by introducing an artificial notion of *incoherent or impossible worlds.* Levesque's approach was subject to the criticism that an agent in the logic is a perfect reasoner in relevance logic. Levesque's ideas have been extended in [PS85] and [Lak86]. Montague has given a possible world semantics that gets around problem (a) of logical consequence. We use the main idea in this model, namely, to define knowledge as a relation between a world and a set of sets of possible worlds. However, we provide the distinction of incorporating time-situatedness. Vardi [Var86] provides a co-relation between restrictions on models in the Montague semantics and the corresponding agent properties that they characterize.

Fagin and Halpern [FH88] have presented a series of interesting approaches to limited reasoning that marry the syntactic and semantic approaches. They provide an extension to Levesque's approach for the multi-agent case, and introduce a notion of *awareness.* They also provide an approach to local reasoning that they call a *society of minds approach.* Fagin and Halpern's awareness notion, in their logic of general awareness acts like a filter on semantic formulations. It has been evaluated and criticized in [Kon86b]. One of the criticisms is that the model is unintuitive, since it is unlikely that an agent can compute all logical consequences, discarding the one's that it is not aware of, say, because of memory limitations, because in fact, agents are also affected by time limitations. There are a number of works that have considered logics of knowledge and time e.g. [Sat77, Leh84, KL88, LR86, Ash88]. Fagin and Halpern discuss the possibility of capturing bounded and situated reasoning by letting the awareness set vary over time. However, no attempt has been made to systematically

study and model situations where the passage of time is a critical issue.

9.2 Step-logics and the omniscience problem

As described in 2, Elgot-Drapkin characterized an array of eight step-logics in increasing order of sophistication with respect to three mechanisms : self-knowledge (S), time (T) and retraction (R)[1]. According to this classification, SL_5 is the simplest dynamic deductive logic with time and self-knowledge capability, but no retraction mechanism (no ability to handle contradictions). An SL_5 logic is a triple $\langle \mathcal{L}, OBS, INF \rangle$ where $\mathcal{L}$ consists of propositions (with the addition of time), OBS is an observation function describing inputs from the world at each step. For simplicity we first consider an agent who does not acquire any new beliefs through observations. INF is a set of inference rules. We describe an SL_5 step-logic to which we provide a modal active-logic analog. The set INF for it is shown in in figure 9.1.

We have chosen the simplest SL_5 since our main interest is in the treatment of time and in modeling agents with nested beliefs. We will impose an additional constraint on models that does not allow for contradictions in the agent's beliefs[2].

9.3 A modal active-logic for reasoning *in* time

With SL_5 as the motivation, we provide a time-situated modal logic. This modal logic is based on Montague's intensional logic of belief [Mon70], that uses structures referred to in the literature as *neighborhood structures* or *minimal structures* [Che80]. They

[1] SL_0:none; SL_1:S; SL_2:T; SL_3:R; SL_4:S,R; SL_5:S,T; SL_6: R,T; and SL_7:S,T,R.

[2] However, this condition may be relaxed if for example, we desire to model an agent with default reasoning capability. Step-logics are inherently nonmonotonic and allow for implicit and explicit contradictions in the agent's reasoning. The modal logic approach which is motivated by the step-logic work is powerful enough to deal with contradictions.

$$\frac{i\ :\ \ldots\quad \alpha,\beta\quad \ldots}{i{+}1\ :\ \ldots\quad \alpha\wedge\beta\quad \ldots} \qquad \text{Conjunction}$$

$$\frac{i\ :\ \ldots\quad \alpha\wedge\beta\quad \ldots}{i{+}1\ :\ \ldots\quad \alpha\quad \ldots} \qquad \text{Detachment}$$

$$\frac{i\ :\ \ldots\quad \alpha\quad \ldots}{i{+}1\ :\ \ldots\quad \alpha\quad \ldots} \qquad \text{Inheritance}$$

Figure 9.1: Inference rules for an SL_5 logic

were first used in [Mon68] and in [Sco70]. Montague gives a possible world semantics to epistemic logic where, unlike in the classical model[3], knowledge is defined as a relation between a world and a set of sets of worlds. An *intension* of a formula ϕ denoted by $||\phi||$ is the set of worlds w such that $w \models \phi$.

We prefer to use *timelines* instead of possible worlds, since this gives us a way to naturally incorporate time into our framework. L denotes the set of timelines [TSSK91]. We consider time lines that are restricted to be finite from one side and infinite from the other (i.e., are rays). At every time point in each timeline some propositions are true and the rest are false. In particular, there is one timeline of most interest, that captures the *real* history of occurrences in the world. We call this line $l_h \in L$ the *history timeline*.

[3]The classical possible-worlds model is based on the idea that besides the true world, there are other possible worlds, some of which may be indistinguishable to the agent from the true world. An agent is said to *believe* a fact ϕ if ϕ is true in all the worlds that she thinks possible. A semantics based on Kripke structures for this classical model suffers from the well known drawback from the point of view of logical omniscience that $K\phi \wedge K(\phi \rightarrow \psi) \rightarrow K\psi$ is an inherent axiom.

9.3.1 Syntax and semantics

In the logic proposed, the agent reasons in a propositional language with time. The interest is in sentences such as:

p: Nell is tied to the railroad tracks at *3 pm*.

q: Dudley is at home at *3:30 pm*.

Formally, we assume that there is a set P of atomic propositions. Let $\mathcal{N}$ denote the set of numerals, namely, "1", "2", "3", etc. The relation "<" denotes the normal total order on the set $\mathcal{N}$. We define $PT = P \times \mathcal{N}$ as the set of propositions extended to include time constant arguments. The elements of PT are the atomic formulas of our language, and we will denote them as $p(\tau)$ where $p \in P$ and $\tau \in \mathcal{N}$. The language $\mathcal{G}$ is the smallest set that contains PT, and is closed under the $\neg, \wedge, \vee, \rightarrow$ connectives, and contains $B_\tau \phi$ whenever ϕ is in the language and $\tau \in \mathcal{N}$. The B operator denotes belief .[4] This language can easily be extended to include multiple agents, by the use of an additional parameter i, so that $B^i_\tau \alpha$ denotes "at time τ agent i believes in α", where α may include beliefs of other agents.

A structure in the proposed time-embedded active logic is: $M = \langle L, \mathcal{N}, v, <, \pi, \mathcal{B} \rangle$ where

- L is a set of timelines.
- $v(n \in \mathcal{N}) = n$ is the interpretation function for time point constants.
- $\pi : PT \times L \rightarrow \{true, false\}$ is a truth assignment to the formula $p(t) \in PT$ for each timeline $l, l \in L$ at the time point $t \in \mathcal{N}$. Thus π defines the intensions of the atomic formulas of our language.

[4] In this language one can express formulas such as p and q above, belief formulas such as $B_{\tau_1} p(\tau_2)$ to mean "at time τ_1 the agent believes that p is true at time τ_2", or nested beliefs formulas such as $B_{\tau_1}(B_{\tau_1+2} p(\tau_2) \vee B_{\tau_1+2} q(\tau_3))$ to mean "at time τ_1 the agent believes that two time points later she will believe $p(\tau_2)$ or she will believe $q(\tau_3)$".

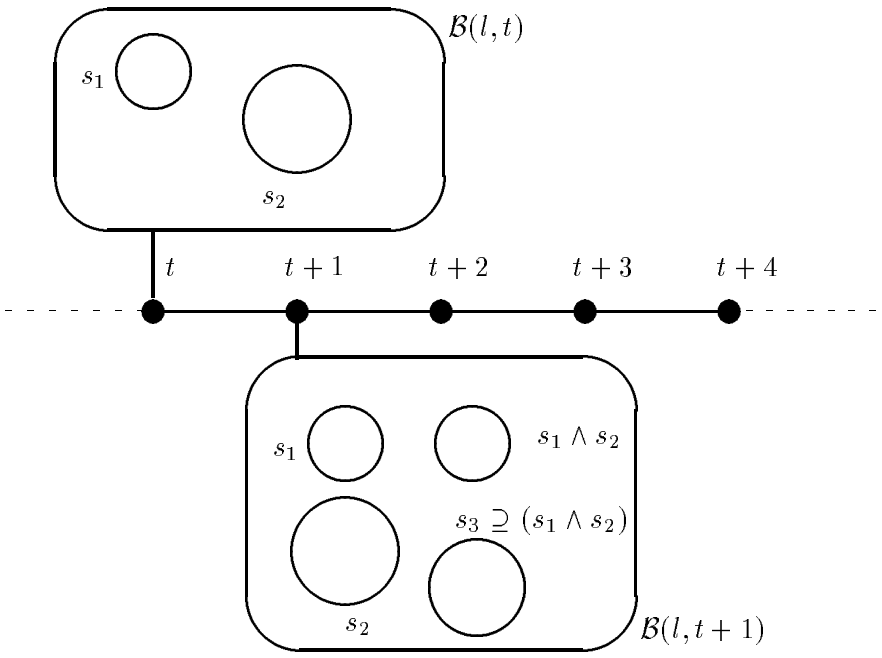

Figure 9.2: Neighborhood structures for the belief operator

- $\mathcal{B} : L \times \mathcal{N} \rightarrow 2^{2^L}$ is a belief accessibility relation, defined for each timeline, time point pair $(l,t), l \in L, t \in \mathcal{N}$.

We will use $\mathcal{B}_t(l)$ to denote $\mathcal{B}(l,t)$, which is the set of sets of time lines related to l at time t through the $\mathcal{B}$ relation. Note the use of the pair (l,t). We are interested in epistemic behavior over time, and this is depicted by the evolution of beliefs (and the corresponding accessibility relations) from (l_h,t) to $(l_h,t+1)$ in the real timeline.

Analogous to the Montague intensional logic, we define $B_\tau\phi$ to denote that an agent "believes a formula ϕ at time τ" and define a satisfiability relation for timelines based on intensions. An intension of a formula ϕ in a structure M denoted by $||\phi||$ is $\{l \mid l \in L,\ M,l \models \phi\}$.

Figure 9.2 illustrates the neighborhood structures for our modal logic[5].

[5]We comment here that is is possible to extend the modal active-logics to multiple agents reasoning in time. A structure for an active logic with multiple agents is $M = \langle L, \mathcal{N}, v, \pi, \mathcal{A}, \mathcal{B}^1, \ldots, \mathcal{B}^n \rangle$. L is

We impose restrictions on models to reflect the step-like reasoning behavior between successive time instances. These restrictions make certain axioms sound in our system. We further characterize the modal active-logic by a sound and complete set of axioms and inference rules. Time is an essential resource in this framework and is consumed in the reasoning process. This logic captures the reasoning process of a non-omniscient resource-limited agent.

We formally define $\models$ for the structure $M = \langle L, \mathcal{N}, v, <, \pi, \mathcal{B} \rangle$ described above as follows:

1. This defines satisfiability of the atomic formulas of our language.

   $M, l \models p(\tau)$ if $\pi(p(\tau), l) = true$.

2. This defines satisfiability of negated formulas.

   $M, l \models \neg\phi$ iff $M, l \not\models \phi$.

3. This defines satisfiability of formulas formed with the $\wedge$ connective.

   $M, l \models (\phi \wedge \psi)$ iff $M, l \models \phi$ and $M, l \models \psi$.

4. This defines the satisfiability of the belief formulas.

   $M, l \models B_\tau \phi$ iff $||\phi|| \in \mathcal{B}_\tau(l)$

The satisfiability of $\vee$ and $\rightarrow$ is defined accordingly. We impose the following restrictions on our models to describe an agent who reasons in a step-like fashion like its motivating step-logic agent described by SL_5.

- **(C0)** $\forall l \in L, \ \ ||true|| = L \in \mathcal{B}_0(l)$.

---

the set of time lines and π is the truth assignments to base formulas as before. $\mathcal{A}$ is the set of agents $\{1, \ldots, n\}$, and each of $\mathcal{B}^i, i = 1, \ldots, n$ associates with a timeline, time point pair (l, t), a set of set of timelines that are belief-accessible from l at time t from the perspective of agent $i \in \mathcal{A}$. In problems such as the three wise men problem mentioned in the introduction, a multi-agent logic where the time of all agents increments synchronously can provide an elegant solution to the problem.

Since the set $\mathcal{N}$ is ordered under $<$, and timelines are defined as rays, we can define a start point 0. This restriction says that the agent believes in *true* at the beginning of time. We note here that this restriction makes the axiom **A0** (to be described) sound in the system. It also dictates that the agent always has a nonempty set of beliefs.

- **(C1)** $\forall l \in L, \forall t \in \mathcal{N}, \ \{\} \notin \mathcal{B}_t(l)$.

  This says that the agent's belief set is consistent at every time point. As explained before, we introduce this restriction to model a simple agent without contradictory beliefs and without any mechanisms for retraction[6].

- **(C2)** $\forall l \in L, \forall t \in \mathcal{N}$, if $s_1 \in \mathcal{B}_t(l)$, and $s_2 \supseteq s_1$ then $s_2 \in \mathcal{B}_t(l)$.

  This restriction says that a detachment of a belief is necessarily also a belief at the same time step (since supersets correspond to detachments in the step logic).

- **(C3)** $\forall l \in L, \forall t \in \mathcal{N}$, if $s_1, s_2 \in \mathcal{B}_t(l)$, then $s_1 \cap s_2 \in \mathcal{B}_{t+1}(l)$.

  This restriction constrains models at successive time points to be one step richer than their predecessors, in the sense that the agent has added all possible pairwise conjunctions of previous beliefs to the current step, but each pair participates just once[7].

---

[6]This restriction can be relaxed if the intent is to model a fallible agent who does default reasoning and may be permitted to have contradictory beliefs at any given time. Without the above restriction the neighborhood structures possibly allow for both $M, l \models B_\tau \phi$ and $M, l \models B_\tau \neg\phi$, since both $||\phi||$ and $||\neg\phi||$ could belong to $\mathcal{B}_\tau(l)$.

[7]For example, if $M, l \models B_\tau \alpha$, $M, l \models B_\tau \beta$ and $M, l \models B_\tau \gamma$ then $M, l \models B_{\tau+1}(\alpha \wedge \beta)$, $M, l \models B_{\tau+1}(\alpha \wedge \gamma)$ and $M, l \models B_{\tau+1}(\beta \wedge \gamma)$ but $M, l \models B_{\tau+1}(\alpha \wedge \beta \wedge \gamma)$ does not follow from this restriction, however $M, l \models B_{\tau+2}(\alpha \wedge \beta \wedge \gamma)$ does.

Axioms :

(**A0**) $B_0 true$.

(**A1**) All tautologies of propositional logic.

(**A2**) $\neg B_\tau false$. Consistency

(**A3**) $B_\tau\phi \wedge B_\tau\psi \rightarrow B_{\tau+1}(\phi \wedge \psi)$. Conjunction[a]

Inference Rules :

(**R1**) If $\vdash \phi$ and $\vdash \phi \rightarrow \psi$ then $\vdash \psi$ Modus Ponens

(**R2**) If $\vdash \phi \rightarrow \psi$ then $\vdash B_\tau\phi \rightarrow B_\tau\psi$ Closure under valid consequence

(**R3**) If $\vdash \phi$ then $\vdash B_\tau\phi$ Belief in tautologies

[a] Inheritance follows from (A3) when $\phi = \psi$.

Figure 9.3: Characterization of the modal active-logic

9.4 Soundness and completeness proof for the modal active logic

Theorem 9.1 : The set of axioms (A0–A3) and inference rules (R1–R3) in Figure 9.3 provide a sound and complete axiomatization of the modal active-logic for reasoning in time.

Proof (sketch) : Soundness follows in a straightforward fashion from the interpretation of $\wedge$ and $\neg$ in the definition of $\models$ and from the restrictions on the models described. The proof of completeness hinges on the definition of a *canonical* model M^c in which every consistent[8] formula is satisfiable.

In M^c we have a timeline l_V corresponding to every *maximal consistent set* V. For definition and properties of maximal consistent sets we refer to [HY92]. Let I_ψ denote

[8] A formula ϕ is *provable* if ϕ is one of the axioms or follows from provable formulas by application of one or more inference rules. A formula ϕ is *consistent* if $\neg\phi$ is not provable.

the set $\{ l_W \mid \psi \in W \}$. The canonical model is defined as: $M^c = \langle L^c, \mathcal{N}, v, <, \pi, \mathcal{B} \rangle$ where

$L^c = \{ l_V \ : V$ is a maximal consistent set $\}$,

$\pi(p(\tau), l_V) = true$ iff $p(\tau) \in V$, and

For $t \in \mathcal{N}$, $\mathcal{B}_t(l_V) =$

$\{ I_\psi \mid B_t\psi \in V \} \cup \{ S \mid S \supseteq S', S' \in \{ I_\psi \mid B_t\psi \in V \}$ and $\forall \psi, \psi \in \mathcal{G} \rightarrow S \neq I_\psi \}$.

We then prove using induction that $M^c, l_V \models \phi$ iff $\phi \in V$, which proves that all consistent formulas are satisfiable in this structure. We provide the details of the proof in the appendix.

9.5 Summary and discussion

Active logics capture the process of reasoning of a resource-limited agent as it goes on in time. As time progresses, the agent draws more inferences (new beliefs) at each time step. Thus, an agent does not draw all the consequences of its current set of beliefs Σ all at once, but continues to add conclusions to this set in accordance with a set of inference rules. This is reflected by the increasing size of $\mathcal{B}(l_h, t)$, where l_h denotes the *real* history of occurrences in the world, and $\mathcal{B}(l_h, t)$ reflects what the agent believes in time t. The agent is certainly not guilty of omniscience under (a) logical consequence[9] since it is trivial to provide a counter-model to $B_\tau\alpha \quad B_\tau(\alpha \rightarrow \beta), \quad \neg B_\tau\beta$. By virtue of a description that is based on *intensions* of formulas, it is difficult to distinguish between semantically equivalent beliefs. As such, (c) belief of valid formulas and (b) closure under valid consequence follows.

[9]The agent may eventually compute all logical consequences of its belief set if it has a set of complete agent inference rules.

However, it is possible to modify our logic by providing a syntactic way to curtail the size of the belief set by introducing an additional element G to the structure M. $G \subseteq \mathcal{G}$ is defined as the agent's language and is closed under subformulas. An agent believes in ψ (i.e., $B_\tau \psi$) only if $\psi \in G$. For this new structure, the set of axioms and inference rules are suitably modified to capture this change (e.g., in (A3) $\phi \wedge \psi \in G$ and in (R3) $\phi \in G$ is added) and appropriate restrictions are placed on M. In essence, $\mathcal{B}_t(l)$ sets are filtered by G for all t and l. It can be proven that the modified set of axioms and inferences are sound and complete with respect to the modified structure. If the model includes more than one agent, each of them may have a different language G. This restricts an agent who believes in ϕ, to only that subset of $[\phi]$ (the equivalence class of ϕ) which is in the agent's language. The agent also believes only those tautologies that are in G. Hence the scope of (b) and (c) is reduced in the modified structure. The agent's language G has similarities to the awareness set concept of [FH88]. If one considers multiagent belief operators $\mathcal{B}^i$ without a time parameter then a modified version of Axiom (A2), and rule (R3) from figure 9.3 are true in the model of local reasoning of [FH88], (without modalities for implicit belief). Note, that we have only *explicit* beliefs, and there is no notion of *implicit* beliefs. In [FH88] the models are still static, in that even though they suggest incorporating reasoning *about* time, and *changing* awareness functions, there is no way to account for inference time in their models.

Chapter 10

Conclusions and future directions

We have designed a declarative planning framework based on active logics that treats planning as an action, and can account for all the time taken to plan, making it suitable for deadline situations. This effort led to a novel treatment of temporal projection and other frame problems, in a real-time setting; a modal semantics for active logics; and other results in reasoning under resource limitations. Some aspects of the design were implemented and tested in prolog.

The *active-logic* approach of accounting for *all* the time spent in acting and deliberating is innovative, and opens up many interesting agendas for future research. Following are some of the research directions that we have identified.

10.1 Future directions

1. *Failures in the physical world*

   While our formalism has room for fallibility in the agent's thinking (default conclusions are constantly withdrawn in the face of contradictory observations or more informed reasoning); mechanisms to monitor the success of the agent's actions in the real world are yet to be incorporated. This is extremely important for a real-world implementation integrated with the appropriate sensor/motor

actions.

2. *Soft deadlines and timing faults*

   We have dealt with an agent in a tight (hard) deadline setting. We wish to supplement this research with a treatment of soft deadlines, where not everything is lost if the deadline is overshot, but rather, the agent must evaluate the negative cost of doing so in the decision making. He/She may formulate a plan to stretch the deadline, but this formulation is also subject to the *same* time constraints as other efforts to meet the deadline.

3. *Multi-agent and distributed reasoning*

   In a time-situated setting, where several agents may need to interact from time-to-time, an agent may find it necessary to keep an account of others' time of reasoning in addition to his/her own. There are many interesting planning problems that evolve when actions of other agents must be interleaved in an agent's own plans.

4. *Deadlines and space-limitations*

   Deadline-coupled planning with the goal of keeping the available memory for plans within a fixed bound opens up an area for interesting experimentation. Experimenting with the value of the parameter used for the size of the short term memory and the associated heuristics for bringing "relevant" information from long-term memory for current plan refinement, could give us insight into the size of an STM for an automated agent that gives desirable behaviors.

5. *Perceptual reasoning*

   It would be desirable to extend the existing work in temporal reasoning to include perceptual reasoning, spatial reasoning and to aspects of planning that involve planning to acquire more information.

6. *Learning aspects and case based real-time planning*

   An agent who can keep track of the planning process employed in a novel deadline situation in a manner which will enable its efficient reuse at a later time can be said to *learn*. One possibility is to distill this knowledge into *cases*. Plan reuse and repair can then be explored for a deadline-coupled reasoner with this capability.

7. *Design of an integrated agent with diverse capabilities*

   A long term goal is to add breadth and depth to our existing development of time-situated formalisms: to design and implement an integrated time-situated system which would be able to reason effectively across various problems and domains, and to combine planning and non-planning problem solving. Eventually, we would like to see "active logics" strengthened to investigate realistic problems that demand a treatment of time, space, and other forms of embeddedness, not merely as external entities to be reasoned about, but rather as features guiding their inferences.

10.2 Implementations

Portions of the active-logic planning and temporal reasoning framework have been implemented in Prolog. The implementation served two purposes in this research: it helped to confirm that the inference engine indeed performs the desired sequence of deliberation and execution, and secondly it gave us the opportunity to enhance the design in an incremental fashion. The implementation also brought to our attention the glaring need to address other problems such as space and computation limitations that we had set aside initially in the interest of providing a treatment of time and deadline issues. We recognized some inherent difficulties with the original step-logic framework by observing the behaviors of the inference engine as we incrementally

added more capabilities to it. Our work in space and computation resource limitations has addressed some of these concerns.

Appendix A

Examples of sample scenarios

This appendix contains examples of Dudley's reasoning during some variations of the Nell and Dudley scenario. The next section gives some sample axioms that encode the domain specific knowledge. The inference rules were described in Chapter 6.

A.1 Sample axioms

Relevant to moving:

- $Run(T_1 : \overline{T}_2, Y, L_1 : L_2) \rightarrow At(T_2, Y, L_2)$,$T_2 = T_1 + (L_2 - L_1)/v_Y$ [1]
- $condition(Run(T_1 : \overline{T}_2, Y, L_1 : L_2), At(T_1, Y, L_1))$
- $result(Run(T_1 : \overline{T}_2, Y, L_1 : L_2), At(T_2, Y, L_2))$

Relevant to untying and releasing:

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

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

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

Relevant to telephones and warning:

- $condition(Warn(S : \overline{T}, X, Y), In\_contact(S : T, X, Y))$
- $result(Repeat\_Until(S : \overline{T}, Dial, Get\_connection, X, Y), In\_contact(X, Y))$
- $condition((Dial, At(S : T, X, L) \wedge At(S : T, phone, L))$

A.2 Dudley, Nell and the rushing train

To give a flavor of the deadline-coupled reasoning, we first consider a very simple scenario and show some steps from Dudley's real-time reasoning.[2] Here Nell is a distance of 30 '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. 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 $\bullet\!\!\rightarrow$ as it appears in $X(S : T \bullet\!\!\rightarrow R, \ldots)$ denotes that X is intended to hold beyond $S : T$ up to R (by default). Its use in a result of an action

[2] For fuller details see [KNP90, NKP91].

indicates that the result must be preserved for use in a later segment of the plan. The number at the right bottom corner of a triplet denotes its place in the plan sequence.

Step 0:

$\mathbf{CS}(0, null, \{\ldots, At(0,d,h)_{obs}, Tied(0,n,r)_{obs}\})$,

$\mathbf{Proj}(0, null\{\})$,

$\mathbf{Goal}(save, Out\_of\_danger(50,n,r), 50)$,

$\mathbf{Unsolved}(0, Out\_of\_danger(50,n,r)), \ldots$

Step 1:

$\mathbf{CS}(1, null, \{\ldots, At(0,d,h)_{obs}, Tied(0,n,r)_{obs}\})$,

$\mathbf{Proj}(1, null, \{At(1:\infty,d,h), Tied(1:\infty,n,r)\})$,

$\mathbf{CS}(1, save, \{\ldots, At(0,d,h)_{obs}, Tied(0,n,r)_{obs}\})$,

$\mathbf{Ppl}(1, save, \left\{ \left[\begin{array}{c} \neg Tied(t_1,n,r) \\ Pull(t_1 : \overline{t}_2, d, n, r) \\ Out\_of\_danger(t_2 \bullet\!\!\rightarrow 50, n, r) \end{array} \right]_1 \right\})$,

$\mathbf{Proj}(1, save, \{\})$,

$\mathbf{WET}(1, save, 0)$,

$\mathbf{Feasible}(1, save), \ldots$

Step 2:

$\mathbf{CS}(2, save, \{\ldots, At(0,d,h)_{obs}, Tied(0,n,r)_{obs}, Pull(t_1 : \overline{t}_2, d, n, r), t_2 \leq 50, t_1 = t_2 - 1\})$,

$\mathbf{Ppl}(2, save, \left\{ \left[\begin{array}{c} At(t_3 : t_4, d, r) \\ Release(t_3 : \overline{t}_4, d, n, r) \\ \neg Tied(t_4 \bullet\!\!\rightarrow t_1, n, r) \end{array} \right]_1 \left[\begin{array}{c} \neg Tied(t_1,n,r) \\ Pull(t_1 : \overline{t}_2, d, n, r) \\ Out\_of\_danger(t_2 \bullet\!\!\rightarrow 50, n, r) \end{array} \right]_2 \right\})$,

$\mathbf{Proj}(2, save, \{At(1:\infty,d,h), Tied(1:\infty,n,r)\})$,

$\mathbf{WET}(2, save, 1)$,

$\mathbf{Feasible}(2, save), \ldots$

Step 3:

$\mathbf{CS}(3, save, \{\ldots, At(0,d,h)_{obs}, Tied(0,n,r)_{obs}, Pull(t_1 : \overline{t}_2, d, n, r), Out\_of\_danger_c(t_2,n,r),$

$Release(t_3 : \overline{t}_4, d, n, r), t_2 \leq 50, t_1 = t_2 - 1, t_3 = t_4 - 3, t_4 \leq t_1\})$,

$$\mathbf{Ppl}(3, save, \left\{ \left[\begin{array}{c} At(t_6, d, L) \\ Run(t_6 : \overline{t}_7, d, L : r) \\ At(t_7 \bullet\!\!\rightarrow t_3, d, r) \end{array} \right]_1 \left[\begin{array}{c} At(t_3 : t_4, d, r) \\ Release(t_3 : \overline{t}_4, d, n, r) \\ \neg Tied(t_4 \bullet\!\!\rightarrow t_1, n, r) \end{array} \right]_{2\ldots} \right\})$$

$\mathbf{Proj}(3, save, \{At(1 : \infty, d, h), Tied(1 : \infty, n, r)\})$,

$\mathbf{WET}(3, save, 4)$,

$\mathbf{Feasible}(3, save)$, ...

Step 4:

$\mathbf{CS}(4, save, \{\ldots, At(0, d, h)_{obs}, Tied(0, n, r)_{obs}, Pull(t_1 : \overline{t}_2, d, n, r), Out\_of\_danger_c(t_2, n, r),$

$Release(t_3 : \overline{t}_4, d, n, r), Run(t_6 : \overline{t}_7, d, L : r), \neg Tied_c(t_4, n, r),$

$t_2 \leq 50, t_1 = t_2 - 1, t_3 = t_4 - 3, t_4 \leq t_1, t_6 < t_7, t_7 \leq t_3\})$,

$$\mathbf{Ppl}(4, save, \left\{ \left[\begin{array}{c} At(t_6, d, h) \\ Run(t_6 : \overline{t}_7, d, h : r) \\ At(t_7 \bullet\!\!\rightarrow t_3, d, r) \end{array} \right]_1 \left[\begin{array}{c} At(t_3 : t_4, d, r) \\ Release(t_3 : \overline{t}_4, d, n, r) \\ \neg Tied(t_4 \bullet\!\!\rightarrow t_1, n, r) \end{array} \right]_{2\ldots} \right\},$$

$\mathbf{Proj}(4, save, \{At(1 : \infty, d, h), Tied(1 : \infty, n, r), Out\_of\_danger(t_2 + 1 : \infty, n, r), \})$,

$\mathbf{WET}(s4, save, 4)$,

$\mathbf{Feasible}(4, save)$, ...

Step 5:

$\mathbf{CS}(5, save, \{\ldots, At(0, d, h)_{obs}, At_c(t_7, d, r), Run(t_6 : \overline{t}_7, d, h : r), Tied(0, n, r)_{obs},$

$Pull(t_1 : \overline{t}_2, d, n, r), Out\_of\_danger_c(t_2, n, r), Release(t_3 : \overline{t}_4, d, n, r) \neg Tied_c(t_4, n, r),$

$t_2 \leq 50, t_1 = t_2 - 1, t_3 = t_4 - 3, t_4 \leq t_1, t_6 = t_7 - 30, t_7 \leq t_3)\})$,

$$\mathbf{Ppl}(5, save, \left\{ \left[\begin{array}{c} At(t_6, d, h) \\ Run(t_6 : \overline{t}_7, d, h : r) \\ At(t_7 \bullet\!\!\rightarrow t_3, d, r) \end{array} \right]_1 \left[\begin{array}{c} At(t_3 : t_4, d, r) \\ Release(t_3 : \overline{t}_4, d, n, r) \\ \neg Tied(t_4 \bullet\!\!\rightarrow t_1, n, r) \end{array} \right]_{2\ldots} \right\},$$

$\mathbf{Proj}(5, save, \{At(1 : \infty, d, h), Out\_of\_danger(t_2 + 1 : \infty, n, r),$

$Tied(1 : t_4 - 1, n, r), \neg Tied(t_4 + 1 : \infty, n, r), \})$,

$\mathbf{WET}(5, save, 34)$,

$\mathbf{Feasible}(5, save)$, ...

Step 6:

$\mathbf{CS}(6, save, \{\ldots, At(0,d,h)_{obs}, At_c(t_7,d,r), Run(t_6 : \overline{t}_7, d, h : r), Tied(0,n,r)_{obs},$

$Pull(t_1 : \overline{t}_2, d, n, r), Out\_of\_danger_c(t_2, n, r), Release(t_3 : \overline{t}_4, d, n, r) \neg Tied_c(t_4, n, r),$

$t_2 \leq 50, t_1 = t_2 - 1, t_3 = t_4 - 3, t_4 \leq t_1, t_6 = t_7 - 30, t_7 \leq t_3)\}),$

$$\mathbf{Ppl}(6, save, \left\{ \begin{array}{c} \left[\begin{array}{c} At(t_6,d,h) \\ Pace(t_6 : \overline{t_6+1}, d, h : h+1) \\ At(t_6+1,d,h+1) \end{array}\right]_1 \\ \left[\begin{array}{c} At(t_6+1,d,h+1) \\ Pace(t_6+1 : \overline{t_6+2}, d, h+1 : h+2) \\ At(t_6+2t_3,d,h+2) \end{array}\right]_2 \\ \ldots \\ \left[\begin{array}{c} At(t_6+29,d,h+29) \\ Pace(t_6+29 : \overline{t_6+30}, d, h+29 : r) \\ At(t_7 \bullet\!\!\rightarrow t_3,d,h+30) \end{array}\right]_{30_{\ldots}} \end{array} \right\},$$

$\mathbf{Proj}(6, save, \{At(1 : t_7 - 1, d, h), At(t_7 + 1 : \infty, d, r), Out\_of\_danger(t_2 + 1 : \infty, n, r),$

$Tied(1 : t_4 - 1, n, r), \neg Tied(t_4 + 1 : \infty, n, r), \}),$

$\mathbf{WET}(6, save, 34),$

$\mathbf{Feasible}(6, save), \quad \ldots$

Step 7:

$\mathbf{CS}(7, save, \{\ldots, At(0,d,h)_{obs}, At_c(t_7,d,r), Run(t_6 : \overline{t}_7, d, h :$

$r), Tied(0,n,r)_{obs}, \neg Tied_c(t_4,n,r),$

$Pull(t_1 : \overline{t}_2, d, n, r), Out\_of\_danger_c(t_2, n, r),$

$Pace(t_6 : t_6 + 1, d, h : h+1), \ldots Pace(t_6 + 29 : t_6 + 30, d, h : h + 30),$

$Release(t_3 : \overline{t}_4, d, n, r), t_2 \leq 50, t_1 = t_2 - 1,$

$t_3 = t_4 - 3, t_4 \leq t_1, t_6 = 8, t_6 = t_7 - 30, t_7 \leq t_3)\}),$

$$\mathbf{Ppl}(7, save, \left\{ \begin{array}{c} \left[\begin{array}{c} At(7,d,h) \\ Pace(7:\overline{8},d,h:h+1) \\ At(8,d,h+1) \end{array} \right]_{1\ldots} \\ \ldots \\ \left[\begin{array}{c} At(36,d,h+29) \\ Pace(36:\overline{37},d,h+29:r) \\ At(37 \bullet\!\!\rightarrow t_3,d,h+30) \end{array} \right]_{30\ldots} \end{array} \right\},$$

$\mathbf{Proj}(7, save, \{At(1:t_7-1,d,h), At(t_7+1:\infty,d,r), Out\_of\_danger(t_2+1:\infty,n,r),$

$Tied(1:t_4-1,n,r), \neg Tied(t_4+1:\infty,n,r),\}),$

$\mathbf{WET}(7, save, 34),$

$\mathbf{Feasible}(7, save), \quad \ldots$

Step 8:

$\mathbf{CS}(8, save, \{\ldots, At(0,d,h)_{obs}, At_c(8,d,h+1), \ldots, At_c(37,d,r),$

$Run(8:\overline{38},d,h:r), Tied(0,n,r)_{obs}, \neg Tied_c(t_4,n,r),$

$Pull(t_1:\overline{t}_2,d,n,r), Out\_of\_danger_c(t_2,n,r),$

$Pace(7:8,d,h:h+1), \ldots Pace(36:37,d,h:h+30), Release(t_3:\overline{t}_4,d,n,r),$

$t_2 \leq 50, t_1 = t_2 - 1, t_3 = t_4 - 3, t_4 \leq t_1, t_6 = 7, t_6 = t_7 - 30, t_7 \leq t_3)\}),$

$$\mathbf{Ppl}(8, save, \left\{ \begin{array}{c} \left[\begin{array}{c} At(8,d,h+1) \\ Pace(8:\overline{9},d,h+1:h+2) \\ At(9,d,h+2) \end{array} \right]_{1\ldots} \\ \ldots \\ \left[\begin{array}{c} At(36,d,h+29) \\ Pace(36:\overline{37},d,h+29:r) \\ At(37 \bullet\!\!\rightarrow t_3,d,h+30) \end{array} \right]_{29\ldots} \end{array} \right\},$$

$\mathbf{Proj}(8, save, \{At(1:36,d,h), At(38:\infty,d,r), Out\_of\_danger(t_2+1:\infty,n,r),$

$Tied(1:t_4-1,n,r), \neg Tied(t_4+1:\infty,n,r),\}),$

$\mathbf{WET}(8, save, 34),$

$\mathbf{Feasible}(8, save), \quad \ldots$

Dudley's planning and acting continues, he refines the $Release(t_3:t_3+3,d,n,r)$ into its

primitive (known) components. The following gives an idea of the rest of his time-situated reasoning until he has saved Nell:

Step number	**First action in Ppl**	**WET**
$Step9:$	$Pace(9:10, d, h+2:h+3)$	$WET(9, save, 33)$
$\vdots$	$\vdots$	$\vdots$
$Step36:$	$Pace(36:37, d, h+29:r)$	$WET(36, save, 6)$
$Step37:$	$Release_1(t_3:t_3+1, d, n, r)$	$WET(37, save, 5)$
$Step38:$	$Release_1(38:39, d, n, r)$	$WET(38, save, 4)$
$Step39:$	$Release_2(39:40, d, n, r)$	$WET(39, save, 4)$
$Step40:$	$Release_3(40:41, d, n, r)$	$WET(40, save, 3)$
$Step41:$	$Pull(t_1:t_1+1, d, n, r)$	$WET(41, save, 2)$
$Step42:$	$Pull(42:43, d, n, r)$	$WET(42, save, 1)$
$Step43:$	$Null$	$WET(43, save, 1)$

A.3 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 plans 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 $Cut\_ropes(S : \overline{F}, \ldots)$, namely, $At(S : F, d, r) \wedge Have(S : F, d, knife)$. The inference rule 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, fires. 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. It 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 inferencing, 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.

3: $\mathbf{CS}(3, save, \{At(0, d, h)_{obs}, At(0, knife, h), Tied(0, n, r)_{obs}\}),$

$$\mathbf{Ppl}(save, 3, \left\{ \left[\begin{array}{c} At(t_3 : t_4, d, r) \wedge Have(t_3 : t_4, d, knife) \\ Cut\_ropes(t_3 : \overline{t}_4, d, n, r) \\ \neg Tied(t_4 \bullet\!\!\rightarrow t_1, n, r) \end{array} \right]_{1_{\ldots}} \ldots \right\}),$$

$\mathbf{CS}(3, save, \{\ldots, t_3 = t_4 - 3, \ldots\})$

$\mathbf{Proj}(3, save, \{At(1 : \infty, d, h), At(1 : \infty, knife, h), Tied(1 : t_4 - 1, n, r), \ldots\}) \ldots$

$\vdots$

5: $\mathbf{Ppl}(5, save1, \left\{ \begin{array}{c} \left[\begin{array}{c} At(t_6, d, L_1) \wedge At(t_6, knife, L_1) \\ Pick\_up(t_6 : \overline{t}_7, d, knife) \\ Have(t_7 \bullet\!\!\rightarrow t_4, d, knife) \end{array} \right]_1 \\ \left[\begin{array}{c} At(t_8, d, L_2) \\ Run(t_8 : \overline{t}_9, d, L_2 : r) \\ At(t_9 \bullet\!\!\rightarrow t_4, d, r) \end{array} \right]_{2\ldots} \end{array} \right\}),$

$\mathbf{CS}(5, save1, \{\ldots, t_6 = t_7 - 1, \ldots\})$

$\mathbf{Ppl}(5, save2, \left\{ \begin{array}{c} \left[\begin{array}{c} At(t_{10}, d, L_4) \\ Run(t_{10} : \overline{t}_{11}, d, L_4 : r) \\ At(t_{11} \bullet\!\!\rightarrow t_4, d, r) \end{array} \right]_1 \\ \left[\begin{array}{c} At(t_{12}, d, L_3) \wedge At(t_6, knife, L_3) \\ Pick\_up(t_{12} : \overline{t}_{13}, d, knife) \\ Have(t_{13} \bullet\!\!\rightarrow t_4, d, knife) \end{array} \right]_{2\ldots} \end{array} \right\}),$

$\mathbf{CS}(5, save2, \{\ldots, t_{12} = t_{13} - 1, \ldots\})$

$\mathbf{Proj}(5, save1, \{At(1 : \infty, d, h), At(1 : \infty, knife, h), Tied(1 : t_4 - 1, n, r), \ \ldots\})$

$\mathbf{Proj}(5, save2, \{At(1 : \infty, d, h), At(1 : \infty, knife, h), Tied(1 : t_4 - 1, n, r), \ \ldots\}) \ldots$

6: $\mathbf{Ppl}(6, save1, \left\{ \begin{array}{c} \left[\begin{array}{c} At(t_6, d, h) \wedge At(t_6, knife, h) \\ Pick\_up(t_6 : \overline{t}_7, d, knife) \\ Have(t_7 \bullet\!\!\rightarrow t_4, d, knife) \end{array} \right]_1 \\ \left[\begin{array}{c} At(t_8, d, h) \\ Run(t_8 : \overline{t}_9, d, h : r) \\ At(t_9 \bullet\!\!\rightarrow t_4, d, r) \end{array} \right]_{2\ldots} \end{array} \right\}),$

$\mathbf{CS}(6, save1, \{\ldots, t_6 = t_7 - 1, \ldots\})$

$\mathbf{Ppl}(6, save2, \left\{ \begin{array}{c} \left[\begin{array}{c} At(t_{10}, d, h) \\ Run(t_{10} : \overline{t}_{11}, d, h : r) \\ At(t_{11} \bullet\!\!\rightarrow t_4, d, r) \end{array} \right]_1 \\ \left[\begin{array}{c} At(t_{12}, d, h) \wedge At(t_6, knife, h) \\ Pick\_up(t_{12} : \overline{t}_{13}, d, knife) \\ Have(t_{13} \bullet\!\!\rightarrow t_4, d, knife) \end{array} \right]_{2\ldots} \end{array} \right\}),$

$\mathbf{CS}(6, save2, \{\ldots, t_{12} = t_{13} - 1, \ldots\}) \ldots$

7: $\mathbf{Freeze}(7, save2), \ \ldots$

A.4 Another alternative: stop the train!

The examples above was kept as simple as possible to elucidate the workings of the active-logic planner. Now suppose we enhance Dudley's set of axioms so that he knows about stopping trains, warning drivers and making telephone calls. Then, as he synthesizes the above obvious plan to 'run to Nell and untie her', he can simultaneously plan for another alternative – he could get the driver to stop the train in time! But how does he establish contact with the driver? One way is to go to the nearest telephone and call the train station. Dudley knows that it is is 50 time steps until the deadline. Where is the nearest telephone? His neighbor has one, and the neighbor lives only 5 paces away. How long will it take Dudley until he can get the connection? His previous experience with telephones tells him he must allow a good 5 steps, he possibly may have to redial several times (perform a *Repeat_until* type of action[3]). How long will it take him to warn the train driver? Perhaps 5 steps seems adequate. Dudley plans for this 'stop the train' alternative in parallel with the 'run to Nell and untie her' plan. Below, we illustrate some of the steps in the formation of this plan, the step numbers correspond to the matching numbers in the earlier plan. The two should be visualized to be generated in parallel, although they are shown separately here for the sake of simplicity and readability.

This plan involves a dimension that we have not alluded to before; it involves the action of another agent. Unlike in the earlier plan, where all actions were under Dudley's control, this plan depends on an action *Stop_train* which has to be performed by an agent other than Dudley, in this case, the train driver. How can Dudley plan for this? Dudley has the following two axioms:

[3]As a planner, Dudley must reason about various types of actions vis-a-vis his deadline. These include conditional actions, and repeat-until type of actions. A repeat-until type of action is characterized by the appearance of a signaling condition to mark its end. Appropriate rules in Appendix B show Dudley's inference rules to this effect.

$Stop\_train(T : T+2, driver) \rightarrow Out\_of\_danger(T+2, nell, railtrack)$

$Warn(S : T, dudley, driver) \rightarrow Stop\_train(T : T+2, driver)$

The second axiom hints at an unknown in the plan: Can Dudley trust the driver to stop the train? What if the villain is driving the train himself? Suppose that Dudley does believe, that by warning the driver he can get him to stop the train, then his plan must include the action $Warn$, and his total time estimate must allow for the time taken by the driver in performing the stop. He can not attempt to satisfy the conditions for $Stop\_train$ since they are not within his control, but he must proceed with his bit of the plan i.e. with warning the driver. The following steps illustrate the simultaneous formulation of this plan. The abbreviation dr denotes the train driver.

0: $\mathbf{CS}(0, null, \{At(0,d,h)_{obs}, Tied(0,n,r)_{obs}\})$, $\mathbf{Goal}(stop, Out\_of\_danger(50,n,r), 50), \ldots$

1: $\mathbf{CS}(1, null, \{At(0,d,h)_{obs}, Tied(0,n,r)_{obs}\})$, $\mathbf{Goal}(stop, Out\_of\_danger(50,n,r), 50),$

$\ldots \mathbf{Ppl}(1, stop, \left\{ \left[\begin{array}{c} \ldots \\ Stop\_train(\tau_1 : \tau_2, dr) \\ Out\_of\_danger(\tau_2 \bullet\!\!\rightarrow 50, n, r) \end{array} \right]_1 \right\}),$

$\{\tau_2 \leq 50, \tau_1 = \tau_2 - 2\},$

$\mathbf{WET}(0, stop, 0), estimate(Stop\_train(\tau_1 : \tau_2, dr), 2), \mathbf{Feasible}(1, \text{stop}), \ldots$

2: $\mathbf{Ppl}(2, stop, \left\{ \begin{array}{l} \left[\begin{array}{c} In\_contact(\tau_3 : \tau_4, d, dr) \\ Warn(\tau_3 : \tau_4, d, dr) \\ Knows\_about(\tau_4 \bullet\!\!\rightarrow \tau_1, n, dr) \end{array} \right]_1 \\ \left[\begin{array}{c} \ldots \\ Stop\_train(\tau_1 : \tau_2, dr) \\ Out\_of\_danger(\tau_2 \bullet\!\!\rightarrow 50, n, r) \end{array} \right]_2 \end{array} \right\}),$

$\{\tau_2 \leq 50, \tau_1 = \tau_2 - 1, \tau_3 = \tau_4 - 3, \tau_4 \leq \tau_1\}$

$\mathbf{WET}(2, stop, 2), estimate(Warn(\tau_3 : \overline{\tau}_4, d, dr), 5), \mathbf{Feasible}(2, \text{stop}), \ldots$

$$\textbf{3: Ppl}(stop, 4, \left\{ \begin{array}{c} \left[\begin{array}{c} At(\tau_6 : \tau_7, phone, L_1) \wedge At(\tau_6 : \tau_7, d, L_1) \\ Repeat\_until(\tau_6 : \tau_7, Dial, Get\_connection, d, dr) \\ In\_contact(\tau_7 \bullet\!\!\rightarrow \tau_4, d, dr) \end{array} \right]_1 \\ \left[\begin{array}{c} In\_contact(\tau_3 : \overline{\tau}_4, d, dr) \\ Warn(\tau_3 : \overline{\tau}_4, d, dr) \\ Knows\_about(\tau_4 \bullet\!\!\rightarrow \tau_1, n, dr) \end{array} \right]_{2\ldots} \end{array} \right\},$$
$$\{\tau_2 \leq 50, \tau_1 = \tau_2 - 1,\ \tau_3 = \tau_4 - 3, \tau_4 \leq \tau_1\ \ \tau_6 < \tau_7, \tau_7 \leq \tau_3\}\ \textbf{WET}(3, stop, 7),$$
$$estimate(Repeat\_until(\tau_6 : \tau_7, Dial, Get\_connection, d, dr), 5), \textbf{Feasible}(3, stop),\ \ \ldots$$
$$\vdots$$

In Section 8.6, we discussed heuristics for choosing between possible (partial) plans, while taking the time for *choosing* also into account within the same framework.

Appendix B

Soundness and completeness of the active modal logic

In this appendix we give a detailed proof of Theorem 9.1. We restate the theorem here:

Theorem 9.1: The set of axioms (A0–A3) and inference rules (R1–R3) in Figure 9.3 provide a sound and complete axiomatization of the modal active-logic for reasoning in time.

B.1 Proof of Theorem 9.1

B.1.1 Soundness

To show soundness is to show that all the axioms are sound, and that the inference rules preserve the soundness.

The axioms hold as a result of the restrictions (constraints C0 thru C3) that we have imposed in the semantics on $\mathcal{M}$, the class of structures that satisfy these constraints. Let $M \in \mathcal{M}$ be a structure in this class. We first show that each axiom is sound.

- **A0:** Truth of $B_0 true$ follows directly from restriction **C0** on the models.
- **A1:** If ϕ is an instance of a propositional tautology then $M, l \models \phi$. This fol-

lows from the interpretation of $\wedge$ and $\neg$ in our semantics being the same as in propositional logic.

- **A2:** Restriction **C1** states that $\forall l, \forall t, \{\} \notin \mathcal{B}(l,t)$. From this, it directly follows that $\neg B_t false$ is true in every model.

- **A3:** This holds as a result of the restriction **C3** on the intersections of sets that we impose on our structure between consecutive time points.

Next, we show that R1 thru R3 preserve soundness.

- **R1:** Soundness of **R1** follows similar to that of propositional logic.

- **R2:** To show that **R2** is sound, we must show that if $\models \phi \rightarrow \psi$ and $M, l \models B_t\phi$ then $M, l \models B_t\psi$. If $M, l \models B_t\phi$ then $||\phi|| \in \mathcal{B}_t(l)$. Since $\models \phi \rightarrow \psi$, $||\phi \rightarrow \psi|| = L$. Since $\models \phi \rightarrow \psi$, $||\phi \rightarrow \psi|| = L$, $||\psi||$ is a superset of $||\phi||$. By restriction **C2** on models, $||\psi|| \in \mathcal{B}_t(l)$. Thus, we have $M, l \models B_t\psi$.

- **R3:** **R3** is sound since from constraint **C0**, $||true|| = L$ is in $\mathcal{B}_0(l)$ for all time lines l, and $\vdash \phi$ means that $||\phi|| = ||true||$.

Thus the axioms are sound, and that the inference rules preserve the soundness. □

B.1.2 Completeness

To prove that the modal active logic is complete, we first prove the following lemma:

Lemma B.1 There exist maximal consistent sets satisfying the following properties for each max. consistent set F:

(1) for every formula ϕ either $\phi \in F$ or $\neg\phi \in F$;

(2) $\phi \wedge \psi \in F$ iff $\phi \in F$ and $\psi \in F$;

(3) If $\phi \in F$ and $\phi \rightarrow \psi \in F$ then $\psi \in F$; and

(4) If ϕ is provable then $\phi \in F$.

Furthermore, every consistent formula is contained in some maximal consistent set.

Proof (of Lemma B.1)

The first three are straightforward to prove, for proof refer to [HY92]. For part (4), we need to show that every ϕ which is provable is in F. Suppose this is not the case. From property (1), $\neg\phi \in F$. Thus every model of F is a model of $\neg\phi$. But ϕ is provable, and hence by the soundness part that we just showed, it is valid. so, every model is a model of ϕ, hence a contradiction. This proves the lemma. □

Lemma B.2 Every maximal consistent set has a (canonical) model.

Remark B.3 Completeness follows from the above lemma. To show completeness means to show that every valid formula is provable. It is sufficient to show Lemma B.2. Suppose ϕ is valid. Suppose Lemma B.2 holds, and that ϕ is not provable. Then, neither is $\neg\neg\phi$, and thus $\neg\phi$ is consistent. Thus $\neg\phi$ is contained in some maximal consistent set. But every consistent set has a model by Lemma B.2. Thus $\neg\phi$ has a model. This model must model ϕ, a contradiction. Thus ϕ must be provable. Thus if Lemma B.2 holds then if ϕ is valid, then ϕ is provable. This is the completeness theorem.

Proof(of Lemma B.2):

We construct a canonical structure $M^c = \langle L^c, \mathcal{N}, v, <, \pi, \mathcal{B}\rangle$ as follows:

This canonical structure contains a set of time lines L^c that has a time line l_V for every maximal consistent set V.

- $L^c = \{l_V | \text{V is a maximal consistent set}\}$.
- l_V is the structure

  $\langle\{\{\phi | \phi \text{ is of the form } p(i) \text{ or } B_i\psi \text{ , where } p(i) \in PT, \psi \in \mathcal{G}, \phi \in V\} : i \in \mathcal{N}\}, <\rangle$.

- $v(n \in \mathcal{N}) = n$.
- $\pi(p(\tau), l_V) = true$ iff $p(\tau) \in V$.
- For $t \in \mathcal{N}$, $\mathcal{B}_t(l_V) =$

  $\{\ I_\psi \mid B_t\psi \in V\ \} \cup \{\ S \mid S \supseteq S', S' \in \{\ I_\psi \mid B_t\psi \in V\ \}$ and $\forall\psi, \psi \in \mathcal{G} \rightarrow S \neq I_\psi\}$.

To prove B.2 is to show that every maximal consistent set V has a model, which amounts to showing that

$M^c, l_V \models \phi$ **iff** $\phi \in V$.

We prove this by induction on the length of formula ϕ.

Base case: Let $\phi = p(t)$. By definition of the canonical model, $p(t) \in V$ iff $\pi(p(t), l_V) = true$. Thus, by the truth definition, $M^c, l_V \models p(t)$.

Inductive Hypothesis (I.H) For formulas ψ shorter than ϕ,

$M^c, l_V \models \psi$ iff $\psi \in V$.

INDUCTIVE STEP:

Negation: Let $\phi = \neg\psi$.

$\neg\psi \in V$

$\Updownarrow$ (V is max. consis.)

$\psi \notin V$

$\Updownarrow$ (I. H.)

$M^c, l_V \not\models \psi$

$\Updownarrow$ (truth defn)

$M^c, l_V \models \neg\psi$.

Similarly we can show the result for other forms of ϕ involving $\wedge$ etc. The interesting case is the following.

Belief: Let $\phi = B_t\psi$.

$B_t\psi \in V$

$\Updownarrow$ (M^c construction)

$\{l_W : \psi \in W\} \in \mathcal{B}(l_V, t)$

$\Updownarrow$ (I. H.)

$\{l_W : M^c, l_W \models \psi\} \in \mathcal{B}(l_V, t)$

$\Updownarrow$ (defn of $||\psi||$)

$||\psi|| \in \mathcal{B}(l_V, t)$

$\Updownarrow$ (truth defn)

$M^c, l_V \models B_t\psi$.

And thus we have proven Lemma B.2 which in turn proves the completeness theorem. □

We will find it useful to state the following corollary that follows directly from the above result

Corollary B.4 $||\phi|| = \{l_W : M^c, l_W \models \phi\} = \{l_W : \phi \in W\}$. □

We also prove the following simple Lemma:

Lemma B.5 For every set $S \in \mathcal{B}_t(l_V)$ in the canonical model, there is some set $S' \in \mathcal{B}_t(l_V)$ such that $S \supseteq S'$ and $S' = ||\phi||$ for some formula $\phi \in \mathcal{G}$.

Proof (of Lemma B.5): By the construction of the canonical model, if $S \in \mathcal{B}_t(l_V)$, then either S is the intension of some formula or it is the superset of some set which itself is an intension. □

B.2 The canonical structure satisfies C0 – C3

We need to show that the canonical structure M^c defined above indeed satisfies all the constraints **C0 – C3** that we have imposed on the models.

- **C0:** To show that: $||true|| \in \mathcal{B}(l_V, 0)$.

  From axiom A0, $\vdash B_0 true$. Since V is a maximal consistent set, $B_0 true \in V$. From the definition of the canonical model it follows that $||true|| \in \mathcal{B}(l_V, 0)$.

- **C1:** To show that: $\forall t, \forall V, \{\} \notin \mathcal{B}(l_V, t)$ in the structure M^c.

  Suppose, for some V and t, $\{\} \in \mathcal{B}(l_V, t)$. $\{\} = \{l_W | false \in W\}$. By the definition of the structure M^c, This means that $B_t false \in V$. By axiom A2, $\vdash \neg B_t false$, therefore $\neg B_t false \in V$, since V is a maximal consistent set. This is a contradiction since V is a maximal consistent set.

  Therefore, $\forall t, \forall V, \{\} \notin \mathcal{B}(l_V, t)$.

- **C2:** To show that: If $s_1 \in \mathcal{B}(l_V, t)$ and $s_2 \supseteq s_1$, then $s_2 \in \mathcal{B}(l_V, t)$.

  Let $s_1 \in \mathcal{B}(l_V, t)$ and $s_2 \supseteq s_1$. There are two cases:

  Case (i)

  Set s_2 is not the intension of any formula in $\mathcal{G}$. Whether or not s_1 is an intension, by Lemma B.5, $s_1 \supseteq s_1'$, where s_1' is an intension, and $s_1' \in \mathcal{B}_t(l_V)$. Thus, $s_2 \supseteq s_1'$, and by construction of the canonical model, since $s_1' \in \mathcal{B}_t(l_V)$, $s_2 \in \mathcal{B}_t(l_V)$.

  Case (ii)

  Set s_2 is itself an intension of some formula ψ. i.e. $s_2 = ||\psi||$. By Lemma B.5, $s_1 \supseteq s_1'$, where s_1' is an intension of some formula ϕ, and $s_1' = ||\phi|| \in \mathcal{B}_t(l_V)$. Thus, By Corollory B.4, $\{l_W : \phi \in W\} \in \mathcal{B}_t(l_V)$, and by the canonical model construction, $B_t\phi \in V$.

  Since $s_2 \supseteq s_1'$, $||\psi|| \supseteq ||\phi||$. We claim that $\vdash \phi \rightarrow \psi$. Suppose $\nvdash \phi \rightarrow \psi$. Then $\phi \wedge \neg\psi$ is consistent, and there is some maximal consistent set W such that $\phi \in W$, and $\psi \notin W$. Thus $l_W \in ||\phi||$, and $l_W \notin ||\psi||$. But, this contradicts $||\psi|| \supseteq ||\phi||$.

Thus, $\vdash \phi \rightarrow \psi$. Thus, by **R2**, $\vdash B_t\phi \rightarrow B_t\psi$. Thus $B_t\phi \rightarrow B_t\psi \in V$. But $B_t\phi \in V$, and V is a maximal consistent set. Therefore, $B_t\psi \in V$. It then follows by Corollary B.4 and the the canonical model construction that $s_2 = ||\psi|| \in \mathcal{B}_t(l_V)$.

- **C3:** To show that: If $s_1, s_2 \in \mathcal{B}(l_V, t)$, then $s_1 \cap s_2 \in \mathcal{B}(l_V, t+1)$.

  By Lemma B.5, there exist sets $s_1{}'$ and $s_2{}'$ in $\mathcal{B}_t(l_V)$, such that $s_1 \supseteq s_1{}'$, and $s_2 \supseteq s_2{}'$, and such that both $s_1{}'$ and $s_2{}'$ are intensions of some formulas. Then $s_1{}' = ||\phi||$ and $s_2{}' = ||\psi||$ for some $\phi, \psi \in \mathcal{G}$. By Corollory B.4, and the canonical model construction,

  $B_t\phi \in V$ and $B_t\psi \in V$.

  By Axiom **A3**, $B_{t+1}(\phi \wedge \psi) \in V$. Once again, by Corollary B.4,

  $||\phi \wedge \psi|| \in \mathcal{B}_{t+1}(l_V)$.

  But $||\phi \wedge \psi|| = s_1{}' \cap s_2{}'$. Thus $s_1{}' \cap s_2{}' \in \mathcal{B}_{t+1}(l_V)$. Now, $s_1 \cap s_2 \supseteq s_1{}' \cap s_2{}'$, and we have just shown that the canonical model satisfies restriction C2. Thus, $s_1 \cap s_2 \in \mathcal{B}_{t+1}(l_V)$.

This shows that the canonical model indeed satisfies all the constraints that we have imposed on our models.

Bibliography

[AC87] P. E. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceeding, AAAI-87*, pages 268–272, 1987.

[AHT90] J. Allen, J. Hendler, and A. Tate. *Readings in Planning*. Morgan Kaufmann Publishers, Inc., 1990.

[All84] J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[Ams91] J. Amsterdam. Temporal reasoning and narrative conventions. In *Proc. of KR-91*, pages 15–21, 1991.

[Ash88] N. Asher. Reasoning about belief and knowledge with self-reference and time. In *Proceedings of the second conferennce on Theoretical Aspects of Reasoning About Knowledge*, pages 61–81. Morgan Kaufmann, 1988.

[Bak89] Andrew B. Baker. A simple solution to the Yale Shooting Problem. In *Proceedings, KR89*, pages 11–20, 1989.

[BD94] M. Boddy and T. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 1994. To appear.

[BH91] S. Shekhar B. Hamidzadeh. Dynora: A real-time planning algorithm to meet response-time constraints in dynamic environments. In *Proceedings, IEEE international Conf. on Tools for AI*, 1991.

[Bro86] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[Bro91] R. Brooks. Intelligence without reason. In *Proceedings of IJCAI-91, Prepared for Computers and Thought*, 1991.

[Cha87] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–378, 1987.

[Che80] B. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, 1980.

[CL90] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[Dav88] D. E. Davis. Inferring ignorance from the locality of visual perception. In *Proceedings, AAAI-88*, St. Paul, Minnesota, 1988.

[DB88] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings, AAAI-88*, pages 49–54, St. Paul, Minnesota, 1988.

[DF89] B. D'Ambrosio and M. Fehling. Resource bounded-agents in an uncertain world. In *Proceeding, AAAI symposium on limited rationality*, Stanford, California, 1989.

[DFM88] Thomas Dean, R. James Firby, and David Miller. Hierarchical planning involving deadlines, travel time and resources. *Computational Intelligence*, 4:381–389, 1988.

[DM87] T. Dean and D. McDermott. Temporal data base management. *Artificial Intelligence*, 32(1):1–55, 1987.

[Doy82] J. Doyle. Some theories of reasoned assumptions: An essay in rational psychology. Technical report, Department of Computer Science, Carnegie Mellon University, 1982.

[Doy88] J. Doyle. Artificial intelligence and rational self-government. In *TR CS-88-124, CMU, Pittsburgh*, 1988.

[Ebe74] R. A. Eberle. A logic of believing, knowing and inferring. *Synthese*, 26:356–382, 1974.

[ED88a] J. Elgot-Drapkin. *Step-logic: Reasoning Situated in Time*. PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 1988.

[ED88b] J. Elgot-Drapkin. *Step-logic: Reasoning Situated in Time*. PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 1988.

[ED91] J. Elgot-Drapkin. Step-logic and the three wise men problem. In *Proceeding, AAAI-91*, Anaheim, California, 1991.

[EDMP87] J. Elgot-Drapkin, M. Miller, and D. Perlis. Life on a desert island: Ongoing work on real-time reasoning. In F. M. Brown, editor, *Proceedings of the 1987 Workshop on The Frame Problem*, pages 349–357. Morgan Kaufmann, 1987. Lawrence, Kansas.

[EDP90] J. Elgot-Drapkin and D. Perlis. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(1):75–98, 1990.

[Etz91] Oren Etzioni. Embedding decision-analytic control in a learning architecture. *Artificial Intelligence*, 49:129–159, 1991.

[FH88] R. Fagin and J. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, 1988.

[FN71] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Gel88] Michael Gelfond. Autoepistemic logic and formalization of common-sense reasoning. In *Proceedings of the second international workshop on nonmonotonic reasoning*, 1988. Munich 88.

[Geo87] Michael P. Georgeff. Many agents are better than one. In F. M. Brown, editor, *Proceedings of the 1987 Workshop on The Frame Problem*, pages 59–75x. Morgan Kaufmann, 1987. Lawrence, Kansas.

[Ger90] M.T. Gervasio. Learning general completable reactive plans. In *Proceeding, AAAI-90*, pages 1016–1021, 1990.

[GL88] M. Georgeff and A. Lansky. Reactive reasoning and planning. In *Proceedings AAAI-88*, pages 677–682, 1988.

[GL94] A. Garvey and V. Lesser. A survey of research in deliberative real-time ai. *Journal of Real Time Sytems*, 6:317–347, 1994.

[GS87] M. L. Ginsberg and D. E. Smith. Reasoning about action I: A possible worlds approach. In F. M. Brown, editor, *Proceedings of the 1987 Workshop on The Frame Problem*, pages 233–258, Lawrence, KS, 1987. Morgan Kaufmann.

[Ham89] K. Hammond. Case-based planning: Viewing planning as a memory task. *Perspectives in AI*, 1, 1989.

[Hau87] Brian A. Haugh. Simple causal minimizations for temporal persistence and projection. In *Proceedings of the sixth national conference on artificial intelligence*, pages 218–223, 1987.

[HB90] Eric J. Horvitz and John S. Breese. Ideal partition of resources for metareasoning. Technical Report KSL-90-26, Knowledge Systems Laboratory, Stanford University, March 1990.

[HHC90] Adele E. Howe, David M. Hart, and Paul R. Cohen. Addressing real-time constraints in the design of autonomous agents. *Journal of Real-Time Systems*, 2(12):81–97, 1990.

[Hin62] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

[HM87] S. Hanks and D. McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[Hor88] E. J. Horvitz. Reasoning under varying and uncertain resource constraints. In *Proceeding, AAAI-88*, pages 111–116, St. Paul, Minnesota, 1988.

[Hor89] Eric J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In T.S. Levitt L.N. Kanal and J.F. Lemmer, editors, *Uncertainity in Artificial Intelligence 3*. Elevesier Science Publishers, 1989.

[HR79] B. Hayes-Roth. A cognitive model of planning. *Cognitive Science*, pages 275–310, 1979.

[HR90] B. Hayes-Roth. Architectural foundations for real-time eperformance in intelligent agents. *Journal of Real Time Sytems*, 1990.

[HR91] Eric J. Horvitz and Geoffrey Rutledge. Time-dependent utility and action under uncertainity. In *Uncertainity in Artificial Intelligence 6*, 1991.

[HRWA+92] B. Hayes-Roth, R. Washington, D. Ash, A. Collinot, A. Vina, and A. Sevier. Guardian: A prototype intensive-care monitoring agent. *Artificial Intelligence in Medicine*, 4:165–185, 1992.

[HY92] J. Halpern and Moses Y. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[IG90] F. Ingrand and M. Georgeff. Managing deliberation and reasoning in real-time ai systems. In *Proceedings of 1990 DARPA workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 284–291, San Diego, CA, November 1990.

[Kae87] L. Kaelbling. An architecture for intelligent reactive systems. In M. Georgeff and A. Lansky, editors, *Reasoning about Actions and Plans*. Morgan-Kaufmann, 1987.

[Kae88] L. Kaelbling. Goals as parallel program specifications. In *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 60–65, St. Paul, Minnesota, 1988.

[Kau86] H. Kautz. The logic of persistence. In *Proceedings, AAAI-86*, pages 401–405, 1986.

[KL88] S. Kraus and D. Lehmann. Knowledge, belief and time. *Theoretical Computer Science*, 58:155–174, 1988.

[KNP90] S. Kraus, M. Nirkhe, and D. Perlis. 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, 1990.

[Kon86a] K. Konolige. *A Deduction Model of Belief*. Pitman, London, 1986.

[Kon86b] K. Konolige. What awareness isn't. In J. Halpern, editor, *Proceedings of the first conferennce on Theoretical Aspects of Reasoning About Knowledge*, Monterey, CA, 1986. Morgan Kaufmann.

[Kor90] R. E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42:189–211, 1990.

[KP89] S. Kraus and D. Perlis. Assessing others' knowledge and ignorance. In *Proc. of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 220–225, Charlotte, North Carolina, 1989.

[KR92] S. Kraus and J. Rosenschein. The role of representation in interaction: Discovering focal points among alternative solutions. In *Decentralized Artificial Intelligence, Volume 3*, Germany, 1992. Elsevier Science Publishers.

[Lak86] G. Lakemeyer. Steps towards a first order logic of explicit and implicit belief. In J. Halpern, editor, *Proceedings of the first conferennce on Theoretical Aspects of Reasoning About Knowledge*, Monterey, CA, 1986. Morgan Kaufmann.

[Leh84] D. J. Lehmann. Knowledge, common knowledge and related puzzles. In *Proceedings of the third annual ACM conference on principles of distributed computing*, pages 62–67, 1984.

[Lev84] H. Levesque. A logic of implicit and explicit belief. In *Proceedings of the National Conference on Artificial Intelligence*, pages 198–202, Austin, TX, 1984. AAAI.

[Lif87a] V. Lifschitz. Pointwise circumscription. In *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann, 1987.

[Lif87b] Vladimir Lifschitz. Formal theories of action. In *The Frame Problem in Artificial Intelligence*, pages 35–57, Los Altos, CA, 1987. Morgan-Kaufmann.

[LNR87] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[LPD88] Victor Lesser, Jasmina Pavlin, and Edmund Durfee. Approximate processing in real-time problem solving. *AI Magazine*, 9(1):49–61, '988.

[LR86] R. Ladner and J. H. Reif. The logic of distributed protocols. In J. Halpern, editor, *Proceedings of the first conferennce on Theoretical Aspects of Reasoning About Knowledge*, Monterey, CA, 1986. Morgan Kaufmann.

[McD78] D. McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[McD82] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6, 1982.

[McD87] D. McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[MH69] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Ma-*

chine Intelligence, volume 4, pages 463–502. Edinburgh University Press, 1969.

[MH79] R. C. Moore and G. Hendrix. Computational models of beliefs and the semantics of belief sentences. Technical Report Technical note 187, SRI International, Menlo Park, 1979.

[Mil56] G. Miller. The magical number seven plus or minus two. *The Psychological Review*, 63:81–97, 1956.

[Mil92] Michael J. Miller. *A view of one's past and other aspects of reasoned change in belief.* PhD thesis, Department of Computer Science, University of Maryland, Collge Park, MD, U.S.A, 1992. Ph.D. dissertation.

[Mon68] R. Montague. Pragmatics. *Contemporary Philosophy*, pages 101–121, 1968. La Nuova Italia Editrice.

[Mon70] R. Montague. Universal grammar. *Theoria*, 36:373–398, 1970.

[Mor88] P. H. Morris. The anomalous extension problem in default reasoning. *Artificial Intelligence*, 35:383–399, 1988.

[MS87] E. McKenzie and R. Snodgrass. Extending the relational algebra to support transaction time. In *Proceedings of the SIGMOD conference*. ACM, 1987.

[MS88] L. Morgenstern and L. A. Stein. Why things go wrong: A formal theory of causal reasoning. In *Proceeding, AAAI-88*, pages 518–523, 1988.

[Nil83] N. Nilsson. Artificial intelligence prepares for 2001. *AI Magazine*, 4(4):7–14, 1983.

[NKP91] M. Nirkhe, S. Kraus, and D. Perlis. Fully deadline-coupled planning: One step at a time. In *Proceedings of the sixth international symposium on methodologies for intelligent systems*, Charlotte, NC, 1991.

[NKP93] M. Nirkhe, S. Kraus, and D. Perlis. Situated reasoning within tight deadlines and realistic space and computation bounds. In *Proceedings of the Second Symposium On Logical Formalizations Of Commonsense Reasoning*, Austin, TX, 1993.

[Pea88] Judea Pearl. On logic and probability. *Computational Intelligence*, 4:99–103, 1988.

[PEDM] D. Perlis, J. Elgot-Drapkin, and M. Miller. Stop the world! – I want to think! Invited paper, *International J. of Intelligent Systems*, special issue on Temporal Reasoning, K. Ford and F. Anger (eds.), vol. 6, 1991, pp. 443–456.

[PR90] M. E. Pollack and M. Ringuette. Introducing the tileworld: Experimentally evaluating agent architectures. In *Proceedings, AAAI-90*, pages 183–189, 1990.

[PS85] P.F. Patel-Schneider. A decidable first order logic for knowledge representation. In *Proceedings of the 9th Int'l Joint Conference on Artificial Intelligence*, pages 455–458, Los Angeles, CA, 1985.

[RK86] S.J. Rosenschein and L.P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In *Proceedings of Workshop on Theoretical Aspects of Knowledge*, Monterey, CA, 1986.

[RK89] S.J. Rosenschein and L.P. Kaelbling. Integrating planning and reactive control. In *Proceedings of NASA Telerobotics conference*, 1989. Pasadena, CA.

[RW91] S. Russell and E. Wefald. *Do the right thing.* MIT press, 1991.

[Sac73] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. In *Proceedings of the 3rd Int'l Joint Conference on Artificial Intelligence*, Palo Alto, California, 1973.

[Sac75] Earl Sacerdoti. The non-linear nature of plans. In *Advance papers for IJCAI – 75*, 1975.

[Sat77] M. Sato. A study of kripke-style methods of some modal logics by gentzen's sequential method. *Publications of the REsearch Institute for Mathematical Sciences*, 13(2), 1977.

[Sco70] D. Scott. Advice on modal logic. *Philosophical Problems in Logic*, pages 143–173, 1970.

[Sho88] Yoav Shoham. Chronological ignorance: Experiments in nonmonotonic temporal reasoning. *Artificial Intelligence*, 36:279–331, 1988.

[Sho91] Yoav Shoham. Agent0: A simple agent language and its interpreter. In *Proc. Ninth National Conference on Artificial Intelligence*, pages 704–709. American Association for Artificial Intelligence, 1991.

[Sim82] H. Simon. *Models of bounded rationality.* MIT Press, Cambridge, Mass., 1982.

[Ste81] M. J. Stefik. Planning with constraints. *Artificial Intelligence*, 16:111–140, 1981.

[Sus73] G.A. Sussman. A computational model of skill acquistion. Technical Report AI-TR-297, MIT AI Lab, 1973.

[Tat75] A. Tate. Interacting goals and their use. In *Proceedings of the 4th Int'l Joint Conference on Artificial Intelligence*, pages 215–218, 1975.

[Tat77a] A. Tate. Generating project networks. In *Proceedings of the 5th Int'l Joint Conference on Artificial Intelligence*, 1977.

[Tat77b] A. Tate. Project planning using a hierarchical non-linear planner. Technical Report Report 25, Dept. of Artificial Intelligence, Edinburgh University, 1977.

[Tho90] R. Thomason. Accommodation, meaning, and implicature: interdisciplinary foundations for pragmatics. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 325–363. MIT Press, 1990.

[TSSK91] Becky Thomas, Yoav Shoham, Anton Schwartz, and Sarit Kraus. Preliminary thoughts on an agent description language. *International Journal of Intelligent Systems*, 6(5):497–508, August 1991.

[Var86] M. Vardi. On epistemic logic and logical omniscience. In J. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 293–305, Monterey, CA, 1986. Morgan Kaufmann.

[Ver83] S. Vere. Planning in time: Windows and durations for activities and goals. In *Proceedings, IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 246–267, 1983.

[Wal75] R. Waldinger. Achieving several goals simultaneously. Technical Report Technical Note 107, SRI AI Center, 1975.

[WHR89] R. Washington and B. Hayes-Roth. Input data management for real-time ai systems. In *Proceedings of IJCAI-89*, Detroit, Michigan, 1989.

[Wil83a] R. Wilensky. *Planning and understanding.* Addison Wesley, Reading, Mass, 1983.

[Wil83b] D.E. Wilkins. Representation in a domain-independent planner. In *Proceedings of the 8th Int'l Joint Conference on Artificial Intelligence*, pages 733–740, 1983.

[ZR92] S. Zilberstein and S. Russell. Constructing utility-driven real-time systems using anytime algorithms. In *Proceedings of the IEEE workshop on imprecise and approximate computation*, pages 6–10, 1992.