Motivation-based direction of planning attention in agents with goal autonomy

Timothy James Forester Norman

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

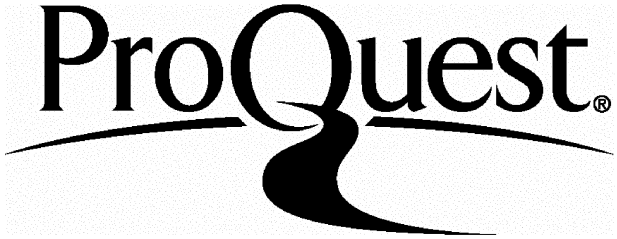
of the

University of London.

Department of Computer Science

University College London

1997



Abstract

The action of an agent with goal autonomy will be driven by goals generated with reference to its own beliefs and desires. This ability is essential for agents that are required to act in their own interests in a domain that is not entirely predictable. At any time, the situation may warrant the generation of new goals. However, it is not always the case that changes in the domain that lead to the generation of a goal are detected immediately before the goal should be pursued. Action may not be appropriate for some time. Furthermore, an agent may be influenced by goals that tend to recur periodically, or at particular times of the day or week for example. Such goals serve to motivate an agent towards interacting with other agents or processes with certain types of predictable behaviour patterns. This thesis provides a model of a goal autonomous agent that may generate goals in response to unexpected changes in its domain or cyclically through automatic processes.

An important effect of goal autonomy is that the agent exhibiting this capability will have a varying, potentially unlimited, but certainly unpredictable number of goals. Goals that hold planning attention consume resources, and real agents are resource bounded. Hence, there is a limit to the number of goals that can hold planning attention before bookkeeping and search operations become the primary mode of activity; i.e. before cognitive overload. In this thesis, an heuristic mechanism is proposed for the directing and limiting of planning attention in agents with goal autonomy. These "alarm processing" mechanisms serve to focus the attention of an agent on a limited number of the most salient goals, and thereby avoid unnecessary reasoning and prevent cognitive overload. In this way, a resource-bounded agent can employ modern planning and reasoning methods effectively.

Acknowledgements

I would like to thank Derek Long for his guidance and supervision. I am grateful for the weekly in-depth discussions that kept the momentum going. Thanks also to Maria Fox for her encouragement and feedback. There are a number of others who have help through discussion and comments, namely Miles Pebody, Luc Beaudoin, Ian Wright, Mike Luck and Mark d'Inverno.

My doctoral studies were funded by the Engineering and Physical Science Research Council. This thesis was generated using the LaTeX2e macro package and the UCL thesis document class provided and supported by Russel Winder.

My (allegedly) intact sanity is due to Vijaya who, after four years of marriage, can now try to forget that I ever did this.

Contents

List of Figures

Chapter 1

Introduction

For an agent to reason effectively, it must first avoid its unnecessary use.

1.1 Objective

The objective of the research presented in this thesis is the direction and limiting of planning attention in agents with goal autonomy. If such an agent is capable of successfully *directing* planning attention, it will focus on the most salient of its goals. If such an agent is capable of successfully *limiting* planning attention, it will focus on a sufficiently small number of goals to prevent its reasoning resources from becoming overloaded regardless of the number of goals it has to pursue.

1.2 A broad context

There is an important trend in the field of Artificial Intelligence (AI) research. The focus is shifting from the investigation of isolated processes such as planning, scheduling or belief revision algorithms to the integration of these various elements into complete "agent" designs (Bates et al., 1992; Georgeff & Lansky, 1987; Shoham, 1993; Wooldridge & Jennings, 1995a; Wooldridge & Jennings, 1995b; Wooldridge et al., 1996). With this shift of emphasis, AI methods are being judged more and more on how they perform in realistic environments, requiring real-time responses. The evolution of planning technology is a case in point. The early planning algorithms, for example GPS (Newell & Simon, 1972), STRIPS (Fikes & Nilsson, 1971), NOAH (Sacerdoti, 1975), NONLIN (Tate, 1977), and TWEAK (Chapman, 1987), and more recently SNLP (McAllester & Rosenblitt, 1991) and UCPOP (Penberthy & Weld, 1992) are principally designed to generate good plans to satisfy a conjunction of goals as efficiently as possible. These "classical planning" techniques have been criticised in a number of ways:

- The planning problem is inherently intractable (NP-hard): There is no guarantee that any planning algorithm will generate a satisfactory solution to a given problem within any

fixed period of time.

- A planner relies on a complete world description: It is assumed that there is a complete and correct model of the domain. This problem has been addressed to a certain extent through making sensory actions available to the planning system so that the agent can "plan to perceive" (Etzioni et al., 1992; Pryor & Collins, 1992a; Collins & Pryor, 1995).

- A planner relies on a complete action description: It is assumed that the effects of the agent's actions are completely and correctly modelled.

- The goals that the planner is designed to pursue are all-or-nothing: There is no representation of the partial achievement of goals. This problem has been addressed to a certain extent by employing decision-theoretic techniques in planning (Wellman & Doyle, 1991; Haddawy & Hanks, 1992). However, these systems do not in any way address the other criticisms of classical planning identified here.

- A planner's activity is essentially "one shot" (Hammond et al., 1995): There is no model of on-going interaction between the planner and its domain.

In the light of these criticisms, there has been some re-emergence of the behaviour-based AI (BBAI) paradigm (Boden, 1994) due, in part, to the success of a number of behaviour-based robots (Brooks, 1991). In the development of BBAI systems, the focus is on the dynamics of a particular agent-environment interaction rather than general algorithms for plan generation (Brooks, 1986; Maes, 1989; Agre & Chapman, 1990; Kaelbling & Rosenschein, 1990; Chapman, 1991; Nilsson, 1994). Typically, an agent is designed with a number of behaviours, each corresponding to a single type of interaction that the agent can have with its environment (e.g. approach-object or run-away-from-object), and some mechanism for selecting between them. For example, Maes (1989) (Maes, 1991) presents a behaviour selection algorithm that uses activation, which flows through a network of behaviours, to select the most appropriate behaviour in the present context; i.e. an associative memory approach to the action selection problem, cf. Norman & Shallice (1986), Anderson (1993), Polk & Rosenbloom (1994) and Cooper et al. (1995). The edges of this network represent the relevant links between the pre- and post-conditions of behaviours; each behaviour being represented as a STRIPS (Fikes & Nilsson, 1971) operator. For instance, the behaviour approach-object will conflict with the behaviour run-away-from-object, and so each behaviour decreases the activation level of the other by a fraction of its own activation level (mutual inhibition). Although these systems have proven effective in the low level control of agent behaviour as predicted by Simon (1981), it is difficult to see how systems that do

not employ knowledge representation and reasoning will scale up to more complex tasks (Ginsberg, 1989; Norman, 1994).

An alternative approach is "reactive planning" (Firby, 1987; Georgeff & Lansky, 1987; Schoppers, 1987). This type of control system provides an agent with a number of low level behaviours (e.g. Reactive Action Packages (Firby, 1987)) with certain characteristics that have the potential to be combined using more deliberative action selection machinery (see Musliner (1994) for a system designed to enable a planner to generate looping reactive plans dynamically). Probably the best known example is the Procedural Reasoning System (PRS) (Georgeff & Lansky, 1987). This control system is capable of more extensive planning as well as reacting to changing situations as it interacts with its environment. PRS is an implementation of a more general Belief, Desire and Intention (BDI) architecture (Bratman et al., 1988; Shoham, 1993; Rao & Georgeff, 1992; Rao, 1996), which is one of a number of broad agent architectures proposed in the literature (see chapter 3). A PRS agent performs simple belief revision, planning and scheduling functions as it interacts with a changing environment, and more recently a multi-BDI-agent system has been developed as an aid to the management of flights in and out of Sydney airport (Rao & Georgeff, 1995).

The integration of perception, various types of deliberative processes and physical action in the development of agents that are designed for useful tasks in realistic environments presents a number of important challenges for AI research. The perceptive processes of an agent must combine with belief revision to maintain a grounded world model that may be used for decision making (Warrick & Fox, 1994). Planning must be controlled so that the agent makes decisions fast enough so that it may act in time (Dean & Wellman, 1991; Musliner et al., 1995). The agent may be required to use its world model to plan communicative acts in negotiating and cooperating with other agents.

The specific problem that is investigated in this thesis is the integration of the goal generation and planning functions within a complete agent architecture.

1.3 The problem

If an agent is required to interact with an environment that is not entirely predictable, a static list of goals is not a sufficiently flexible representation of its purpose. The state of the environment may change at any time so that pursuing a goal may no longer be realistic, or desirable. A single goal may need to be satisfied more than once, or periodically, depending on how the environment, the agent, and the relationship between them (i.e. the domain) change over time. For an agent to be effective in such a domain, it must be *goal autonomous*. In other words, it must be

capable of generating goals on the fly in response to a changing domain, and consequently altering the goal or goals it is presently acting on as its goals and the priorities of those goals change. However, although these requirements may be necessary in agent design, they are not sufficient.

Consider an agent with limited reasoning capabilities; a resource-bounded agent (Simon, 1957). Suppose this agent is capable of generating its own goals in response to changes in its internal state and the state of the external environment; i.e. the agent is goal autonomous. A goal autonomous agent is an agent that will generate goals on-the-fly in response to changes in is environment in the service of a multiplicity of motives (Neisser, 1963; Simon, 1967). It is feasible that such an agent may respond to an event now by generating a goal that does not require action until later that day or next week for example. It may not even be possible to achieve the goal for some time, but the goal exists, and thus will influence the agent's behaviour. Furthermore, the agent may be required to act cyclically to successfully interact with some other agent or process, or to maintain the state of some variable. For example, an agent may need to mitigate hunger at various times of the day, or wish to attend lectures according to some timetable. Such activities must be scheduled along with other non-routine goals. Therefore, at any one time an agent with goal autonomy will be influenced by a varying and potentially unlimited, but certainly unpredictable number of goals.

Once a goal is generated it demands reasoning resources whether or not it is appropriate for the agent to act on the goal now. Suppose the agent has a number of goals, all with various importances and time requirements. This agent must decide which goals should be planned for and acted on at any one time, but computing the best combination of goals to attend to is NP-hard.[1] The agent may be physically capable of achieving all its goals given time, but there is a limit to the reasoning resources that are available for planning, goal management and other deliberative processes, and so there is a limit to the demand for these resources that can be met. If the number of goals undergoing deliberative processing is such that the demand for resources exceeds those available, an agent will be swamped by bookkeeping and search operations to the exclusion of all other modes of activity (Cherniak, 1986; Dennett, 1987). Such a situation is referred to in this thesis as cognitive overload; this is where the agent cannot produce any kind of meaningful decision due to the complexity of the problem and the different concerns involved in generating a solution to this problem.

For an agent with goal autonomy to use planning and other deliberative processes effec-

[1] In fact Bylander (1994) has shown that in general it is PSPACE-complete to determine if a given planning instance has any solutions, and extremely severe restrictions are required to guarantee polynomial time or even NP-completeness. Such severe restrictions make the planning problem trivial.

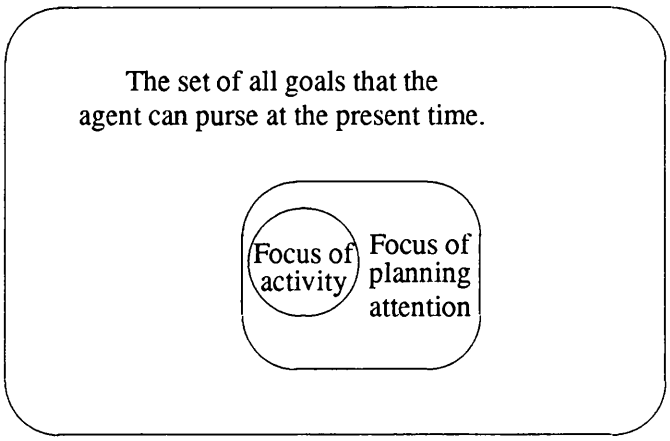


Figure 1.1: Focus of attention: 1.

tively, it must focus the use of its reasoning resources on a limited number of relevant goals so that cognitive overload is prevented (Norman & Long, 1995b). In other words, for an agent reason effectively, it must first avoid its unnecessary use. So, an agent that is designed to interact with a realistic domain must be capable of: (1) generating goals on the fly; (2) limiting the goals that are considered as candidates for action such that cognitive overload is prevented regardless of the number of goals and their distribution over time (this problem *must* be solved heuristically); and (3) directing attention to goals that are most appropriate for action at the present time. The combination of requirements 2 and 3 have the effect of structuring the space of goals that the agent may pursue now in a way similar to that illustrated in figure 1.1. Goals that are candidates for action are a subset of the set of all goals that the agent could pursue: this is the agent's "focus of planning attention". Then the goals that the agent is acting on at present are a subset of the goals within this focus of planning attention; this is the agent's "focus of activity". The goals that lie in this focus of activity are those the agent is in active pursuit of. The problem that is addressed in this thesis is the maintenance of an appropriate focus of planning attention such that cognitive overload is prevented regardless of the number of goals the agent can pursue.

1.4 Contributions

This thesis contributes to AI primarily in that a detailed specification and prototypical implementation of a system for limiting and directing an agent's planning attention is presented: the alarm processing machinery. The design is motivated by issues of computational validity, and to some extent psychological plausibility. The primary requirement is that the processes developed for the generation, suspension, recollection and consideration of goals serve to avoid unnecessary

reasoning.[2]

Various types of goal (Schank & Abelson, 1977; Ortony et al., 1988; Slade, 1994) or types of goal generator (Carbonell, 1982) have been proposed in the literature without any concrete mechanistic explanation of how these different motivators affect the behaviour of an agent. In this thesis, three distinct sources of goals are identified: goals generated through decision, and goals automatically generated (i.e. avoiding the reasoning involved in generating goals through decision) either periodically or according to some timetable (chapter 4). This classification is motivated by both the need to avoid unnecessary reasoning and the need to exhibit certain types of goal directed behaviour, and hence all three types of goal generator have important functional roles in the agent architecture (Norman & Long, 1995a).

Hammond (1989b) (Patalano et al., 1993) presents a mechanism through which an agent that has suspended a goal may recall that goal in the light of an opportunity to achieve it. This opportunistic recollection of suspended goals may serve to avoid unnecessary planning and activity; an opportunity will enable the goal to be achieved with the minimum of further planning effort, and possibly less physical effort (see section 5.1). The alarm processing machinery presented in this thesis contributes to this area of research by providing a more complete account of the recollection of goals in the light of opportunities. Furthermore, the influence of opportunities are seen as one part of a complete goal recollection mechanism which also accounts for the influences of deadlines, delay times, dangers to the timely satisfaction of suspended goals, and prior time commitments (chapter 5) (Norman & Long, 1996).

The need for an agent with goal autonomy to effectively direct and limit its planning attention has been discussed by Simon (1967), Sloman & Croucher (1981), Sloman (1987), and Norman & Long (1995b). To perform such a function, an agent must have "fast but stupid subsystems, including filters for new motives" (Sloman, 1987).[3] Expanding on the filter penetration theory of cognition and affect (Sloman & Croucher, 1981; Sloman, 1992), Beaudoin (1994) (Beaudoin & Sloman, 1993) discusses a mechanism by which the consideration of new goals is regulated by some coarse heuristic filter; the threshold of this filter depending on certain aspects the agent's state including its "busyness". This thesis contributes to this area of research by providing a mechanistic account of how the load on the reasoning resources of an agent can affect

[2]A suspended goal is one that does not hold attention; i.e. does not undergo deliberative processing. The recollection of a goal is the mechanism by which a suspended goal enters a state in which it is considered for activation. The terms "active" and "inactive" are also used to refer to a goal that is and is not within the agent's focus of planning attention. An inactive goal may become active through activation (these terms are explained in greater detail in chapters 4–6).

[3]It should be noted that 'stupid' is an undesirable consequence of such subsystems being 'fast'.

the threshold of such a filter, and more specifically, affect the likelihood of the agent recalling a particular suspended goal (section 6.1). This load on the reasoning resources of an agent (a.k.a. cognitive load) is assumed to be closely related to the time it has committed to action and the importance of the goals, in the pursuit of which those actions are to be taken.

1.5 Thesis outline

The outline of the thesis is as follows. The characteristics of the environment and the relationship between the agent and its environment (i.e. the domain) constrain the design of an agent that must operate in that environment in important ways. Chapter 2 discusses the constraints imposed on agent design due to a real world domain, i.e. a domain the can neither be completely nor correctly modelled, that are of primary interest to the problem addressed in this thesis. In chapter 3, a broad agent architecture, "motivated agency", is presented which provides the mechanisms presented in the following chapters a wider architectural context. This architecture is then compared to two other related agent architectures. Chapters 4, 5, and 6 detail the alarm processing machinery; an implementation of a goal management process which serves to limit the goals attended to by the agent; i.e. provide an agent with a focus of planning attention. These three chapters comprise the core of this thesis, and gradually introduce the various aspects of the alarm processing machinery. Chapter 4 discusses the mechanisms by which goals may be generated and introduces the idea of an alarm which serves to prevent a goal from being processed by the deliberative functions of an agent. In chapter 5, unexpected changes in the agent's internal state and the state of the external environment and their effect on when a goal is considered are discussed. Then, chapter 6 describes the conditions under which a goal is brought to the agent's attention and the subsequent processing of that goal. Chapter 7 serves to summarise the previous three chapters, giving an overview of the alarm processing machinery and an evaluation of the algorithm. Chapter 8 is a concluding chapter, in which the open issues of the management of goals and the limiting of planning attention are discussed.

Chapter 2

The warehouse domain test-bed

In the previous chapter, it was argued that there is a need for comprehensive agent control systems to include processes that limit and direct reasoning attention. Before the broad agent architecture (chapter 3) and the specific contributions of the work (chapters 4–6) are presented, it is important to describe the type of application environment within which such an agent is intended to act. By way of an illustration of the issues involved, consider the two robots Boadicea and TJ, developed at the MIT AI laboratory.[1] These two robots are designed for two different purposes in two different environments. Boadicea is the latest in a series of six-legged robots that is capable of traversing an uneven terrain, including up to 45 degree slopes, at around six inches per second. TJ is a wheeled robot that is capable of being taught the names and locations of places in simple typewritten English, then answering questions about them and navigating to them as requested. Both of these robot designs are effective in the environmental niche for which they are designed, but neither would be effective if their roles were reversed. To fully understand an agent design (or a design in design space (Sloman, 1994)), it is important to understand how an environment (or a niche in niche space (Sloman, 1994)) influences and constrains the design (Hayes-Roth, 1995). Furthermore it is important to analyse the general characteristics of the domain of interest (i.e. the environmental niche, the agent and the relationships between them) before the development of an implementation so that design constraints that are peculiar to the implementation can be distinguished from more general constraints that may lead to general principles of agent design.

In section 2.1 certain characteristics of a realistic environment are described, and how these influence and constrain the design of an agent is discussed (reference is made to a number of implementations documented in the literature). Then section 2.2 describes the warehouse domain; a notional test-bed that has been used in the development and prototyping of the motivated agent architecture. Finally, in section 2.3 the warehouse test-bed is critically examined.

[1] Refer to the URL `http://www.ai.mit.edu/projects/mobile-robots`.

2.1 The agent-environment relationship

In this section, the relationship between an agent and its environment is analysed. This analysis serves to highlight certain motivations for the work. In describing an agent-environment relationship there are two primary considerations: (1) the agent's sensory and affective capabilities with respect to its purpose in the environment; i.e. the interface between the agent and its environmental niche (section 2.1.1); and (2) the dynamics of the environment; i.e. how the environment itself can change independently from the actions of the agent (section 2.1.2). Out of this analysis, a number of constraints on the design of an agent control system that have motivated the development of the goal management mechanisms described in this thesis are identified, see section 2.1.3.

For the purposes of this discussion, something in the agent's environment can be either an agent or an object. The term "agent" is used to refer to something in the domain that has the ability to change its own state and the state of other things in the environment by acting on them, but the state of an object will only change due to the action of an agent or through some process which is governed by the physics of the simulated environment.[2]

2.1.1 Agent-environment interface

An agent interacts with its environment by sensing certain aspects of, and acting to change state variables in the environment. From the point of view of an agent design, the important issues in simulating its sensory processes are: (1) sensor scope; (2) data accuracy; and (3) sensor efficacy. The scope of a simulated sensor defines how much of the environment the agent can detect at any one time. For example, in the blocks world (Rich & Knight, 1991) (illustrated in figure 2.1) the agent manipulating the blocks can detect the positions of all the blocks in its environment at all times. Similarly, Pengi (Agre & Chapman, 1987) can view the state of the entire Pengo game board. In contrast, the nurse maid agent that is designed to manage a simulated nursery (Beaudoin, 1994) has an "eye" that can be moved from one room to another, and only sense the detectable state variables of things in the room it is in. This "eye" must be moved from room to room for the agent to build a model of its environment, but as it is sampling the states of things in one room, the states of things in other rooms continue to change. The longer the interval since the state of something in a particular room was sampled, the more out of date the agent's information about that state variable will be. The unavailability of sensor data can lead to inaccuracies in the agent's model of its environment. A second source of model inaccuracy is inaccurate data from the agent's sensors. For example, the Distributed Vehicle Monitoring Test-bed (DVMT)

[2]It is not claimed that the autonomous generation of goals is a sufficient condition for autonomy (although this has been argued elsewhere (Luck & d'Inverno, 1995)), but for an agent to be autonomous this ability is necessary.

(Lesser et al., 1988) and the Tileworld simulator (Pollack & Ringuette, 1990) simulate sensors with various qualities.

Typically, an agent can only detect the external state of things in the environment; sensors have limited efficacy. For instance, Pengi (Agre & Chapman, 1987; Agre & Chapman, 1990) is an agent that inhabits an environment of ice blocks along with a number of bees. Pengi can only detect the present location of a bee in the environment, and from that and its previous states predict the trajectory of the bee. Pengi cannot detect the internal states of the bee that govern where it will go next. In general, the detectable state of another agent will appear to change stochastically, although patterns may be observed in some circumstances. Furthermore, if the state of an object can be changed by another agent, the behaviour of that object may not be entirely predictable. The depth of an agent's sensory capabilities in a multi-agent environment can be improved through communication. In this way the agent can get some indication of the internal states of other agents in the environment, and hence improve its model, if there is sufficient similarity in the behaviour of the agents.

An agent can attempt to change certain aspects of its environment by acting on itself, objects or other agents in the environment. From the point of view of an agent design, the important issues in simulating the affective capabilities of an agent are: (1) the scope of those affective capabilities with respect to the agent's purpose in the environment; and (2) the efficacy of the agent's abilities. If an agent has a sufficient scope of affective capabilities to satisfy its purpose without the need for cooperation with other agents, it is not necessary for the agent to communicate with other agents. For example, the purpose of the Pengi agent in the game Pengo (Agre & Chapman, 1987) is to collect together a number of magic ice blocks. Pengi must pursue this task while avoiding bees, or kicking ice blocks at them if necessary. There is no communication involved between Pengi and the bees in the game because it is neither necessary nor possible for Pengi to cooperate with bees to successfully collect together the magic ice blocks. If an agent either requires aid from other agents to succeed or wishes to use cooperation to achieve its goals more easily, inter-agent communication is necessary. In fact some tasks are not possible without cooperation and coordination between various, possibly heterogeneous, agents (Jennings, 1995). The second issue in simulating the affective capabilities of an agent is the possibility that an action is unsuccessful (Brooks, 1986). For example, if an agent attempts to pick up an object it may be unsuccessful and have to attempt the same operation two or more times.

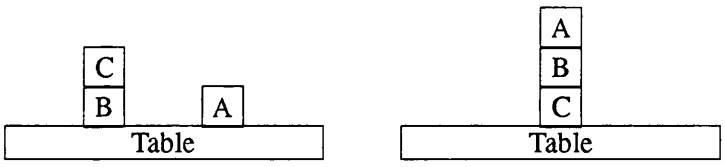
2.1.2 Environmental change

In the construction of an implementation, a simulated environment is commonly modelled as having a, possibly variable, number of objects or agents, each with a number of state variables,

e.g. Shoham (1993), Doukidis & Angelides (1994), and Sloman & Poli (1996). For example, the objects marked A and B in figure 2.1 are blocks of constant size and shape, but in figure 2.1(a) A is on the table and in figure 2.1(b) A is on B; so the position of block A in the environment (a state variable) can vary. Consider a further example taken from Pengo (Agre & Chapman, 1987; Agre & Chapman, 1990). Pengi, inhabits a maze of ice blocks along with a number of bee agents. These bees chase Pengi with the intention of kicking ice blocks at it (an accurately kicked ice block can be fatal). It is important in the design of Pengi to know both what a kicked ice block can do to Pengi and how fast bees can travel around the maze. If a bee can travel significantly faster than a particular penguin design, then that design is not likely to survive long in the environment. So, in the design of an agent for a particular environment it is important to determine not only the types of thing that inhabit that environment, but also how the states of those things can change.

A change in the state of an object or agent in the environment can be modelled by determining what caused the change, and how fast the state changes. The source of change in the state of something in a simulated environment is important in determining the degree of control that an agent can hold over it, and hence is an important influence on agent design. The source of change in an object can be either an agent or some process governed by the physics of the simulated environment, and the source of change in an agent can be itself, another agent, or some environmental process. The simplest environment is one in which there are only inanimate objects in the environment and the agent to be designed for some purpose in that environment is the sole source of change in the states of those objects; this is the case in the blocks world, see section 2.1.3 and figure 2.1. If the state of an object can change due to some process that is initiated by an agent (e.g. a hot cup of coffee will cool in time) the design of an agent for that environment must take account of these effects when deciding how to act and, importantly, when to act (Vere, 1983; Dean et al., 1988; Boddy & Dean, 1994). Once an agent is to be designed to inhabit a multi-agent environment, it is only one of a number of agents that can initiate change in objects in the environment. The degree of control that an agent has over various aspects of its environment will now depend on the relative efficacy of it and other agents in the environment. Furthermore, it may be possible for agents to influence each other either through physical or communicative actions.

The rate of change of state of an object or other agent is important to the design of an agent if that variable is relevant to the purpose of the agent in the environment. The agent must be *fast enough* to respond appropriately to significant changes in the state of things in the environment (see section 2.1.3). Therefore, for an agent design to be effective in a particular environmental



(a) Initial state. (b) Goal state.

Figure 2.1: A simple blocks world scenario.

niche the granularity of its ability to act must at least match the granularity at which the states of relevant objects or agents significantly change. For example, a Pengi design must be able to make decisions and act at least at a comparable rate to the bees (Agre & Chapman, 1987).

2.1.3 Motivating design constraints

The following three design constraints serve to identify critical and motivating issues behind the approach to the design of intelligent systems presented in this thesis.

Multiple and changing goals

Often, AI planning systems assume that the purpose of an agent is to reach a well-defined end point represented by a conjunction of goals that is known in the initial state. In many domains, a goal is characterised as a proposition in the world model to be made true, sometimes qualified by temporal constraints such as deadlines. Then once an agent's goals are satisfied, its task is completed. A classic example is the blocks world domain in which there are three blocks marked A, B, and C on a table, see figure 2.1. A typical task of the agent is to stack the blocks so that block C is on the table, block B is on block C and block A is on block B; i.e. satisfy the conjunction of goals $\texttt{on(A, B)} \wedge \texttt{on(B, C)} \wedge \texttt{on(C, Table)}$ (figure 2.1(b)), where $\texttt{on(x, y)}$ is a valid proposition in the blocks world domain. Once the agent has achieved this state, its task is completed.

Such simple domains (i.e. domains in which the agent is the sole source of change in the states of objects) have proved useful in understanding the complexity of the planning problem, and in the development of various planning algorithms (Fikes & Nilsson, 1971; Allen & Koomen, 1983; Chapman, 1987); see Rich & Knight (1991) or Lee et al. (1993) for a review. However in practical applications, an agent is required to interact with a domain that can change independently from the actions of that agent. In such a domain, a static list of goals is not a sufficiently flexible characterisation of the agent's purpose. The example used by Georgeff & Lansky (1987) in discussing the Procedural Reasoning System clearly illustrates this point. This scenario involves a robot that is employed as an astronaut's assistant in a space station. The robot is asked to get a wrench, and proceeds to pursue this goal. As the robot is fetching the wrench

a warning light is detected indicating that there is a malfunction in one of the reactant control jets of the space station. In response, the robot generates a goal to diagnose the fault, suspending the original task, which is resumed (if still relevant) on completion of that newly generated goal. For the agent to be effective in this domain, it must (1) be capable of generating goals on-the-fly; and (2) be able to alter its focus of attention, i.e. change the goal it is presently acting on, as its goals and the priorities of those goals change. These capabilities are essential for an agent that is required to act in a domain that can change independently from the agent in a way that is not entirely predictable. In the blocks world only the agent itself can cause the state of the domain to change. In the space station scenario, the state of the space station changes independently from the action of the robot.

Unavailability of information

In the previous example from Georgeff & Lansky (1987), the astronaut's assistant responds to a detected warning light while pursuing some other goal. However, the assistant must be in a position to detect that warning light to respond to it. If the warning is not detected and analysed in a reasonable period of time, the fault may become more serious. The detection of such a fault should not be left to chance. So, if changes in the state of an important variable can occur spontaneously or through the action of another agent, and the agent can only sense a limited part of the domain at any one time, then the behaviour of the agent must be influenced by the need to monitor that variable. Consider an astronaut's assistant that has two purposes: the detection, diagnosis and reporting of faults in the space station, and following the instructions of the astronaut, and it is not always in a position to detect whether or not a fault is indicated. For this agent to be effective in its environmental niche, it must check warning lights that indicate faults at appropriate intervals. The rate at which significant changes occur and how long it takes to fix a typical fault as well as the possible cost of failure will govern the frequency at which a variable should be monitored. For example, if faults in the reactant control jets are not attended to within an hour of them occurring, then more serious problems tend to occur. In an agent that has a limited sensory scope there must be motivation directing the agent to ensure important variables are monitored.

This design requirement is additional to the need for an agent to gather information about its environment so that a goal can be achieved (Etzioni et al., 1992; Pryor & Collins, 1992a; Collins & Pryor, 1995). The ability of an agent to generate goals to sample the value of a state variable in the service of some top level goal is an important requirement for an agent acting in an uncertain domain. However, this behaviour is directed by the goals that the agent is presently pursuing. If there is a variable that is important to the purposes of the agent, but is not related to

its current focus of attention, the agent should still be curious about the state of that variable in case it has changed in an important way. There is nothing in the agent's present focus of attention that will motivate this information-seeking behaviour. Therefore, the agent must motivate itself by generating goals to seek information about the state of its environment.

Agent-domain synchronisation

In the blocks world domain, the agent is the sole source of change in the environment state variables; i.e. it is the sole actor, the initial conditions of the environment are stable, and the postconditions of all actions that the agent can perform are stable. An example of an unstable condition is the proposition which states that a certain container of water that has been boiled is hot. As time passes, the boiled water will cool in accordance with the physics of the environment rather than the action of some other agent. If the initial conditions of the environment and postconditions of all actions that an agent can perform are stable, and the agent is the sole actor in that domain then there is no need for it to synchronise its activities with the domain.

If important aspects of the state of the domain are unstable, the agent is able to perform an action with an unstable postcondition, or there is some other agent affecting the domain, to achieve certain goals the agent *must* synchronise its behaviour with the domain. If an agent must act in time to satisfy its goals, the domain of interest is referred to as a real-time, or time critical domain (Laffey et al., 1988; Strosnider & Paul, 1994; Musliner et al., 1995). The ability of an agent to act fast enough and at the right time to achieve its goals is essential for that agent to behave effectively in a time critical domain, whether simulated or real. In the Phoenix project (Cohen et al., 1989), a fireboss agent is required to manage simulated forest fires in an American national park by deploying simulated bulldozers, fire crews, etc. The fireboss reasons about time and in time to control forest fires that evolve independently from the activities of the bulldozers and other agents directed by the fireboss. Pengi must act fast enough to survive in the Pengo game, see section 2.1.2. The nurse maid domain, developed by Beaudoin & Sloman (1993) (Beaudoin, 1994), simulates real-time to a degree. The nursery contains hazards such as ditches, and if a baby falls in a ditch it dies. The nurse maid agent must act fast enough to prevent a baby that is near a ditch from falling in. (Also see Russell & Wefald (1991), and Boddy & Dean (1989).)

In the blocks world domain, the agent manipulating the blocks is in total control of what happens in that domain, and so changes in the domain are synchronised with the agent however long it takes the agent to decide what to do. As a consequence, time can be completely ignored. However, in a domain such as the Pengo game the time that it takes for the agent to make decisions and act on those decisions is critical in determining whether or not the agent is successful.

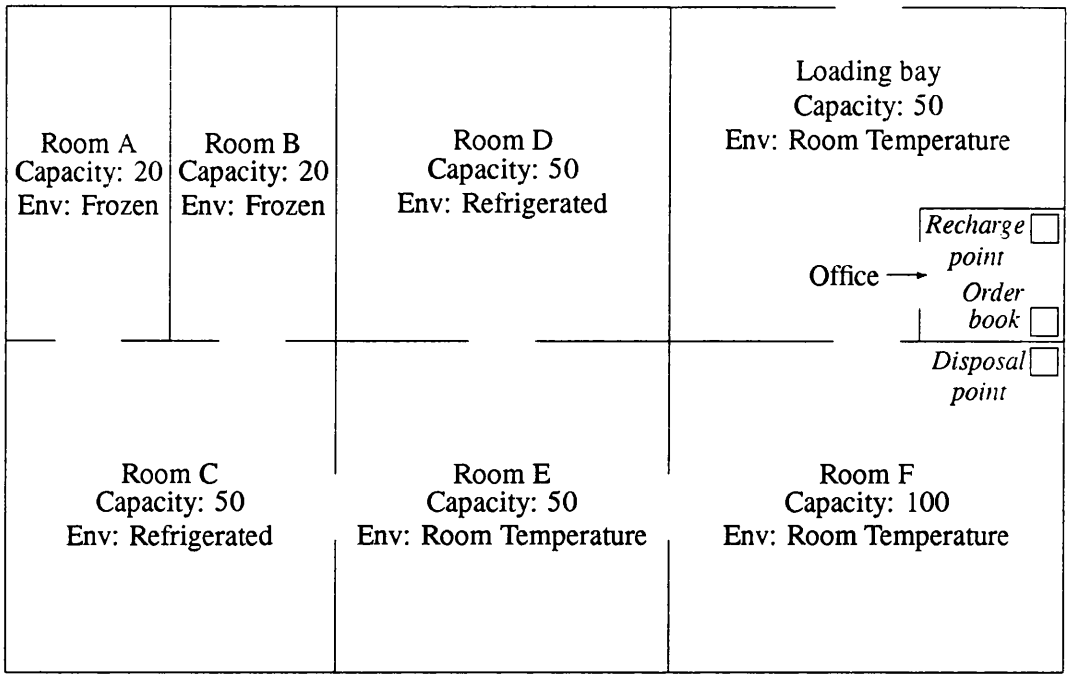


Figure 2.2: A plan view of the warehouse environment.

The nature of Pengo, and *all* real-time or simulated real-time domains, is such that an agent that uses classical planning methods alone to make decisions is unlikely to be successful. The use of behavioural routines such as those in Pengi (Agre & Chapman, 1987; Agre & Chapman, 1990), PRS (Georgeff & Lansky, 1987; Georgeff & Ingrand, 1989), RAPs (Firby, 1987) and the behaviour based layer in the INTERRAP architecture (Fischer et al., 1996) is one approach to the use of AI methods in real-time control; see Musliner et al. (1995) for a review. However, it is not just at the behaviour level that an agent must act in time, but at every level of the agent's activity including planning and goal management. Not only must the agent be physically and 'mentally' capable of performing its purpose in the domain, but those capabilities must be used in such a way that the agent meets real-time constraints.

2.2 The warehouse domain

The prototype warehouse agent discussed in this thesis is implemented in a simple warehouse domain. The implementation does not include a planner or any simulated agent-environment interaction. The implementation concentrates on the goal management mechanisms that are presented in chapters 4–6; it serves as an executable specification of a part of a complete agent and is not intended to be a full simulation.

2.2.1 The warehouse environment

The warehouse environment consists of a number of rooms with various characteristics and doors between these rooms; figure 2.2 gives a plan view of the environment. The room characteristics shown in the diagram are the capacity of the room and the type of environment in the room. The capacity of a room governs how many units of stock can be stored in the room at any time, and the type of storage environment that the room provides governs the type of commodity that is best suited for storage in that room. For example, Room A can hold 20 units of stock and is normally kept at a temperature a few degrees below freezing making it suitable for the storage of commodities such as frozen peas. The loading bay is not normally used for the storage of stock, but is used for the temporary storage of delivered goods and the preparation of orders that are to be collected by customers. Both the delivery of goods from suppliers and the collection of orders by customers are done through the loading bay. The office is not used for the storage of stock at any time. The office contains a recharge point, and the order book with which the warehouse agent communicates with suppliers and customers. The processes concerned with communication between agents is not a concern in this thesis, so the order book is a simplification of the various methods of communication that a real agent may use. The agent uses the recharge point to recharge its batteries. The diagram also includes a disposal point situated in room F, which is used to dispose of commodities that are past their sell by time.

Messages. The order book can be used by the agent to send and receive messages to and from both suppliers and customers. The agent can send requests to suppliers, and either reject or accept messages to customers. There is no model of negotiation between these agents, and the communication is minimal. The main purpose of the order book is for the agent to initiate the process of restocking the warehouse with particular commodities and to receive messages from customers. A message from a customer is assumed to be a request for an order, a correction to a previously requested order or the withdrawal of an order. In all cases the messages are received in a standard form that can be easily interpreted by the agent. For example a request for an order from a prospective customer is a 4-tuple. The syntax used to define this type of message and all other types in this chapter is the Miranda[3] type structure. (A summary of the syntax is given in appendix B.1.)

```
Request (customer-identifier, order-number,
         time-of-arrival, [(commodity, amount)])
```

[3]Miranda is a trademark of Research Software Ltd.

The `customer-identifier` is a unique identifier that distinguishes a particular customer, the `order-number` is an integer that identifies the particular order, the `time-of-arrival` is the time that the customer will arrive to collect the order, and `[(commodity, amount)]` is the order itself which is a list of commodities and the number of units of that commodity required.

Stock. A stock object is a tuple that indicates a unit of stock in the warehouse:

```
Stock (commodity, sell-by-time)
```

The `commodity` indicates what the type of stock is and the `sell-by-time` indicates the time before which the stock unit must be sold. For example, a particular unit of frozen peas may need to be sold by Tuesday.

Agent. An agent can be the warehouse agent, a customer or a supplier. Normally, customers and suppliers are only found at the entrance to the warehouse in the loading bay.

Resource. A resource is indicated by a square in figure 2.2. The resources available are the recharge point, the order book and the disposal point.

Room. A room object is a 8-tuple that identifies the room and what it contains:

```
Room (room-identifier, capacity, environment-type,
      temperature, [door], [agent], [resource], [stock])
```

The `room-identifier` indicates which room is being referred to, the `capacity` is the capacity of the room, the `environment-type` is the type of environment that this room is designed to provide, the `temperature` is the actual temperature of the room, `[door]` is a list of doors which indicates the rooms that can be moved to from this one, `[agent]` is a list of agents in the room, `[resource]` is a list of resources available in the room, and `[stock]` is a list of stock units in the room. A room would be referred to as a compound object by Sloman & Poli (1996), i.e. an object that can contain other objects.

2.2.2 The warehouse agent

When the warehouse agent is in a room it is free to sample the detectable state of any object or agent in that room. For example, if the agent is in the office and wishes to know the state of the order book, it can sample the state of that object, and in this way receive all the new messages

from customers and suppliers. If the agent wishes to know the state of stock levels in a particular room, it must first move to that room and then sample that aspect of the state of the room. The agent will only detect the arrival of either a customer or a recently delivered supply of stock by moving to the loading bay and sensing the state of that room.

The warehouse agent has a number of purposes in the warehouse environment, for example it must ensure that there is an adequate level of stock in the warehouse, meet customer orders, etc. These are the "motives" of the warehouse agent; for a discussion of what is meant by the term motive, refer to section 3.1.1.

Charge

As the agent acts in the environment it uses up a simulated charge. This charge level must be refreshed periodically.

Curiosity

If the purpose of an agent requires it to monitor a variable in its domain and the state of that variable is not always within its sensory scope, then the agent may need to act to sample the state of that variable in some circumstances (see section 2.1.3). The warehouse agent is "curious" about particular variables in the domain and will, if necessary, move to a location simply to sample the state of an object that can be detected from that location.

The warehouse agent will be curious about the temperatures of rooms A B C and D, and the state of the order book. These variables are essential for the agent to ensure that the stock in the warehouse is being stored at appropriate temperatures and for the agent to know about orders requested by customers. (For example, if the temperature in room A indicates a fault in this freezer room, the agent may move the stock to room B as a temporary measure while the temperature control mechanism is fixed; see section C.12.) Furthermore, the agent will be curious about whether or not a supplier has arrived with new stock around the times that the agent has requested stock to be delivered and curious about whether or not a customer has arrived to collect an order.

Maintain stock

For the agent to satisfy the orders placed by customers there must be sufficient stock available in the warehouse. However, the agent will not necessarily have sufficient time to order the required stock from suppliers and have it delivered to the warehouse between the time that an order is received and the customer arrives for the order. There is a significant delay between ordering new stock and its arrival. As stock is sold the agent must replace it, but also ensure that the capacity of the warehouse is not exceeded. Furthermore, if the agent finds that the levels of

a particular commodity are below some threshold, it should respond to this event by ordering new stock as soon as possible. Such an event may occur during an unusually high demand for a particular commodity, e.g. ice cream in a heat wave. This threshold can change as the normal consumption rate of that commodity changes. For example, the consumption rate of ice cream will typically drop in winter, and this threshold will drop with it. (Hammond et al. (1995) refer to this type of behaviour as the agent acting to "stabilise" its environment.)

The agent will respond if a commodity is detected in a room with an inadequate environment type. For example, if the agent detects that there are three units of frozen peas in the loading bay, it will wish to remove those stock units to a room with a frozen environment type. This ensures that the agent prevents the stock that is currently in the warehouse from degrading too rapidly.

The implementation at present assumes that the agent uses a single supplier, and the cost of the commodities ordered are not a factor in deciding the volume of stock to order. However, if the agent is able to negotiate for the best deal with a number of suppliers, the agent may be motivated to minimise the cost of restocking the warehouse. Commodities from different suppliers may vary in price and quality and the suppliers may vary in reliability. The agent can use this information to decide which supplier it should place an order with. With a model of negotiation, the agent may also set a selling price with a customer to ensure a good profit is made.

Prepare orders

To ensure that an order is successfully completed, the commodities requested by the customer should be ready in the loading bay when the customer arrives. The customer will only stay in the loading bay for a limited period of time and if the agent does not have the order ready during that time window, the customer will leave without the order. The agent must synchronise its behaviour with the customer for orders to be satisfied. Again, if the agent is able to negotiate with customers, the agent may gain something if the order is late. For example, a customer may pay less if the agent is late or provides an incomplete order, but the agent must still act to synchronise its activities with its customers.

The agent has a limited ability to vet the reliability of its customers. If the customer does not collect an order, the agent puts a "black mark" against that customer which may cause the warehouse agent to reject subsequent orders. If the customer cancels previously requested orders too frequently this may also cause that customer to be considered unreliable by the warehouse agent.

Tidiness

If the agent orders stock and that stock is not used before the sell by time, it is not possible for the agent to use that stock to complete an order placed by a customer. Furthermore, that stock unit remains in the warehouse taking up a unit of space. The agent will be motivated to clear out any old stock that can no longer be used by disposing of it in the disposal point located in room F.

2.3 Discussion

2.3.1 Limitations of the warehouse domain test-bed

As indicated in section 2.1.3, if the agent is provided with a model of communication the assumptions made in simulating inter-agent interaction in this implementation may be lifted. However, the lack of a model of communication and cooperation does not detract from the efficacy of the goal management machinery presented in this thesis. Goals that motivate an agent to communicate with others are no different; they involve reasoning and planning effort, and so also require management. The integration of planning communicative actions and negotiation is itself an active area of research in distributed artificial intelligence (Wooldridge & Jennings, 1995a).

The present domain loosens the assumption that the agent has a complete and correct model of its domain with which to make decisions. The agent has a limited sensory scope and must move to a particular location to detect the state variables that can be sensed from that location. So, at any one time the majority of the present state of the environment is unavailable to the agent. However, if the agent can sense a particular state variable, the data is assumed to be perfect. The assumption that the data from the agent's sensors is perfect still holds.

The warehouse agent is the primary actor in the domain. Other agents can request orders, collect orders and deliver new stock, but the state of the warehouse is relatively stable (cf. Agre & Chapman (1987)). There are few immediate time critical goals that the agent can generate in this environment; i.e. little requirement for reactive behaviour. However, the focus of the research is to manage longer term goals and longer term time constraints rather than the timely reaction to short term changes in a dynamic environment. Therefore, the domain is better suited to the investigation of goal management machinery rather than behaviour selection machinery, although both have their place within a complete agent architecture.

2.3.2 Conclusion

The warehouse environment constrains the design of an agent in three important ways. (1) A warehouse agent has limited control over its environment; it must synchronise its activities with customers arriving to collect orders. (2) The availability of information is limited, thus the agent

must motivate itself to maintain an up to date record of particular state variables. (3) The agent has limited sensory capabilities and the domain is able to change independently from the actions of the agent, so it is not possible for a warehouse agent to know in advance all the goals it will wish to pursue during its interaction with the warehouse environment. The agent must be able to generate goals on-the-fly in response to a changes in the warehouse domain. Therefore, the warehouse domain provides an acceptable test-bed for investigating agent designs that satisfy these constraints on machinery for the direction of planning attention.

Chapter 3

An abstract agent architecture: Motivated agency

This chapter presents an abstract architecture for the development of artificially intelligent systems capable of generating and effectively managing their own goals. The intention is to provide a framework for the development of a computational model of goal generation and management. The architecture, "motivated agency", is described in terms of the elements and processes of a motivated agent (section 3.1). The notation used in this chapter to specify the types of elements and processes of this and other agent architectures is the Miranda^{TM} type and function structure (summarised in appendix B.1). This notation is used for clarity and conciseness, and as an indication of the first steps towards a full implementation (a prototypical motivated agent is implemented in Miranda and presented in detail in appendicies B and C). In section 3.2 motivated agency is compared to the IRMA architecture (Bratman et al., 1988), an architecture based on beliefs, desires and intentions (also see Rao & Georgeff (1992) and Haddadi & Sundermeyer (1996)), and in section 3.3 the Motive Processing Architecture (MPA) (Beaudoin & Sloman, 1993; Beaudoin, 1994). There are a number of related agent architectures (described in varying degrees of detail in the literature), and other relevant work in both psychology and AI. However, only IRMA and MPA are discussed here because the motivations behind their development include the need to limit computationally intensive processes such as planning and scheduling. A third system that addresses this "control problem" (Hayes-Roth, 1985) is the Adaptive Intelligent System (AIS) architecture (Hayes-Roth et al., 1989; Hayes-Roth, 1995). AIS is a blackboard-based architecture that uses dynamic control plans to guide these meta-level decisions (i.e. deciding what goal(s) to focus attention upon). Refer to Beaudoin (1994, chapter 2) for a detailed discussion and criticism of AIS and its comparison with MPA.

Note that in addition to this and the previous chapter, other related literature is discussed at the end of chapters 4, 5 and 6. This related work is best presented in the context of each of these

chapters rather than collected together in a single literature review.

3.1 Motivated agency

3.1.1 Motives

Motives cause an agent to act: they have no other purpose. The function of a motive is to observe the environment and the internal state of the agent and ensure that a particular interest is served. For example, the warehouse agent has a motive to maintain the levels of stock in the warehouse (see section 2.2.2). This motive will ensure that changes in the environment and in the internal state of the agent that are relevant to this interest are detected. Then, if necessary, specific goals will be generated that have the potential to direct the activities of the agent to support this interest. Consider an agent that is interested in maintaining the widget stock in a warehouse. As widgets are sold, the stock must be replenished by ordering new widgets from a supplier. This motive will influence the agent by generating goals to order new stocks of widgets at appropriate times. Furthermore, widgets may need to be kept refrigerated to ensure that their quality does not degrade too quickly. If the agent detects that a unit of widgets is in a room that is above 5°C, the motive will influence the agent by generating a goal to move this unit of widgets to a room with more appropriate storage conditions. A motive is a mapping from the beliefs of the agent about the state of the domain to a, possibly empty, list of motivated goals (`[mgoal]`).

```
motive == [belief] -> [mgoal]
```

The beliefs are a list of propositions that represent the internal state of the agent and the state of the external environment that can be perceived by the agent. As time passes, the agent's beliefs change, and hence a motive may be triggered to generate a motivated goal (see section 3.1.5). The generation of a goal may involve making a decision on what goal to generate and under what conditions it should be acted on; the effort involved in this process may vary, see sections 4.4 and 4.5.

3.1.2 Motivated goals

A motivated goal (`mgoal`) is a goal which is associated with some motivation. A goal is assumed to be a proposition that the agent wishes to make true (i.e. a partially specified state of affairs) at some point in the future, see section 4.1. The motivation associated with a goal changes over time as the agent's beliefs change and reflects the relevance of that goal to the agent (see section 3.1.3).

```
mgoal == (motivation, goal)
```

If the motivation level of one goal is greater than that of another the agent is more motivated to act on the former rather than the latter. Consider a warehouse agent that has two motivated goals, one to have prepared an order in a few hours time and one to tidy away some old stock from a room in the warehouse. Tidying the warehouse is less important than preparing the order, but it is more relevant for the agent to tidy the room now than to prepare an order for a customer that will not arrive for a few hours. So, the motivation to tidy the room may be greater than the motivation to prepare the order (see section 3.1.3).

3.1.3 Motivation

A motivation is a function that may be used to *heuristically* evaluate some intensity level. The intensity of a motivation is designed to reflect the relevance of an associated goal to consideration at the present time. Intensity will change as the agent's beliefs change and has no meaning unless it is associated with a goal.

```
motivation == [belief] -> intensity
```

The example discussed in section 3.1.2 illustrates the two dimensions of motivation: an estimate of the intrinsic importance of achieving the goal and its temporal relevance (i.e. how relevant it is for the agent to act on the goal at the present time). The example indicates that an agent may be more motivated to act on a goal with low importance but high temporal relevance than a goal with high importance but low temporal relevance. Sloman (1992) uses the example of "stopping for a meal because one has plenty of time before the important meeting" to illustrate a similar distinction between the importance of a goal and its relevance for action in the present. In generating a value for a motivation it is important to find a balance between the influence of these two dimensions on intensity. Suppose that estimates of subjective importance have too great an influence on the intensity of a motivation. In this case, the agent will tend to consider nothing but the most important goals regardless of when these goals are to be achieved. Conversely, if the temporal relevance of the goal has too great an influence on intensity, the agent will tend to consider only its most urgent goals regardless of their importance. Such pathological behaviour is undesirable.

How may the importance and temporal relevance of a goal be combined so that the relative motivations of two different goals can be compared? Grune (1987) argues that to compare two objects, a and b, the first with a vector (x_a, y_a) and the second with vector (x_b, y_b), with no other knowledge available about the values of a and b, the only meaningful combination of the x and y dimensions is multiplicative; i.e. a has a higher value than b if $(x_a y_a > x_b y_b)$. Therefore, since there is no further information available to indicate a different combination of the subjective im-

portance of the goal and the relevance of that goal to the present, an agent's motivation towards acting on a goal is taken to be the multiplicative combination of its estimate of these dimensions.

3.1.4 A motivated agent

A motivated agent is distinctive in that it performs two important functions: goal generation and goal activation. A motivated agent is driven, not by a conjunction of top level goals, but by a number of motives that have the capacity to generate motivated goals in response to detected changes in the domain. This process is referred to as goal generation, and motivated goals that are generated through this process are added to the agent's list of motivated goals.[1]

```
generation :: [motive] -> [belief] -> [mgoal]
generation ms bs = concat [m bs | m <- ms]
```

The second distinctive function of a motivated agent is the activation of goals. Only if a goal is active will the agent act on that goal. A goal will be activated if: (1) the intensity of the motivation to acting on the goal is sufficient to exceed a threshold; and (2) the agent decides that it is relevant to act on the goal now. If the intensity of the motivation associated with a goal exceeds the threshold, the goal triggers. The threshold is derived from the state of the planner and is manipulated to control the triggering of goals. Typically, if the threshold is high, goals will be less likely to trigger, and hence less likely to be considered for activation.[2]

```
triggering :: [mgoal] -> [belief] -> threshold -> [goal]
triggering mgs bs t = [g | (m, g) <- mgs ; m bs > t]
```

If a goal triggers, it is considered by a deliberative process, and if selected it is added to the agent's focus of planning attention (see section 6.2.1). This deliberative process checks the conditions under which the goal was generated. If these conditions still hold, and the agent decides that the goal should still be acted on, it is passed to the planner.

---

[1] The process of generation is represented as a Miranda™ function. The first line is the type of the function: i.e. the `generation` function has two arguments, a list of motives (`[motive]`) and a list of beliefs (`[belief]`), and evaluates to a list of motivated goals (`[mgoal]`). The second line specifies the function. In this case the function uses list comprehension, where `[m bs | m <- ms]` corresponds to the function of taking every motive (`m`) in the list of motives (`ms`) and evaluating each one on the basis of the agent's current beliefs (`bs`). This produces a list of motivated goals for each motive (i.e. a list of lists), which are concatenated to form a single list of newly generated motivated goals using the standard Miranda function `concat` (see appendix B.1).

[2] This definition again uses the Miranda list comprehension mechanism. The motivation function (`m`) of each motivated goal (`(m, g)`) in `mgs` is tested to see if, when evaluated in the present state, the intensity of the motivation exceeds the threshold, `t`; i.e. `m bs > t`. If the intensity exceeds the threshold, the goal associated with that motivation function is returned in the resultant list of goals.

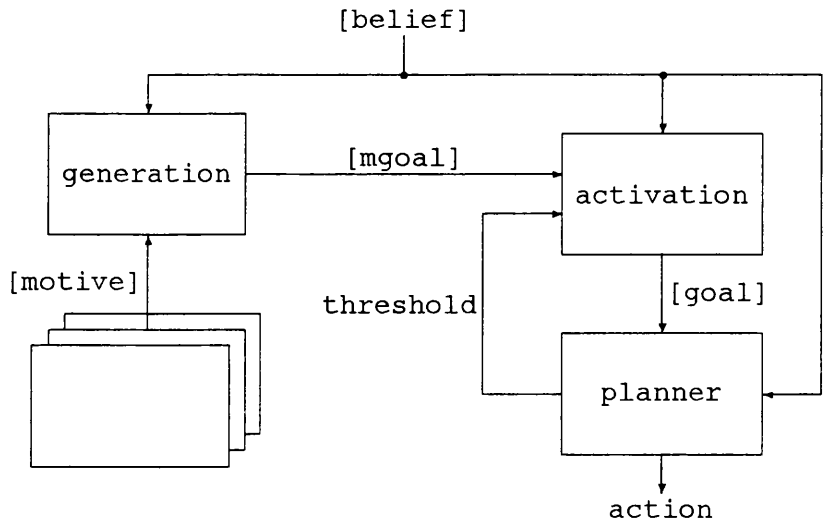


Figure 3.1: A process diagram of the motivated agent architecture.

```
considergoal :: [belief] -> [goal] -> [goal]
```

So, the activation process acts as a two-stage filter where only goals with an associated motivation that exceeds the threshold and are considered relevant for action will pass.

```
activation :: [mgoal] -> [belief] -> threshold -> [goal]
activation mgs bs t = considergoal bs (triggering mgs bs t)
```

Only if a goal is activated will it influence the planning processes of the agent: i.e. the activation of a goal brings that goal to the attention of the agent. So, if the level of the threshold depends on the load on the agent's planning processes, the goals that are active should be: (1) sufficiently small in number so that the planner is not overloaded with tasks to achieve; and (2) the most relevant goals to the agent, i.e. the goals that the agent is most motivated to achieve. On activation, a goal is added to the store of goals which constitutes that agent's focus of planning attention. The function of the planner is then to direct action in the pursuit of its active goals. However, its specific capabilities are not prescribed by the architecture. (The role of the planner is discussed in more detail in section 7.3.)

3.1.5 The motivated agent goal processing cycle

A motivated agent processes its goals to maintain a limited and directed list of activated goals (i.e. a focus of planning attention), see figure 3.1. (The function of the planner is not considered in this summary, but may either operate in parallel with the goal processing machinery, or be called as a subroutine between the steps 3 and 4.)

1. The agent generates a list of motivated goals on the basis of changes in its beliefs; i.e. the `generation` function. These are added to its list of motivated goals for evaluation and possible triggering.

2. The motivation of each motivated goal is evaluated and compared to the current threshold level; i.e. the `triggering` function.

3. These triggered goals are then considered for activation; i.e. the `considergoal` function. If the agent decides to activate a goal, it is added to its focus of planning attention. If the agent decides that the goal is no longer required, it is deleted. Otherwise the motivated goal is reset to be considered some time later (this is referred to as mitigation, see section 6.2.3).

4. Update both the threshold and the agent's beliefs and repeat from 1.

3.2 Belief, desire and intention architectures

In this section, a number of agent architectures are discussed under the general class of belief, desire and intention (BDI) architectures with specific focus given to the IRMA architecture (a good example of a BDI agent architecture). The same notation as that used in section 3.1 is used in this and the following section (section 3.3) so that the different architectures may be compared more easily.

3.2.1 Overview of BDI-architectures

In a BDI architecture, the state of the agent is represented by three structures: its beliefs, desires and intentions. The beliefs of an agent are its model of the domain, its desires provide some sort of ordering between states, and its intentions are the things it has decided to do. The intentions of a BDI agent may be defined at various levels of abstraction; for example, an agent may intend to buy a particular book, but may not have decided which book shop to buy it from. The intention structure described by Bratman et al. (1988) is similar to a hierarchical plan, in that operators at various levels of abstraction in a plan and intentions in an intention structure have similar properties. An action in a hierarchical plan typically represents a task that the agent has committed itself to in the pursuit of some goal(s) as does an intention in an intention structure.[3] An agent's intentions are described as being "structurally partial" and "temporally partial" (Bratman et al., 1988). Intentions are structurally partial because they may require refinement before action can

[3] An agent may have a plan recipe that may be relevant to a goal, and to which the agent has not committed itself. However, in selecting such a recipe or an operator in the process of planning, the agent commits itself to that action as its intended method of achieving the goal.

commence. Intentions are temporally partial because not all time is necessarily accounted for. However, there is a further dimension to the temporal partiality of a plan that is not mentioned: The temporal order in which actions within a plan are to be performed may be partially specified (intentions may be partially ordered). Thus, the intention structure generated in the IRMA architecture (Bratman et al., 1988) represents a linear hierarchical plan of action. In more general terms, a number of precedence constraints may exist between the intentions in an intention structure and between intentions and fixed time points. Similarly, it is common for actions within a non-linear hierarchical plan to be constrained to be performed before other actions in that plan or certain fixed time points without a total order between these action being available. A BDI agent gradually refines its intentions to primitive actions that can be executed. Rao & Georgeff (1992) (Georgeff & Lansky, 1987; Georgeff & Ingrand, 1989; Rao & Georgeff, 1991) characterise a BDI agent as one that performs a function similar to the following.

1. The agent generates a list of options that are available to it. These options are the means by which its current intentions can be satisfied and new options generated on the basis of its beliefs and desires (see figure 3.2, reproduced from Pollack et al. (1994)). The options available to the agent that are the means by which to satisfy its current intentions are generated through means-ends reasoning (a method of refining current intentions using plan recipes).

   ```
   refinement :: [intention] -> [belief] -> [option]
   ```

   The options available to the agent that are alternatives to its current intentions are generated on the basis of its current beliefs and desires.

   ```
   generatealternatives :: [desire] -> [belief] -> [option]
   ```

2. A subset of these options are then selected for deliberation using some heuristic selection function. The selection is made on the basis of the agent's current intentions (Bratman et al., 1988). (An option is passed if it is compatable with current intentions; a selection processes that is subject to possible override.)

   ```
   select :: [option] -> [option] -> [intention] -> [option]
   ```

   The new intentions (i.e. the options that the agent has decided to adopt through filtering and deliberation) are added to the intention structure.

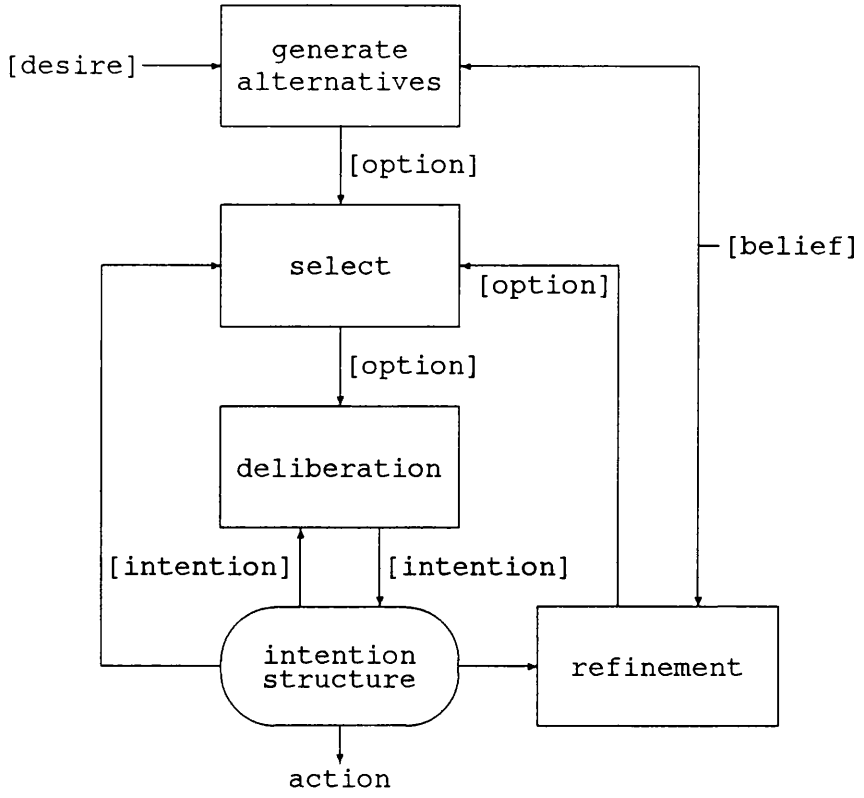


Figure 3.2: A process diagram of a BDI-type architecture (IRMA).

3. If there is an atomic action that can be performed within the intention structure, it is executed.

4. Then, if the agent has satisfied an intention or decided that an intention can no longer be satisfied, that intention is dropped.

5. The agent's beliefs are updated, and the cycle is repeated from 1.

3.2.2 A comparison of the IRMA and motivated agent architectures

The focus of the IRMA agent architecture is to investigate the role of intention in the direction of the activities of an agent and the relationship between beliefs, desires and intentions.[4] The focus of the motivated agent architecture is to investigate the generation and management of multiple long, medium and short-term goals and the direction of planning attention. The intention structure within an IRMA agent represents a linear hierarchical plan of action, and the processes of

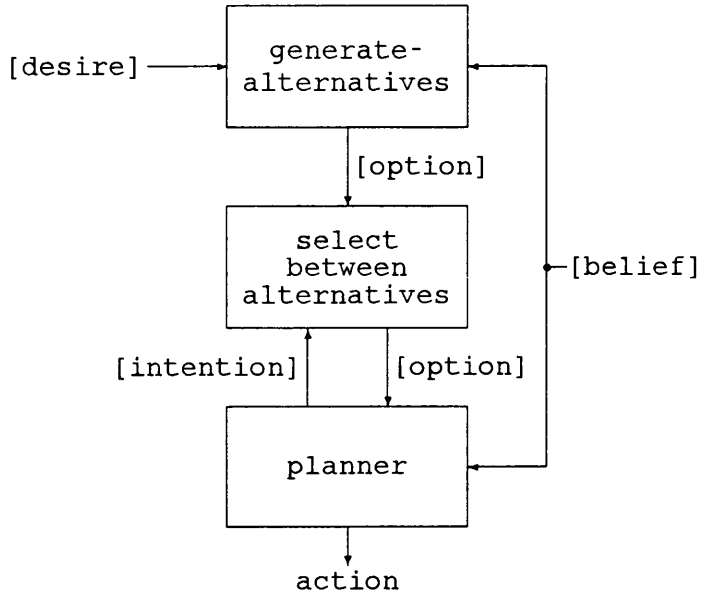
[4]The IRMA architecture is implemented using the Tileworld system (Pollack & Ringuette, 1990). The architecture differs from PRS (Georgeff & Lansky, 1987) in the use of a filter mechanism in the option selection process. Also note that neither IRMA nor PRS do any planning, they use means-ends reasoning to select from a library of plan fragments (or procedures, hence the name Procedural Reasoning System, PRS) in the refinement process.

refinement. selection of options and the insertion of these options into the intention structure is a type of hierarchical planning. However, an important difference between an IRMA agent and a hierarchical planning algorithm is the possibility that options not derived from the refinement process may be introduced as intentions into the intention structure (i.e. generate alternatives. figure 3.2). and that these new options may be selected with the options that the agent generates through refinement.

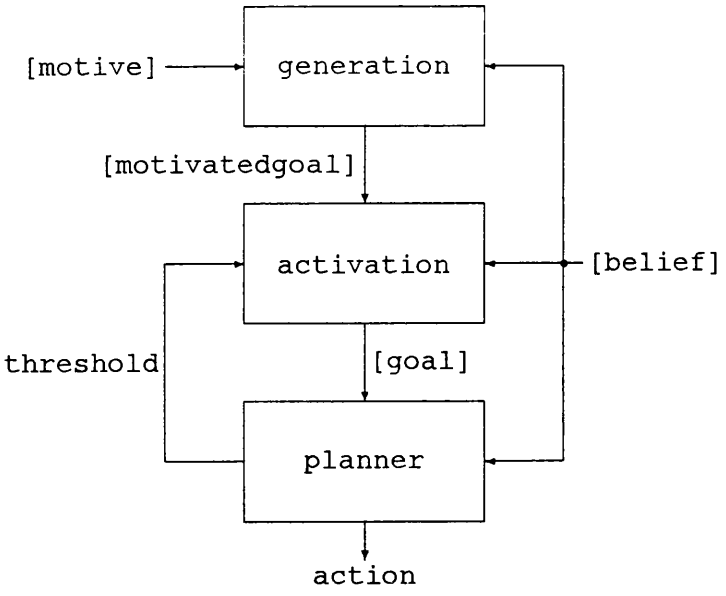
The motivated agent architecture does not assume any particular type of planning process (other than that it should conform to required input and output types). Primary consideration is given to the processes involved in suggesting alternative goals for the agent to pursue. So, to compare the IRMA architecture and the motivated agent architecture, figure 3.2 is simplified to figure 3.3(a). Note that the process of selecting between the options generated through the generate-alternatives process in the BDI architecture described by Bratman et al. (1988) is similar to the filtering of new goals in a motivated agent architecture. (The select process in figure 3.2 is a combination of both the "select between alternatives" process in this figure, and the selection between different possible refinements of an existing intention. the latter being a function of the planner.) The difference between the two mechanisms is in the characteristics of the filtering process itself. In IRMA, options are filtered on the basis of how similar they are to current intentions, and so the feedback from the planner is the agent's set of current intentions. In the motivated agent architecture, goals are filtered on the basis of their importance, temporal relevance and the current load on the cognitive resources of the agent, and so the feedback from the planner is some measure of its load.

Suppose that an IRMA agent is presented with a situation in which there are a large number of goals that it may pursue. The generate alternatives process will generate all the goals that the agent may pursue at the present time through some mechanism that examines its beliefs and desires. These options are then heuristically evaluated in some way on the basis of the agent's current intentions. The agent then deliberates about the options chosen and adds them to the intention structure. The agent's beliefs are updated and again used to test the conditions for goal generation. Therefore, a BDI agent operates in a generate and test cycle. The computational resources consumed in this process depends on: (1) the cost of generating an alternative option; and (2) the cost of evaluating these options.

In the IRMA architecture, an option is generated in response to a set of conditions that hold in the domain. Therefore, the generate alternatives process (figure 3.2) is similar to a set of "Noticers" (Wilensky, 1983, pp. 22–25). "A Noticer monitors changes in the external environment and in the internal states of the system. When it detects the presence of something that it was



(a) A simplified IRMA architecture.



(b) The motivated agent architecture.

Figure 3.3: The relationship between IRMA and motivated agency.

previously instructed to monitor, it reports this occurrence to the source that originally told it to look for that event." All top level goals that can be generated by such an agent must therefore have an associated set of conditions that, in all circumstances require the generation of that goal.[5] Furthermore, this set of conditions *must* be explicitly represented by the agent. If deliberation is required for the agent to determine whether or not a goal is required, then the computational requirements of the option generation mechanism may not be predicted. Suppose that the warehouse agent (see section 2.2.2) receives a request from a customer to fulfill an order. The agent is designed so that it will only generate a goal to fulfill the order if the customer is not considered unreliable and the order is profitable. For it to generate such a goal it must maintain a record of what customers are and are not reliable (e.g. maintain a black list of customers). However, the agent must also know *a priori* if an order is profitable; it must have some explicit representation of what orders can be classed as profitable and a mechanism for testing any order without deliberation. This may not be possible. Market conditions may change continually, and the agent may need to take into account the resources available (e.g. stock in the warehouse) at the time at which it may be required. There is a danger that the maintanence of the information required to enable such goals to be generated in this way will become difficult to manage as the number of possible goals that the agent can generate increases.

A more practical alternative is to restrict the goals generated in this way to those that simply motivate an agent to *consider* generating a goal of a particular type in the first instance. Suppose that the receipt of a request to fulfill an order triggers the generation of a goal to consider generating a goal to fulfill this order. Then, if this goal is adopted and the agent subsequently decides that a goal is required, a goal to have fulfilled the order may be generated. The IRMA architecture does support this to some extent (although the possibility is not discussed). Suppose that a goal to have considered accepting an order is generated, and processed in the normal way. If this passes the filter and is achieved, a consequence of achieving this goal may be to alter the internal state of the agent in such a way that the conditions for the generation of a goal to have prepared the order are satisfied. However, an important consideration is that the time at which a goal is generated is not necessarily the time at which it is relevant for consideration; the customer may plan to arrive to collect the order in three weeks time. This may only become apparent once the agent has considered generating the goal. The temporal context in which the agent considers it

[5] An agent may generate goals in the process of planning (such goals are typically the unsatisfied preconditions of actions in a plan). The agent has decided to achieve these goals in order to satisfy some top level goal that was generated in response to some change in the internal state of the agent or the perceived state of the environment. These goals are therefore *material* to the top level goal.

relevant to act on this goal must play a role in setting the conditions for it being adopted as an intention. If the temporal relevance of this goal does not play a role in its generation, it may be adopted at an inappropriate time. If the goal is adopted too early, there is no point in the agent acting on it at the present time, but this goal will consume computational resources: all intentions must be scheduled, refined and the possible refinements evaluated. Therefore, the agent is forced to perform reasoning that is unnecessary at this time. If the option is adopted as an intention at a more appropriate time, this unnecessary reasoning could be avoided. However, the influence of time on goal generation or recall is little considered in the literature.

Bratman et al. (1988) describe a filter-based option selection function that is designed to "reduce the amount of computation in practical reasoning" (Bratman et al., 1988, p. 351). This filtering mechanism comprises of a compatibility filter and an override filter operating in parallel. The compatibility filter passes options that are compatible with the agent's current intentions, and the override filter passes options that may be incompatible but are sufficiently relevant to warrant consideration. If the override filter is over-sensitive to alternative options, the IRMA agent will tend to be too easily distracted from its present intentions. However, if the override filter is too insensitive to alternatives, the agent will behave in a single-minded manner. The options that pass either of these filters are then considered by a deliberative process and, if selected, are added to the agent's intention structure. A potential difficulty with this filtering mechanism is that the behaviour of the filter does not depend on the load on the agent's computational resources. Suppose that as time passes, a number of alternatives are generated that are all compatible with the agent's current intentions. As the size of the intention structure increases, the computational requirements of scheduling, refining and evaluating these intentions will increase. This may lead to cognitive overload (see section 1.3).

Pollack et al. (1994) describe some experiments performed using an implementation of the IRMA architecture. These are designed to illustrate the effects of commitment to current intentions on agent performance in the Tileworld. The sensitivity of the filter override mechanism is varied. This varies the likelihood of an alternative option that is unrelated to the agent's current intentions being considered. The more alternatives that are considered, the more likely it is that the agent redirects its activity towards one of these alternatives. It is shown that, to a limit, increased commitment to current intentions (an decrease in the sensitivity of the filter override mechanism increases the commitment to current intentions) increases the mean effectiveness of the agent to a degree. Furthermore, this effect is shown to be consistent as the "dynamism" (or average rate of change) of the environment changes. However, the primary effect of modifying the sensitivity of the filter override mechanism is to *limit* the number of alternative options

that the agent considers. The secondary effect is to *direct* attention to those alternatives that are consistent with current intentions. Thus, it is not clear whether the increased mean effectiveness is due to the limiting of attention *per se*, or to alternatives that are consistent with current intentions having priority. The experiments do not indicate whether the observed effect on mean effectiveness is due to the limiting or the direction of attention, or which is the most important factor.

The motivated agent architecture provides an external mechanism by which goals are generated and presented as options for adoption; i.e. the process of selecting between alternative goals is not combined with the process of selecting between possible refinements of adopted goals (or intentions). If the motivation that is associated with a particular goal (the motivation depends on both the importance and temporal relevance of the goal) exceeds the current level of a threshold, the goal is considered for activation. Importantly, this threshold depends on a measure of the load on the agent's computational resources, and so the triggering of goals, the considering of goals, and hence the activation of goals is limited by the computational resources available. A goal is activated only if it triggers and is considered appropriate for active achievement. This serves to limit the computational effort involved in the direction of planning attention, and avoid unnecessary reasoning. The distinctive function of a motivated agent (section 3.1.4) complements and extends rather than contrasts with the function of an IRMA agent. (This is also true for PRS (Georgeff & Lansky, 1987) and other existing BDI agent architectures.)

3.3 The motive-processing architecture

The intention of the motive-processing architecture (figure 3.4) developed by Beaudoin & Sloman (1993) (Beaudoin, 1994) is to investigate processes for the generation and management of goals (sometimes referred to as motivators) and the consequence of such processing on emotion and attention. The primary motivation of this work is to explain human goal-directed behaviour rather than to build effective artificially intelligent systems. However, as the title suggests ("goal processing in autonomous agents"), the author does not consider these to be unrelated goals. Beaudoin (1994) presents a "broad but shallow" specification for a complete agent architecture, parts of which have subsequently been implemented using the SIM_AGENT TOOLKIT (Sloman & Poli, 1996) in the Nursemaid scenario (Beaudoin & Sloman, 1993; Wright et al., to appear). There are three basic components to the motive-processing architecture: goal generactivation, goal management and goal meta-management.

Goal generactivation is a process that monitors the agent's beliefs and "generactivates" goals on the basis of its desires (see figure 3.4). Once a goal is generated it is automatically

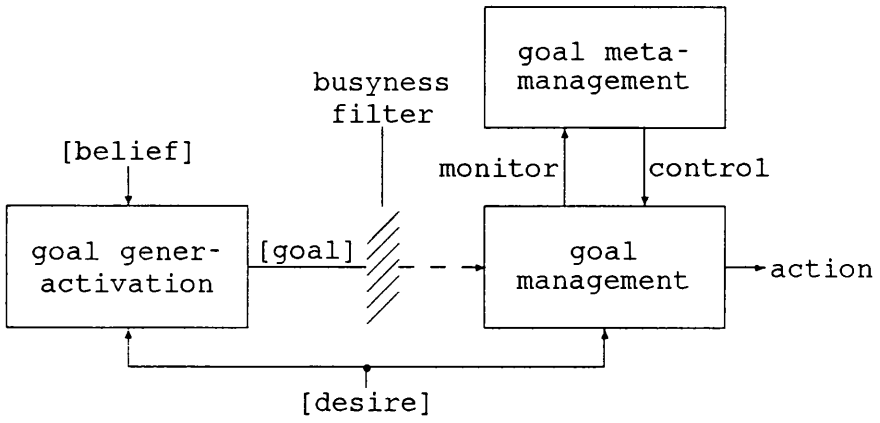


Figure 3.4: A simplified process diagram of the motive-processing architecture.

activated. However, the use of the term "activation" is different from activation in a motivated agent. In MPA, "goal activation is a process that makes the goal control state a candidate for directing management processes" (Beaudoin, 1994, p. 65): i.e. the goal is tested to determine whether or not it will pass through the "busyness filter". If a goal does pass through this filter it is said to have "surfaced". The busyness filter that an activated goal attempts to pass through is similar to the compatibility filter and filter override mechanism described by Bratman et al. (1988). However, the likelihood of a goal passing through the busyness filter depends on the load on the agent's planning processes (i.e. how "busy" the agent is). Therefore, MPA directly addresses the problem of directing and limiting reasoning attention (see section 1.3).

"Goal generactivation refers to the generation of a goal, if it does not already exist, or the activation of that goal if it does." (Beaudoin, 1994, p. 65). This implies that if the goal is not activated, it is not evaluated with respect to the busyness filter. There is however, no indication how or in what way the conditions for goal generation are different from those for activation, and so it can only be assumed that the conditions for generation and activation are identical. The generactivation mechanism in MPA is thus equivalent to the generate alternatives process in IRMA. Hence, a designer of an MPA agent must also consider the issues discussed in section 3.2.2 when determining the conditions for the generactivation of goals. A practical solution again is to use specialised goal generactivators that generate and activate goals to have considered generating goals in response to changes in the perceived state of the external environment. The decision that a goal is required (a possible consequence of satisfying this first goal) will then cause the goal that has just been considered to be generactivated: a goal that will be further processed. (This possibility is not discussed by either Beaudoin & Sloman (1993) or Bratman et al. (1988).)

The essential difference between MPA and IRMA is the nature of their filtering mechanisms. The requirement that is presented for the filtering mechanism in MPA is that the filter depends on the load on the cognitive resources of the agent, whereas the filter in IRMA is based on compatability with current intentions with the possibility of override. However, neither consider the influence of time. The temporal relevance of a goal is clearly important in directing the attention of an agent to relevant goals. If the attention of an agent is directed towards a goal a significant time after its deadline there will often be little point in acting on that goal, but prior to this the agent could have been acting on goals that are not so urgent. If the attention of an agent is directed towards acting on a goal that is not worth achieving for some time (this type of goal is often ignored), the agent will invest effort in scheduling and planning for the achievement of this goal to the detriment of other, possibly more relevant goals. Time is an factor that is central to the motivated agent architecture, and the machinery presented in this thesis is novel in this regard.

3.4 Conclusion

The motivated agent architecture is similar to both the IRMA architecture and MPA, but approaches the problem of limiting the attention of computationally expensive processes in a novel way. Both IRMA and MPA propose a filter-based mechanism for limiting and directing attention, but neither consider in detail the influence of time on when a goal should be considered. Furthermore, the literature does not provide an adequate explanation of how either of these architectures can handle goals that may only be generated through the agent deliberating about a change in the perceived state of the environment. A possible solution has been suggested in this chapter.

Chapter 4

Alarm generation

This chapter introduces alarms and describes the mechanisms by which they are specified and generated. In the simplest terms, an alarm, a, is a structure that associates a goal, g, with an intensity that may change over time; i.e. an implementation of the notion of a motivated goal, section 3.1.2. This intensity represents how appropriate the goal is for consideration at the present time, and hence how likely it is to hold attention; i.e. a motivation, section 3.1.3. Typically, as time passes the intensity of the alarm increases to a maximum. If the intensity of an alarm exceeds some threshold, the goal is considered; i.e. triggering, section 3.1.4. This threshold changes as the situation changes; e.g. if the agent is busy, the threshold may be high and vice versa (see chapter 6). The alarm structure is essentially an indexing scheme for the suspension and subsequent reminding of goals (cf. Birnbaum (1986), Ellis & Nimmo-Smith (1993), Ellis (1994), Patalano et al. (1993), Hammond (1989b), and Simina & Kolodner (1995)). However, the influence of time is of primary interest in this thesis; an issue that has been little considered in the literature. The goal management mechanisms that are discussed in this and the next three chapters use alarms to focus the agent's attention on a limited number of salient goals.

In this chapter, section 4.1 discusses goals and goal generators, and two distinct classes of goal generator are identified: D-Goals (section 4.1.1) and R-Goals (section 4.1.2). Section 4.2 describes the representation of time used in the alarm processing machinery. In sections 4.4 and 4.5, the mechanisms for the generation of goals and their encapsulation in alarm structures are discussed. These alarms, and the goals they encapsulate, are indistinguishable once they are generated, but the mechanisms that produce them are importantly different. Related work is evaluated in section 4.6.1, and section 4.6.2 details the conclusions of this chapter. The notation used in this chapter and in chapters 5, 6 and 7 is summarised in appendix A.

4.1 Goals and goal generators

In general, goals are states of affairs that an agent is motivated to achieve. Thus, the goals of an autonomous agent are the problems that it has set itself to solve through some sort of decision-making capability. However, the term goal is used to denote different concepts in different contexts, often without a clear definition. The following definition specifies what the term goal refers to in this thesis.

Definition 4.1 A goal is a proposition, p, that is to be made true associated with the importance of satisfying the proposition, i_{max}, and the temporal context in which it should be satisfied. This temporal context consists of the time before which there is no point in satisfying p, t_{dt}, an estimate of how long it will take to achieve p, $\Delta_a t$, and the time before which the agent wishes p satisfied, t_{dl}.[1]

$$g = (p, i_{max}, t_{dt}, \Delta_a t, t_{dl})$$

where an importance, i, is modelled as a real number.

$$\forall i \bullet i \in \Re$$

The variable t_{dl} is commonly referred to as the deadline of the goal. However, the fact that a goal has a deadline does not necessarily imply that there is no point in achieving the goal once the deadline has passed. In general, a deadline is a time point before which the agent *wishes* the goal to be satisfied. Consider a warehouse agent that has two goals to pursue: (1) the goal to have prepared an order for a customer who should arrive at around t_{arrive}; and (2) the goal to have placed an order with a manufacturer of widgets so that the warehouse is restocked with widgets. If the first goal is not satisfied before t_{arrive}, it may not be possible for the agent to satisfy the goal; the customer may not be prepared to wait very long. This is an example of a hard deadline. If the second goal is not satisfied before the deadline, it may still be possible for the goal to be satisfied. This softer deadline represents the time by which the agent would prefer the goal to be satisfied, but if not as soon as possible after that time.

A consequence of this definition of a goal (i.e. that a goal has a well-defined temporal context) is that all goals have a limited life: They are generated, planned for, and once they have been satisfied, cannot be satisfied, or are no longer required, they are deleted. This use of the term goal is importantly different from that implied by authors such as Schank & Abelson (1977), Ortony et al. (1988), and Slade (1994). These authors refer to different "types" of goal including goals

[1] The representation of time and the temporal operators used in this thesis are described in section 4.2. A summary of the syntax used in the definitions presented in this chapter and chapters 5 and 6 in given in appendix A.

that persist after they are satisfied (e.g. cyclical satisfaction goals (Schank & Abelson, 1977)). The advantage of defining goals as propositions to be made true within some temporal context is that a clear distinction can be made between the process that generated the goal and the goal itself. For example, "hunger" is commonly referred to as a goal (Ortony et al., 1988, p. 42). In this thesis hunger is not a goal, but a generator of goals, or a "motive" (see chapter 3). The motive of hunger will generate goals to have mitigated hunger that have specific temporal contexts. Once hunger is mitigated, or the agent decides not to satisfy the goal, the goal is deleted, but the motive of hunger remains and may generate other goals with different temporal contexts. Consider an example taken from the warehouse domain test-bed. Suppose that the warehouse agent receives a request for an order to be satisfied from a potential customer. If the agent decides to accept the order, a goal is generated. Then, if the order is satisfied, can no longer be satisfied, or if the agent no longer wishes to satisfy the order, the goal is deleted. (The deletion of a goal for whatever reason may influence other processes, or even lead to the generation of other goals.) This goal is only one of a number of different goals that will be generated at various times in the service of the motive to satisfy customer orders.

In addition to it being important to distinguish between goals and the processes that generate goals, it is important to distinguish between goals (i.e. what to have done) and the actions that could be used to achieve those goals (i.e. what to do). The reason for the frequent confusion between these concepts is that a goal often has a single *primary* satisfying action with certain subsidiary actions required to satisfy its preconditions. For example, the goal to have mitigated hunger is primarily satisfied through eating, but it may also be satisfied by taking a placebo with no nutritional content (e.g. a diet pill), or by using intravenous nutrition, etc. An action is a transformation between states, and the selection of actions is the function of a planner. In this thesis, an action is assumed to consist of a list of preconditions that must hold for the action to be executed and a list of postconditions that describe the changes that the action is known to make in the domain. (The use of a specific action representation is necessary to define certain aspects of the alarm processing machinery with sufficient clarity. A STRIPS-like (Fikes & Nilsson, 1971) action representation is chosen for its simplicity, but other more complex forms may be used if necessary; this choice of representation does not restrict the scope of the alarm processing machinery.)

Definition 4.2 A state is a set of propositions.

$$S = \{p_1, p_2, \ldots, p_n\}$$

Definition 4.3 The domain, D, of an agent is its internal state, S_i, and the state of the external environment that the agent perceives, S_e.

$$D = S_i \cup S_e$$

Definition 4.4 An action, a, has the potential to change the domain. An action consists of a list of propositions that are its preconditions, $pre(a)$, and a list of propositions that are its post-conditions, $post(a)$. In addition to these, when the action is instantiated in a plan the following information is available: (1) an estimate of the time required to perform the action, $duration(a)$; (2) the time before which the action is to be performed, $end(a)$; and (3) the importance that the agent ascribes to the performance of the action, $imp(a)$.

All goals are treated in the same way by the goal management mechanisms once they are generated, but goals can be generated through different processes. Two broad classes of goal generators are considered here: D-Goals and R-Goals. (The terms D-Goal and R-Goal refer to both the goal and the source of that goal.) Once generated, goals from different sources are indistinguishable; there is no difference between a goal to have achieved the proposition p generated through decision (D-Goal) or replenishment (R-Goal). A D-Goal is a process that generates goals through a decision to have something done, and an R-Goal is a process that replenishes goals automatically according to some pre-defined, but potentially adaptable specification. This is not intended as a taxonomy of goal generators, there are many sub-classes that can be specified, for example see Carbonell (1982). However, the distinction of these two types of goal generator is motivated by necessity rather than to classify different types of human goal-directed behaviour (Norman & Long, 1995b; Norman & Long, 1995a).

4.1.1 D-Goals

Ortony et al. (1988) characterise active pursuit goals to be states of affairs that the agent wishes to achieve under certain conditions. A prerequisite to the generation of a goal to achieve some state in response to an event detected in the agent's environment such as being asked to attend a meeting is a decision based on the agent's beliefs (Castelfranchi, 1995). For this reason, the term D-Goal is used to refer to a goal that has been generated because the agent has deliberated about the present state of the domain and *decided* to generate a goal in response. Consider an agent that is asked to attend a meeting at some specified time. In response to this change of belief (i.e. the agent now believes that some other agent has requested that it attend a meeting), the agent may consider this request,[2] and if the adoption of a goal to have attended the meeting is consistent

[2]Whether or not such a state change causes the agent to consider the request at any time may be subject to filtering as discussed in section 3.2.2. This is not discussed in any detail as existing work (such as the goal generactivators

with the agent's beliefs, it will generate that goal.

Given that an agent has decided to generate a goal (i.e. decided *whether* to have something done), it must then decide *when* to have it done. The time at which the agent recognises that something must be done (i.e. the time that a goal is generated through decision) is not necessarily the time at which its attention should be directed towards doing it (e.g. the meeting may be next week). It is common for goals to be generated that are not appropriate for some time; the agent may wish to act on a goal that is generated now, days or even weeks later.

4.1.2 R-Goals

Some goals tend to recur periodically, or at particular times of the day or week, for example. (Goals that recur in this way have been referred to as satisfaction goals (Schank & Abelson, 1977) and replenishment goals (Ortony et al., 1988).) The process that causes the cyclical generation of the same goal in this way is replenishment, hence the term R-Goal. Replenishment is an autonomic process, so it does not involve reasoning. The use of the term replenishment indicates a similarity between this class of goal generator and the replenishment goal type identified by Ortony et al. (1988). However, an R-Goal process generates goals cyclically (see section 4.5 for a description of the replenishment mechanism), where Ortony et al. (1988) view replenishment goals as semi-permanent goals that simply change in intensity (see section 4.6.1).

The existence of this type of goal generator is both advantageous (the reasoning involved in generating goals through decision consumes computational resources, R-Goals avoid this cost) and necessary (D-Goals are insufficient for some types of interaction with other agents or processes).

1. It is advantageous for agents to generate goals through replenishment rather than through decision. Replenishment requires negligible resources because no decision is required for the goal to be generated. Consider an agent that is required to manage a warehouse. This agent may have certain customers that regularly request the same order. If this customer is reliable, the warehouse agent may agree to prepare orders for that customer at regular times without requiring requests for every order; the warehouse agent assumes that the customer will collect the orders at the agreed times. Therefore, there is no need for the customer to explicitly request every order, and no need for the agent to respond to each request by deciding whether or not to accept the order. The agent need only remind itself to prepare the regular order at the agreed times given that the customer remains reliable. The generation of cyclical goals through replenishment avoids unnecessary reasoning.

---

and busyness filter of Beaudoin & Sloman (1993)) addresses this.

2. For an agent to perform certain types of goal directed behaviour, it is necessary for it to generate goals cyclically rather than simply responding to detected events. An agent in a real-world domain is not the sole actor in that domain. For the agent to affect the domain in useful ways, it may be necessary for it to synchronise its activities with aspects of the environment that are not under its control. For example, in the warehouse domain customers communicate with the agent through a device that is situated in the office referred to as the order book. The warehouse agent can only sense the contents of the order book if it is in the office. There is no event to which the agent can respond, and so the agent must motivate itself to check the order book by generating goals. The agent will generate goals to have mitigated its curiosity about the state of the order book periodically through some autonomic replenishment process.

4.2 Temporal representation

The representation of and reasoning with time plays a central role in the processing of alarms. It is important for the agent to be reminded of suspended goals at appropriate times; for example, in time for the agent to achieve the goal before its deadline. However, the agent will not necessarily be completely certain about how long it will take to achieve a particular goal, or when that goal should be satisfied, for example. So, the agent must represent and reason appropriately with uncertain knowledge about time points and intervals. (Note, the notation used in this chapter is summarised in appendix A.)

4.2.1 Representing time points and intervals

In practice, there is no single correct representation of time. The representation and inference mechanism used must fit the requirements of the system (Long, 1989). Consider a warehouse agent that has agreed to satisfy an order requested by a particular customer. From the agent's experience, it normally takes about twenty minutes to prepare an order of a similar size, assuming there is sufficient stock in the warehouse. However this is only an estimate; in practice the time required to prepare the order may vary. Suppose that a customer will only wait five minutes in the loading bay and will leave if the order is not ready in that time. If the agent assumes that it will take twenty minutes to prepare the order, it actually takes closer to half an hour, and the customer arrives on time, then the agent will not have the order prepared on time. The representation of time must therefore account for uncertainty in an agent's predictions of when particular events will occur or how long particular tasks take to perform. An agent must be able to represent knowledge such as "it will take twenty minutes to prepare the order give or take ten minutes" so that it may be used in reasoning with sufficient confidence.

A second issue in selecting a representation of time is the granularity of time. Some goals take far longer to pursue than others; for example, an agent may have one goal to have composed an Email message and another to have written a thesis. An acceptable estimate for the length of time required to compose an Email message is about five to ten minutes, but a thesis will take months to write. It is neither appropriate to represent the time required to write a thesis in minutes nor to represent the time required to compose an Email in months. Therefore, an appropriate representation should be able to express time points and intervals at various levels of granularity.

A time interval, Δt is represented by a tuple (ε, λ); where ε represents the estimated length of the time interval and λ is a measure of the accuracy of that estimate. Suppose that the warehouse agent predicts that it will take $\Delta_{order}t = (\varepsilon_{order}, \lambda_{order})$ to prepare an order for a customer. This means that, on the basis of the agent's experience, it will take between $(subtract(\varepsilon_{order}, \lambda_{order}))$ and $(add(\varepsilon_{order}, \lambda_{order}))$ to complete the order. Therefore, the agent has, on the basis of experience, an estimate that the duration of some future interval of time will be less than or equal to the duration that is represented by $add(\varepsilon, \lambda)$. Suppose an agent predicts that it will take twenty minutes to prepare an order give or take ten minutes. This knowledge can be represented as $\Delta_{order}t = (\text{twenty minutes}, \text{ten minutes})$, and the agent can therefore predict that it will take less than or equal to thirty minutes to prepare the order.

A time point is represented as some interval after the fixed point t_0. So, the point in time t is represented by a tuple (ε, λ); where ε represents the estimated period after t_0 and λ is a measure of the accuracy of that estimate.

To represent time points and intervals at varying levels of abstraction, ε is expressed as an n-tuple $(e_1, e_2, \ldots, e_n)$ where the element e_j represents the number of units of time at the jth level of granularity (and equivalently λ will be represented as an n-tuple $(l_1, l_2, \ldots, l_n)$). For example, if ε is chosen to be represented as a number of hours, a number of tens of minutes, a number of minutes and a number of seconds, then ε will be a 4-tuple (e_h, e_{tm}, e_m, e_s). Then, the "two hours and twenty one minutes" may be represented as the 4-tuple $(2, 2, 1, 0)$. In general, a time point after the fixed point t_0 or an interval of time may be represented as a tuple $(\varepsilon, \lambda) = ((e_h, e_{tm}, e_m, e_s), (l_h, l_{tm}, l_m, l_s))$. For example, if the agent estimates that it will take twenty minutes to prepare an order, give or take ten minutes, $\Delta_{order}t = (\varepsilon_{order}, \lambda_{order}) = ((0, 2, 0, 0), (0, 1, 0, 0))$.

4.2.2 Reasoning with time points and intervals

Consider estimates of when a customer will arrive, and how long it will take for the agent to prepare the order required by that customer. When should the agent start preparing the order

to get it done on time? This agent requires mechanisms for making valid inferences with its temporal knowledge to answer such a question. The inference machinery used in the prototype agent performs addition, subtraction, and a number of other combination functions as well as a comparison function. In this section the combination functions ($\dot{+}$) and ($\dot{-}$), and the comparison function *before* are defined; these serve to illustrate the principles involved in specifying such mechanisms.

When adding ($\dot{+}$) or subtracting ($\dot{-}$) time points and intervals it is important to maintain the accuracy of the agent's estimates. Suppose that the agent that intends to prepare a order for a customer in time for their arrival predicts that the customer will arrive three hours after t_0 give or take five minutes, $t_{arrive} = ((3,0,0,0),(0,0,5,0))$, and that it will take twenty minutes to prepare the order give or take ten minutes, $\Delta_{order}t = ((0,2,0,0),(0,1,0,0))$. Consider the worst case: the customer actually arrives five minutes early and it takes thirty minutes to prepare the order. The agent should start preparing the order by two hours and twenty five minutes after t_0; i.e. at the time point $(2,2,5,0)$. Consider the average case: the customer arrives on time and it takes twenty minutes to prepare the order. The agent should start preparing the order by two hours and forty minutes after t_0; i.e. at the time point $(2,4,0,0)$. To maintain the accuracy of the agent's predictions: $subtract(\varepsilon_{prepare}, \lambda_{prepare}) = (2,2,5,0)$, and $\varepsilon_{prepare} = (2,4,0,0)$. Therefore the agent can predict that it should start preparing the order two hours and forty minutes after t_0 give or take a quarter of an hour; $t_{prepare} = ((2,4,0,0),(0,1,5,0))$.

Definition 4.5 The addition or subtraction of time points and intervals is performed by adding or subtracting ε_1 and ε_2, and by adding λ_1 to λ_2 to maintain confidence. Note that this definition will lead the agent to act in a conservative manner with respect to its estimates of future time points and intervals. An alternative is for the accuracy of the result to be the maximum of λ_1 and λ_2.

$$(\varepsilon_1, \lambda_1) \oplus (\varepsilon_2, \lambda_2) = (add(\varepsilon_1, \varepsilon_2), add(\lambda_1, \lambda_2))$$

$$(\varepsilon_1, \lambda_1) \ominus (\varepsilon_2, \lambda_2) = (subtract(\varepsilon_1, \varepsilon_2), add(\lambda_1, \lambda_2))$$

where the functions add and $subtract$ follow their intuitive interpretations.

However, it is meaningless to represent negative intervals of time, or time points before t_0. Therefore, there are a limited number of legal uses of the operators $\oplus$ and $\ominus$. The result of either $t_j \dot{+} t_k$, $\Delta_j t \dot{+} t_k$ or $\Delta_j t \dot{-} t_k$ will be meaningless. The result of $t_j \ominus t_k$ will be meaningful if the interval $\Delta_k t$ is less than $t_j \ominus t_0$, $\Delta_j t \ominus \Delta_k t$ will be meaningful if the interval $\Delta_k t$ is less than $\Delta_j t$, and $t_j \ominus t_k$ will be meaningful if t_k is before t_j. The result of performing $t_j \oplus \Delta_k t$ or $\Delta_j t \dot{+} \Delta_k t$ will be meaningful in all cases.

In reasoning about time points and intervals it is important to define what it means for one time point to occur before another, and for one interval of time to be less than another. (Note that due to the choice of representation, if two intervals of time, $\Delta_1 t$ and $\Delta_2 t$, satisfy the relation $before(\Delta_1 t, \Delta_2 t)$, then the duration of $\Delta_1 t$ is less that of $\Delta_2 t$.)

Definition 4.6 The relation $before(t_1, t_2)$ is interpreted as true if and only if the agent is sufficiently confident that t_1 is before t_2; i.e. that there is no overlap between the confidence intervals of these time points. Again the definition leads to conservative behaviour.

$$before((\varepsilon_1, \lambda_1), (\varepsilon_2, \lambda_2)) = lessthan(add(\varepsilon_1, \lambda_1), subtract(\varepsilon_2, \lambda_2))$$

where the function $lessthan$ follows its intuitive interpretation.

Therefore, if an agent predicts that one event will occur at the time $t_1 = (\varepsilon_1, \lambda_1)$ and another event will occur at the time $t_2 = (\varepsilon_2, \lambda_2)$, then it may infer that t_1 will occur before t_2 if the relation $before(t_1, t_2)$ holds. (Refer to appendix B.2.1 for the Miranda^{TM} types and functions that implement this method of representing and reasoning about time.)

4.3 The alarm function

The intention behind these goal management mechanisms is to avoid reasoning about goals that are not relevant to the agent at present. For example, suppose the warehouse agent has generated a goal to have prepared an order for collection on Friday at 10am, and today is Monday. In normal circumstances, there is little point in the agent considering acting on that goal until Thursday evening or Friday morning. Ideally, the agent should be reminded of the goal at the time it is most appropriate for the goal to be acted on. At the latest, the agent should be reminded of the goal in enough time for it to be satisfied before some deadline. In addition to this requirement, the processes by which goals are brought to the attention of the agent must be heuristic. Finally, there are two further requirements for the alarm function: (1) the structure of the alarm function should be uniform for all goals generated through all processes; this will aid analysis of the behaviour of the goal management mechanisms; and (2) the alarm function and all other influences on the intensity of an alarm (see the following two chapters) must be simple; this will aid both the formalisation of the goal management processes and their analysis.

An alarm function is a time varying function of intensity; the intensity at a particular time being a measure of how appropriate the goal encapsulated in that alarm is to the agent. However, not every possible function is appropriate. The intensity of the alarm function must be zero before the delay time of the goal, t_{dt}. (The delay time is the time before which the agent predicts

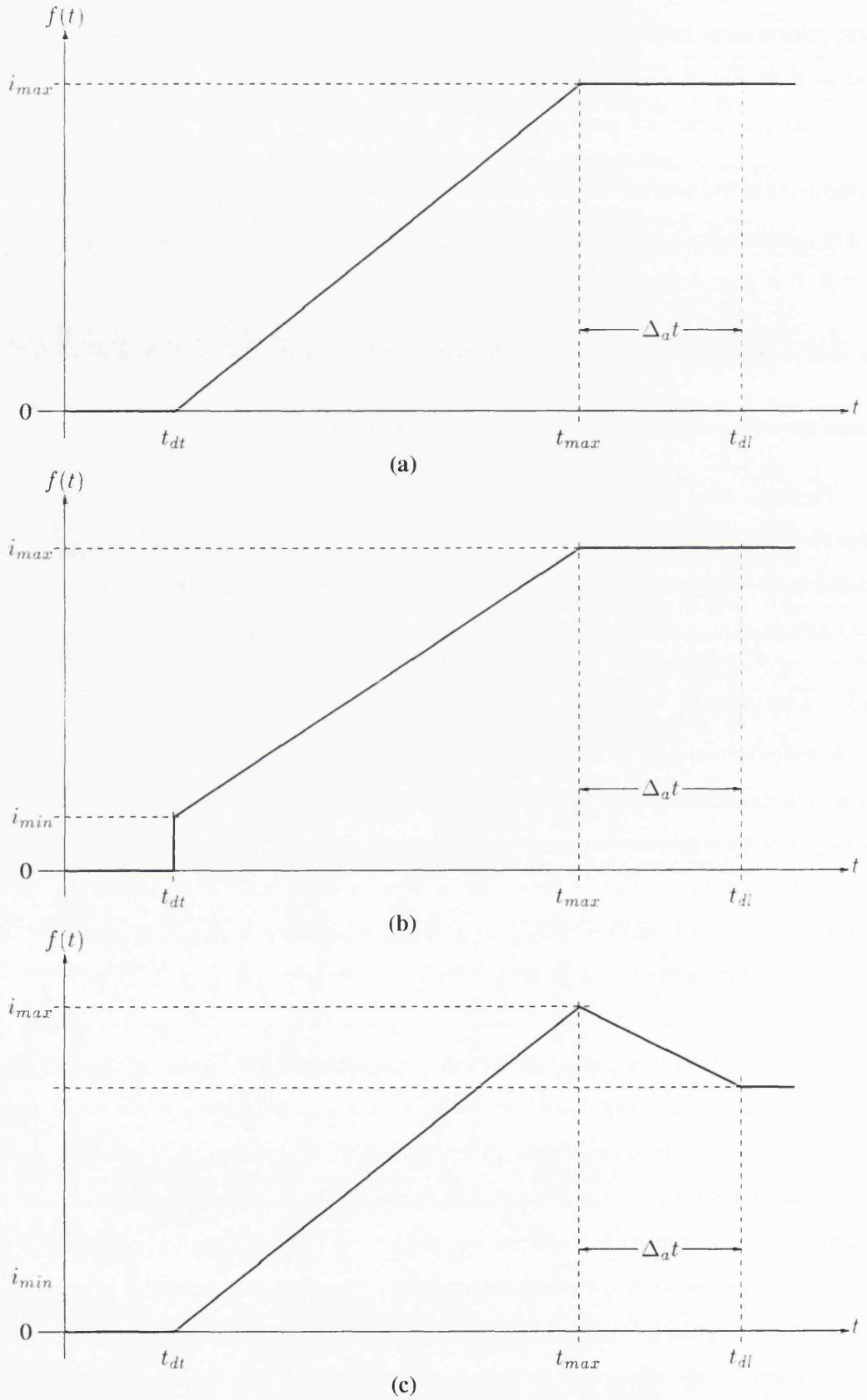


Figure 4.1: Exemplar alarm functions.

that there is no point in acting on the goal.) The intensity of the alarm function must be maximum at the time $t_{max} = t_{dl} \ominus \Delta_a t$. (This is the last point in time that the agent predicts it can achieve the goal before the deadline; the deadline being t_{dl} and the estimated time required to achieve the goal being $\Delta_a t$.) These characteristics leave many possible alternatives. A few are discussed below.

- The simplest function is that illustrated in figure 4.1(a). The intensity of an alarm remains at zero before t_{dt}, it increases linearly to the maximum, i_{max}, at t_{max}, and remains at i_{max}. (Note that i_{max} is the importance of the goal.) This is the minimal alarm function that meets the characteristics described and is the one assumed throughout this thesis, and in the simulation described in appendix C.

- The transition between an intensity of zero at t_{dt} and an intensity of i_{max} at t_{max} may be changed through the use of a curve with certain characteristics, or by adding a minimum intensity after the delay time as in figure 4.1(b). The intensity i_{min}, is the minimal intensity of the alarm after the delay time, and between t_{dt} and t_{max} the intensity linearly increases from i_{min} to i_{max}. The addition of this minimum intensity may capture situations in which action to achieve a goal is maximally relevant immediately after the delay time; i.e. when $i_{min} = i_{max}$, the alarm function becomes a step function.

- The intensity of the alarm may drop after t_{max}. For example, in figure 4.1(c) the intensity of the alarm function drops to i_{min} at t_{dl}. This may capture situations in which considering the goal as a candidate for action after t_{max} is less relevant than at t_{max}. However, it should be noted that the intensity of an alarm is not a measure of the utility of achieving the goal, but a coarse measure of how relevant it is for the agent to *consider* acting on the goal. The values that the alarm function is based on are estimates of deadlines, estimates of how long it will take to achieve a goal, etc. All such predictions are fallible. Therefore, it may be difficult to justify reducing the intensity of an alarm function before t_{dl}.

- The intensity of the alarm may vary cyclically. Consider an agent that wishes to know the weather forecast for tomorrow. Suppose that the agent knows that a weather forecast is given on the radio at around five minutes to every hour. The agent may predict that it will be relevant for it to consider this goal at around this time every hour, and so the alarm function may be defined to vary cyclically. However, this is not necessary. Suppose that an alarm encapsulating a goal to have mitigated its curiosity about the weather forecast, is set so that the intensity reaches maximum at ten minutes to ten o'clock, but the alarm does not trigger until ten o'clock (alarm triggering is discussed in detail in chapter 6). At this

point, the agent realises that the goal cannot be satisfied (note *this* goal with *this* temporal context). The goal is deleted, but the agent still wishes to know the weather forecast, and so a new goal is generated with a new temporal context and an appropriate alarm is set. The intensity of this alarm then increases to maximum at around ten minutes to eleven o'clock. It is feasible to use cyclic alarm functions for goals with similar characteristics, but this compromises the requirement of uniformity (see above). It is not necessary to allow alarms to vary cyclically, and so the uniformity of alarm function for all goals may be maintained.

- The intensity of the alarm may vary non-linearly. For example, instead of the intensity linearly increasing from zero to i_{max} as in figure 4.1(a), this straight line can be replaced by a sinusoid. Similarly, the other two functions illustrated may be given non-linear characteristics. However, it is not clear whether such an additional complexity can be justified; the shape of the curve will depend of additional variables.

The alarm function, $f(t)$, used here is generated on the basis of the temporal relevance of that goal and its importance; i.e. the variables t_{dt}, $\Delta_a t$, t_{dl}, and i_{max} (see figure 4.1(a)). (Note, the notation used in this chapter is summarised in appendix A.)

Definition 4.7 The alarm function of an alarm a is zero before t_{dt}, increases from zero to i_{max} from t_{dt} to t_{max}, and then remains at i_{max}.

$$f(t) = \begin{cases} 0 & \text{if } before(t, t_{dt}) \\ i_{max} & \text{if } \neg before(t, t_{max}) \\ i_{max} \times \left(\frac{t \ominus t_{dt}}{t_{max} \ominus t_{dt}}\right) & \textbf{otherwise} \end{cases}$$

where $t_{max} = t_{dl} \ominus \Delta_a t$ and the ratio $\frac{t \ominus t_{dt}}{t_{max} \ominus t_{dt}}$ is expressed as a real number.[3]

The deadline (t_{dl}) is the time at which the agent wishes the goal to have been satisfied. $\Delta_a t$ is the period of time that the agent expects will be required to act to satisfy the goal. With this value and the deadline, an estimate can be made of the last point at which the agent should attend to the goal to have it satisfied in time, and hence the time at which the intensity of the alarm should be maximal: $t_{max} = t_{dl} \ominus \Delta_a t$. However, it is not always possible to accurately predict how long it will take to satisfy a goal without having decided how it should be satisfied;

[3]This ratio of two periods of time, $\Delta_1 t / \Delta_2 t$, is found by determining the number of basic units of time that $add(\varepsilon_1, \lambda_1)$ and $add(\varepsilon_2, \lambda_2)$ represent. For example, if $add(\varepsilon_1, \lambda_1)$ consists of 3 seconds, 5 minutes, 2 tens of minutes and 1 hour, the number of basic units is 5103 seconds. The ratio of the two periods of time is taken to be the ratio of these two integers at an appropriate level of accuracy.

i.e. before planning. These predictions can only be based on the agent's experience or other available evidence, and the accuracy of the prediction taken into account when it is used (see section 4.2). If the agent underestimates $\Delta_a t$, it may delay too long before acting, and a goal that could have been satisfied in time will fail to be satisfied. For this reason, $\Delta_a t$ is determined both through analysing the agent's experience in achieving similar goals, and by deciding the amount of time that it is reasonable to invest in the satisfaction of the goal; i.e. a time limit is *allocated* to each goal for action.

The delay time (t_{dt}) is the time point, before which the agent predicts that it is not appropriate for it to act on the goal. For instance, the warehouse agent may have the goal to have prepared an order which contains frozen commodities for a customer. If the agent takes the frozen commodities out of the freezer and into the loading bay too long before the customer is expected to arrive, the commodities will defrost and be spoilt. There is no point in the warehouse agent preparing an order containing perishable commodities if the customer is expected to arrive after the commodities will have perished. Before t_{dt}, the alarm function remains at zero. The times t_{dt} and t_{max} define a time window where the agent predicts that it is sensible to consider acting on the goal.

Different alarms may have the potential to influence the goal management processes to varying degrees; this potential is the maximum intensity of the alarm, i_{max}. The more important a goal is, the greater the maximum intensity of an alarm encapsulating that goal; i_{max} is the importance of a goal (see definition 4.1). So, if an important goal is highly relevant, the intensity of the associated alarm is greater than if the goal was less important to the agent. In this way, goals of greater importance take precedence. It is assumed in this thesis that the agent is able to assign real numbers to the importance of a goal, i_{max}. The primary focus of this work is on determining the temporal relevance of a goal, rather than its importance. However, it is clear from the work of Sloman (1987), Beaudoin (1994) and Slade (1994) among others that the importance of a goal is not necessarily most easily modelled in terms of real numbers. There may exist only a partial ordering between goals, or the importance of a particular goal may depend on the consequences of not having it satisfied. However, the impact of this on the core ideas presented in this thesis is restricted to implementation; the foundations remain intact. Regardless of the model used to express the importance of goals, an approximate numeric value of importance may be generated. Factors such as the deadline of a goal, how long it will take to achieve as well as a numeric value of importance may not be known precisely. The alarm function is however designed to provide a *coarse heuristic* for determining the relevance of a goal for consideration, and thus the representation of the importance of a goal as a real number is justified.

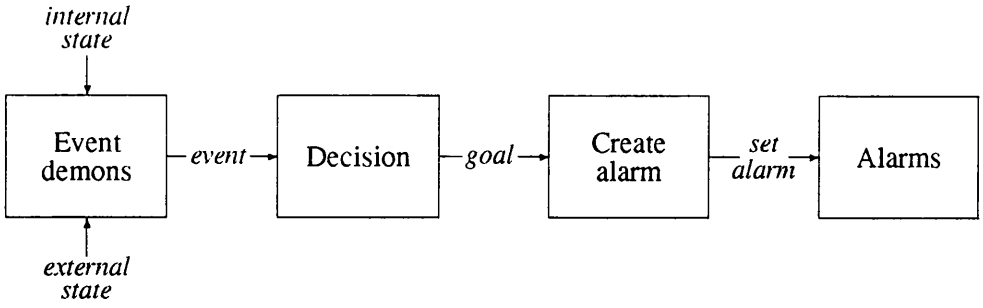


Figure 4.2: Alarm generation through decision.[2]

4.4 Alarm generation through decision

In the system presented here, simple recognition mechanisms, similar to "Noticers" (Wilensky, 1983), are used to trigger the agent to *consider* generating a goal through decision. These mechanisms (demons) have no inferential power, they simply monitor the external environment and internal state of the agent, reporting to appropriate mechanisms when some list of conditions hold. Event demons (figure 4.2) are dedicated to recognising events that may warrant the generation of a goal.

Definition 4.8 A demon, d, is a function characterised by a set of propositions. If all of these propositions hold in the domain, D, then $d(D)$ is true.

$$d(D) = \{p_1, p_2, \ldots, p_n\} \subseteq D$$

If an event demon evaluates to true, the event is reported, and evaluated with reference to the agent's beliefs. The agent will then decide whether or not a goal is required. For example, the warehouse agent has a demon that will respond to new orders requested by customers that are detected in the order book. When the agent receives a request, both the order and the customer are evaluated. The order may be accepted if the customer is not known to be unreliable, and if the order is profitable for example. The results of these deliberations are recorded as the reasons for the generation of this goal for future reference; see section 6.2.2.

When the agent has decided that an event (detected by a demon) warrants the creation of a goal, the temporal context and importance of that goal, must be determined. For example, if a customer requests an order, and this order is acceptable to the agent, then a goal to have met the order is created. At the time of generation, the agent must estimate (through some predictive

[4]This figure is the first in a series of related figures that illustrate the various functions of the alarm processing machinery. The final process diagram is figure 6.3, and intermediate process diagrams are provided as the main functions of the machinery are introduced: figures 4.3, 5.1, 5.4 and 5.9.

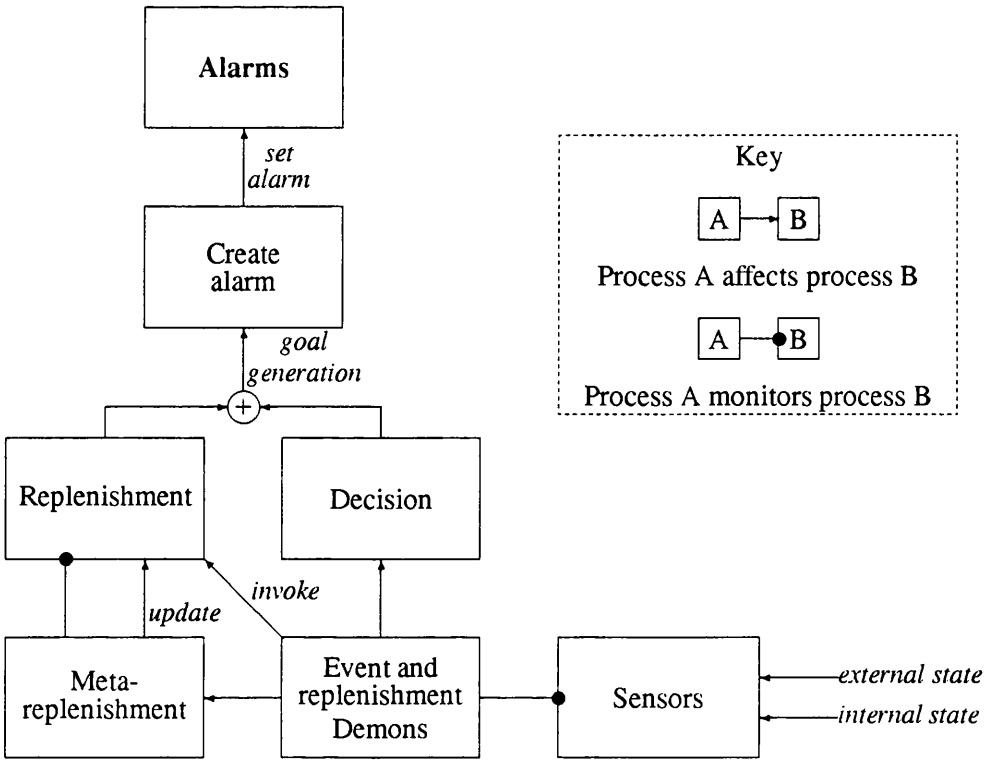


Figure 4.3: Alarm generation.

mechanism) when this goal needs to be achieved by (i.e. the deadline), how long it will take to achieve, any time before which there is no point in acting on the goal, the importance of the goal, etc. With this information, an alarm function may be defined and an alarm set.

4.5 Alarm replenishment

The visible effect of an autonomic replenishment process is the generation of a stream of goals at appropriate times. These goals are treated no different from goals generated through decision either by the alarm processing machinery, or by the planning and control processes.

4.5.1 The mechanism of replenishment

Replenishment is a mechanism by which goals are generated automatically on the basis of a simple rule. The replenishment process is responsible for the generation of goals that recur cyclically or according to some timetable. For example, an agent may restock the warehouse it is responsible for managing by ordering new stock from a manufacturer. However, as the agent satisfies orders requested by its customers or as the stock passes the sell-by date, the level of stock in the warehouse diminishes. Therefore, at some appropriate time in the future the agent must again restock the warehouse. Goals generated in the service of the motive to maintain stock lev-

els will tend to recur periodically. For a particular commodity (e.g. widgets), the period at which goals to restock the warehouse with this commodity recur will depend on variables such as the rate at which widgets are sold to customers and their shelf-life. If these variables are predictable to some degree, it is prudent for the agent to restock with widgets regularly at some appropriately controlled rate. A replenishment process may be defined for this purpose. (Refer back to section 4.1.2 for an explanation of why this type of goal generation mechanism is both advantageous and necessary.) Suppose that the warehouse agent decides that: (1) it should restock the warehouse with widgets every week; (2) there is no point in restocking two days after the last time the warehouse was restocked; (3) it takes about a day for new stock to be delivered; and (4) the importance of having a sufficient stock of widgets is i_{widget}. Then, a replenishment process can be defined on the basis of a specification such that, when invoked at t_{now}, a goal will be generated to have restocked the warehouse with widgets that has a delay time two days from t_{now}, has a deadline a week from t_{now}, $\Delta_a t$ is one day, and i_{max} is i_{widget}. With this information, an alarm function may be defined and an alarm set, which will that direct the agent's attention to this replenished goal at an appropriate time about a week later.[5] Depending on the time that the replenishment process is invoked, a goal to restock the warehouse with widgets will recur about a week after that time.

An important class of replenishment goals are those that motivate the agent to seeking information about the state of its world. For example, the warehouse agent is motivated to periodically check the order book to determine whether or not a new order has arrived or an existing one cancelled (see section 2.2.2 and appendix C). This motive of curiosity generates goals to have mitigated the agent's curiosity about the state of the order book (as well as other goals). The alarm encapsulating this goal may increase in intensity as the time since the agent last checked the order book increases until it reaches i_{max}. This goal may be satisfied simply by the agent seeking information about the state of a particular aspect of its environment; information that may lead to the generation of other goals. For example, after checking the order book, the agent may find that a number of orders have arrived since the last time it was checked. This information may lead the agent to decide to generate a number of D-Goals.

Replenishment processes monitor the state of the domain, and when the last goal generated through this process is satisfied, or has been deleted for some other reason, a new goal is replenished. In this way the agent is influenced in some way by a goal generated on the basis of this goal specification for as long as the replenishment process exists (see figure 4.3). For example,

[5]The replenishment rule will also contain information about potential opportunities and dangers for this goal: this is discussed in the following chapter.

as soon as the agent has satisfied the goal to have restocked the warehouse with widgets, a new goal is generated to restock the warehouse a week from now. To detect the deletion of a specific goal, replenishment processes employ demons that trigger the replenishment of a new goal when there is no similar goal present.

The alarm function of a replenished alarm will increase to maximum over some period of time in the same way as an alarm encapsulating a goal generated through decision (see figure 4.1). The behaviour of goals generated through replenishment in this way is consistent with the observations of Schank & Abelson (1977, pp. 112–113) about "satisfaction goals", Ortony et al. (1988, pp. 39–44) about "replenishment goals", and Slade (1994, pp. 50-51) about both "satisfaction goals" and "preservation goals". These authors all describe an important type of goal that increases in insistence (or intensity) as the time since they were last satisfied increases. As mentioned in section 4.1.2, a replenishment process generates goals cyclically. In contrast Schank & Abelson (1977), Ortony et al. (1988) and Slade (1994) view such goals as semi-permanent with a varying insistence level. This difference enables goals generated though both decision and replenishment to be treated identically by the agent's planning and reasoning machinery; they have the same type. However, the specification of a fixed period of replenishment is not sufficient to fully capture the behaviour required. Consider an agent that has a contract to provide a customer with an identical order every week, and the customer will arrive to collect the order every Friday at 10am. Then, on the basis of this contract, the agent generates a replenishment process that sets alarms to direct the agent's attention towards a goal to prepare the weekly order about a week after the previous order has been satisfied. Suppose that, due to pressure of work, the agent is late with the order one week. The customer is forced to wait for the order to be prepared, but accepts the order and the goal is satisfied at 10.30am. Now, a new goal will be replenished with a deadline a week from now; i.e. at 10.30am rather than 10am next Friday. The agent has agreed to satisfy a regular order, but the time at which the next order must be completed does not depend on when the current order is completed. These regular orders must be prepared according to a timetable rather than some fixed period after the previously replenished goal is satisfied. If the contract to provide the customer with an order every Friday at 10am is encoded as a timetabled R-Goal, and the replenishment process is invoked at 10.30am on a Friday, the deadline of the replenished goal will be next Friday at 10am.

The rule which defines the characteristics of a replenishment process can either be defined to generate goals a fixed period after the last time a goal generated by the same process was satisfied, or according to a timetable of deadlines. These two types of R-Goal process capture all the required characteristics of the behaviour of automatically generated goals.

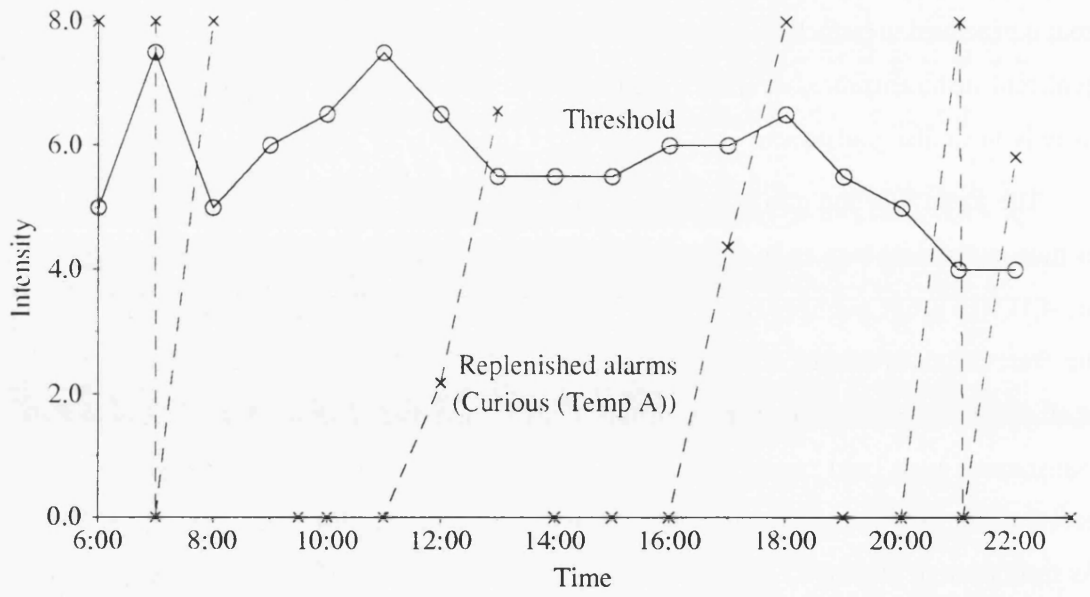


Figure 4.4: The cyclical replenishment of goals.

Example from simulation

Figure 4.4 (produced using data from the simulation of the alarm processing machinery, see appendix C) illustrates the behaviour of instances of the goal to have mitigated curiosity about the temperature of room A in the warehouse domain (see section 2.2). Instances of this goal type are generated through a cyclical replenishment process governed by a template that causes a new goal to be replenished so that its deadline is four hours after the time it was last satisfied, and its delay time two hours before that. The estimated travel time for this goal is 10 minutes, and it has an importance in each instantiation of 8. The data presented shows that at 6am the intensity of the alarm is at maximum, the alarm triggers and the goal is activated. At 7am the goal has been achieved and a new goal replenished. However, at this time (see section C.4) the agent's prior time commitments cause the alarm to be time shifted (an effect described in section 5.3). This means that the alarm triggers (i.e. exceeds the threshold, which at this time is at 7.5), but is mitigated (mitigation is described in section 6.2.3). The intensity of the alarm again rises above the threshold, triggers and the goal is activated. The goal is deleted and again a new goal to have mitigated curiosity about the temperature of room A is replenished. The intensity of this new alarm remains at zero until the delay time (i.e. around 11am) and then increases until it triggers at 1pm. This goal is achieved and again a new alarm generated which increases from zero at 4pm until it triggers at 6pm. If the effects of opportunities, dangers, time commitments and mitigation, all of which are discussed in the following two chapters, are ignored, the clear cyclical behaviour can be seen from this simulation data.

4.5.2 Meta-replenishment

Meta-replenishment, or the manipulation of replenishment processes, serve to generate, delete and modify the processes by which goals are replenished (see figure 4.3). For instance, if the warehouse agent agrees with a customer to supply regular orders, the meta-replenishment processes will generate an R-Goal to generate goals to meet these orders at the appropriate times. The warehouse agent is under some contractual obligation to that customer to have prepared orders at times specified in the contract. Typically, such a contract will also bind the customer to certain activities; for example, the customer may be required to collect and pay for the orders at specific times. If these contract conditions are broken by the customer, then that replenishment process will be deleted. Suppose that the reasons for the existence of an R-Goal process to prepare regular orders for some customer are: (1) the customer requires the regular order; (2) the customer is not known to be unreliable; and (3) the customer pays for the orders less than 24 hours after the order is collected. If the meta-replenishment processes detect that these conditions no longer hold in the domain, the associated R-Goal is deleted. For example, if the customer fails to pay for an order, the replenishment goal is deleted, and a new contract must be negotiated. So, the meta-replenishment processes are responsible for the generation and deletion of such temporary replenishment processes.

The above example describes a replenishment process that persists as long as certain conditions hold in the domain. For example, an agent that owns a car may automatically generate goals to have filled the car with petrol, but this R-Goal may be conditional on the car remaining in the agent's possession. The meta-replenishment processes are responsible for the detection of conditions for the deletion of replenishment processes, and deleting them where appropriate. However, for an agent to survive in its environmental niche, there may be some R-Goals that are essential, and hence unconditional. For example, if the warehouse agent is to survive in its niche, it must maintain the stock in the warehouse and have sufficient battery charge to act on its goals. These replenishment processes are analogous to physiological drives in humans such as hunger and thirst (Carbonell, 1982). There are no conditions under which these R-Goal processes are deleted.

This is not the whole story. Consider the unconditional replenishment process that influences the behaviour of the warehouse agent concerned with the maintenance of the level of widgets in the warehouse. The period of replenishment of goals to have restocked the warehouse with widgets depends on the rate at which widgets are sold, among other variables. If widgets become more popular, the rate at which the agent restocks with widgets should increase accordingly. It is the task of the meta-replenishment processes to adapt the characteristics of an agent's

replenishment rules as the situation changes. Suppose that the agent finds that widgets are being wasted, due to them passing their sell-by date, or that widgets are being delivered faster than being sold. In such a situation, the replenishment period of restocking with this commodity is too short. Suppose that the agent regularly generates goals to restock with widgets in response to the event of having insufficient stock for an order (i.e. a goal to restock the warehouse generated through decision), then the period of replenishment may be too large. Through monitoring the success or failure of an autonomic replenishment process, the characteristics of that process may be adapted as situations change. In this way the agent may attempt to respond to changing circumstances.

Such statistical analysis may be effective for the adaptation of periodic R-Goal processes. However, if the replenishment characteristics of a timetabled replenishment process are out of synchronisation with the environment, it must be reevaluated. For example, suppose that an agent has a timetabled replenishment process that generates R-Goals to have bought a pint of milk every Thursday, and the agent regularly runs out of milk on Monday or Tuesday. When this replenishment process is reevaluated, the agent may either adapt it so that R-Goals recur every Tuesday and Thursday, or it may replace it with a periodic R-Goal with a replenishment period of around three days. This adaptation and reevaluation of automatic goal generation processes require further investigation.

4.6 Discussion

4.6.1 Related work

Ortony et al. (1988, p. 42) distinguish "achievable goals that are not abandoned when achieved" from other goals that are actively pursued. These goals comprise the categories satisfaction goals and preservation goals in the taxonomy proposed by Schank & Abelson (1977) and Slade (1994). The effect of this type of goal, which can be observed, is that of cyclical goal-directed behaviour, but this does not necessarily imply that the goal is treated differently by the agent or is of a different type. This chapter has shown that the same effect can be produced by distinguishing the processes by which goals are generated rather than the goals themselves. Consider the influence on an agent's activities of collecting a regular pay cheque (an example of a replenishment goal used by Ortony et al. (1988)). In normal circumstances it is sufficient to describe the behaviour of the agent driven by this influence as a single goal that "become[s] more insistent as the time [since the] last realisation increases" (Ortony et al., 1988, p. 42). However, this is not always the case. The agent may work extra hours which is paid at an earlier time than the normal pay cheque. In this situation, the agent must have the ability to generate a goal to have collected this extra

pay cheque. Ortony et al. (1988) characterise this type of goal as an active pursuit goal; a different type of goal. In the system presented here, there is no difference between the goal to collect the regular pay cheque and the extra pay check except for the temporal context of the goal and the fact that they are generated through different processes. Furthermore, the characterisation of cyclical goal directed behaviour as simply periodic has been shown to be insufficient. Goals that recur cyclically do not necessarily do so with some fixed period. In the replenishment processes presented here, goals may be replenished either periodically or according to some timetable of deadlines. In fact a timetable of deadlines is generally more appropriate (whether or not there is a constant period of replenishment) when the agent is required to synchronise its activities with other agents or process not under its control. A final note is that intuitively it seems odd that a goal (whether cyclical or not) can never be satisfied, but still have activity directed towards achieving it. If an agent is instead driven by a stream of goals with limited life times, its cyclical goal directed behaviour can be more easily understood.

Carbonell (1982) proposes an initial taxonomy of "goal generators" in the context of a model of human goal-directed behaviour. This taxonomy is restricted and neither includes cyclical goal-directed behaviour not driven by physiological drives, nor the possibility of generating goals according to a timetable. However, cyclic physiological drives are described as "goals generated in response to internal physiological states that change with a certain periodicity". This description suggests that it is the internal physiological state that changes, and that goals are generated in response to this. Therefore, Carbonell (1982) is proposing different types of goal generators rather than different types of goal, a view that is consistent with the ideas presented in this chapter.

Slade (1994, pp. 49–56) expands on the taxonomy of Schank & Abelson (1977) to include fifteen types of goal (although a particular goal may lie in more than one category). The taxonomy includes "satisfaction goals" which roughly correspond to the physiological needs described by Carbonell (1982), and "preservation goals" which are concerned with maintaining the state of some variable or process within the environment (e.g. periodically changing the oil in a car) (also see Hammond et al. (1995)). These two categories correspond to the goals generated through replenishment discussed in section 4.5. The taxonomy follows Schank & Abelson (1977) by adding "enjoyment goals", "achievement goals", "crisis goals", "instrumental goals" and "delta goals". Achievement goals are a general category for achieving some new state of affairs, where enjoyment and crisis goals distinguish the goal in terms of the reason for their generation. Instrumental and delta goals are generated in the pursuit of other, higher level goals. In addition to these, Slade (1994) proposes "wishes and hopes" (such as having bought a lottery

ticket) and "worries and concerns" (such as being concerned about inflation) which are similar to the "interest goals" proposed by Ortony et al. (1988), who indicate that these goals are not normally actively pursued, and hence justify their distinction from "active-pursuit goals" (Ortony et al., 1988, p. 41). "Problems and accidents" and "opportunities" are also identified. These relate to problems with, or opportunities to satisfy an existing goal, although it is not clear whether the existence of a problem or opportunity changes the type of the goal that it concerns, or whether it causes a new goal to be generated to consider taking the opportunity or deal with the problem. Two of the examples used by Slade (1994) to illustrate opportunities highlight these two possible interpretations: (example 3.31, p. 54) John read People Magazine while waiting in the check-out line; and (example 3.32, p. 54) John asked the babysitter to mail a letter for him on her way home. In the first example, John may not have had a prior goal to have read People Magazine (the goal may have been generated simply because John noticed the magazine by the check-out line and wanted to pass the time), whereas in the second example John typically would already be motivated by the goal to have posted the letter. In this thesis, an opportunity to achieve an existing goal that is presently inactive affects the likelihood that the goal will become activated (see section 5.1), and may alter the way the goal is achieved. However, the existence of opportunities may also lead to a new goal being generated; an opportunity to easily reach a highly preferred state of affairs for example. A number of other goal types are proposed, some correspond to the meta-level goals suggested by Wilensky (1983, chap. 4) such as "values, ideals, and principles", and some to other reasons that a human may decide to generate goals such as "tastes and preferences" and "duties and responsibilities". Beaudoin (1994, p. 41) uses the term "motivator" to denote a broad class of control state that includes the subclasses "goal", "attitude" and "standard". A motivator is a state that "contain[s] dispositions to assess situations in a certain way [...] and [...] have the disposition to produce goals". The term goal is therefore used in a more restricted sense than Slade (1994); in fact Beaudoin's (1994) motivator is broadly equivalent to Slade's (1994) goal. This brief discussion has illustrated the wide range of interpretations of the terms "goal", "goal generator", "motivator" etc. and how their definitions are often indistinguishable. This thesis is not concerned with producing another taxonomy of goals or goal generators, but in the role of such control states and processes within an agent architecture. For this purpose, clear definitions of "motive", "goal" and "action" have been provided and are used consistently.

Ellis (1994) (see also Ellis & Nimmo-Smith (1993)) proposes three types of "performance intervals": Pulses, intermediates and steps. These are essentially step functions with varying duration, the pulse being the shortest duration. The duration represents the time interval within

which the agent predicts that a delayed intention should be achieved. For example, if an agent has the intention to attend a meeting at 2 o'clock, this would be represented as a pulse at 2 o'clock, or if the agent has the intention to meet a friend sometime today, this would be represented as a step function with a longer duration. During a performance interval it is appropriate for the agent to act on the delayed intention, and hence appropriate for the agent to be reminded of that delayed intention. However, if the agent is not reminded during this interval, there is no other time at which it is appropriate for the agent to be reminded of the intention. The performance intervals simply capture the predictions that an agent makes about when it is most appropriate for it to achieve a goal, with the width of the interval approximately corresponding to the accuracy of those predictions. These intervals say nothing about how relevant it is to be reminded of a delayed intention at any other time. Predictions of when goals are appropriate to achieve cannot be the only influence on the recall of delayed intention, and hence performance intervals are not sufficient as a basis for either a computational model of human goal-directed behaviour or for the design of an artificially intelligent system. The alarm processing machinery uses predictions of future events represented at various degrees of precision (i.e. predictions that are similar to performance intervals) as a basis for the alarm function. This alarm function however, represents more than predictions of deadlines and travel times, etc.: it provides a heuristic measure of the change in the relevance of a goal for consideration over time.

4.6.2 Conclusion

This chapter has introduced the notion of a goal and an alarm and describes importantly different mechanisms through which goals are generated. Goals may be generated either through decision (i.e. a process that involves reasoning, and is triggered by an event detected in the domain) or through replenishment (i.e. a process that involves no reasoning). An alarm is designed as a structure that encapsulates a goal and has the potential to subsequently trigger the agent's attention to that goal at some appropriate time in the future. The alarm is based on a simple and uniform "alarm function" through which the agent can manage all its goals in the same way, by the same processes, regardless of the mechanisms through which they are generated.

Chapter 5

Opportunities, dangers and time commitments

In the previous chapter, two distinct types of goal generation mechanism were introduced, and the alarm setting process described (sometimes referred to as suspension (Birnbaum, 1986)). This encapsulation prevents the agent from considering the goal until, ideally, the agent is reminded of the goal at some appropriate time between t_{dt} and t_{max}. However, at any time between the time that an alarm is set and its subsequent triggering, the domain may change in an unexpected way. If the domain changes in such a way that it may have some impact on an inactive goal (i.e. a goal that has been suspended by encapsulating it in an alarm structure), the goal may need to be considered in the light of this change. Three distinct and important transient effects on the intensity of an alarm are considered in this chapter: opportunities (section 5.1), dangers (section 5.2) and time commitments (section 5.3). Changes in the intensity of an alarm due to these effects may cause the goal to be considered earlier by the agent. (Note that these influences can only increase, and never decrease the intensity of an alarm.)

5.1 Opportunities to satisfy inactive goals

An opportunity is a timely set of advantageous circumstances, through which it may be possible for the agent (if all goes well) to satisfy a goal with minimum need for further planning. Unfortunately, during the time between the generation and triggering of an alarm, the goal encapsulated in that alarm is invisible to the reasoning and planning processes of the agent. So, there is no way that the agent could take advantage of an opportunity unless it is made aware of the goal when the opportunity is available. It is therefore necessary for the agent to employ mechanisms for the detection of opportunities to satisfy inactive goals. This section is concerned with the detection of and response to opportunities to satisfy inactive goals, while still satisfying the requirement that goals should require little processing effort before triggering.

5.1.1 Opportunity encoding

Opportunities to satisfy inactive goals are detected by demons (see definition 4.8). At the time that an alarm is set, the agent selects and encodes a limited number of opportunity demons. A demon monitors the state of the domain, and on triggering, the alarm associated with that demon is informed of the event (see figure 5.1). The conditions for the triggering of an opportunity demon are the preconditions of an action that, with the minimum of planning effort, can be used to satisfy the associated goal (see definition 5.1).

Opportunity demons are simple processes that do not involve deliberation. The set of propositions that characterise an opportunity demon are the preconditions of an action that satisfies the goal as one of its postconditions. The success of an opportunity demon relies on the agent selecting and encoding appropriate actions as demons. The selection of opportunity demons for a particular alarm, α, (i.e. $anticipated(\alpha)$) is made on the basis of a number of criteria.

1. An opportunity must allow the agent to achieve the goal with the minimum of planning effort. This limits the actions that can be used as opportunities. For example, the actions used in abstraction-based planning (Tenenberg, 1991; Georgeff & Lansky, 1987; Fox, 1993) vary in specificity. The more abstract the action, the more planning effort is required to refine it into primitive operators that can be executed. So, the less specific the action, the less use that action is as an opportunity. Some planners, use plan scripts or recipes (Pollack, 1992) that are designed to minimise planning effort for frequently used, but more long-term activities than primitive actions[1]. For example, an agent that frequently visits restaurants, may develop a well-defined restaurant script. This script requires little further planning effort to achieve, and so is a good candidate for selection as an opportunity. The characteristics of the planner and the actions used by the planner will influence the possible opportunities that can, or should be encoded.

2. A good choice of opportunity demon is one that is likely to occur. The agent's experience in its environmental niche can give an indication of what opportunities are more likely to present themselves. The more frequently an action is used by the agent (for whatever purpose), the more likely the preconditions of the action will hold at some time. For example, an agent may more frequently mitigate the need for cash by using an automatic cash machine outside a bank rather than entering the bank to cash a cheque. The use of an automatic cash machine may be available more frequently because the bank is only open

[1]Planners that use scripts or recipes are commonly referred to as case-based planners. In case-based planning these recipes are based on generalisations of prior plans that are recalled from a store of plans, possibly modified, and instantiated in the present circumstances (Kolodner et al., 1985; Hammond, 1989a).

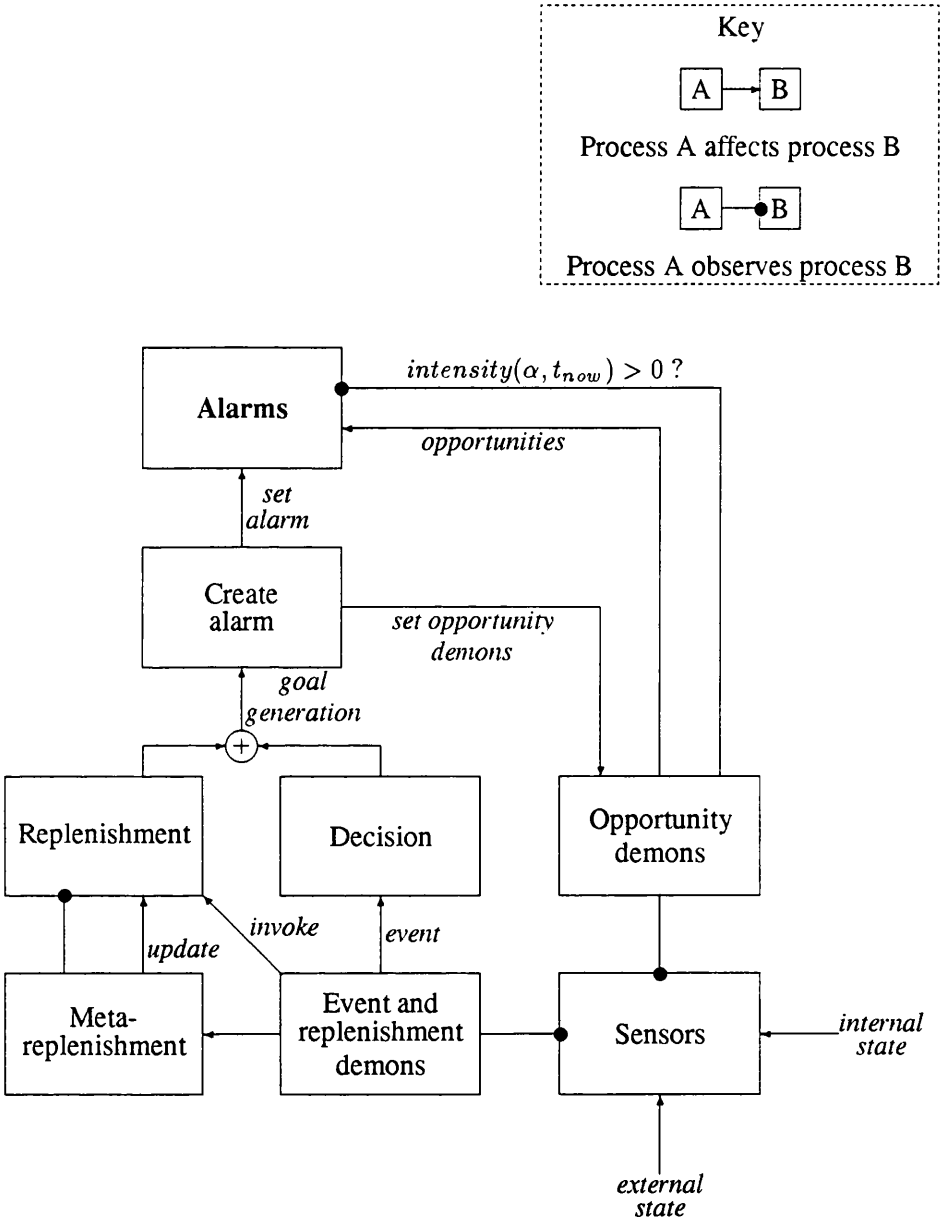


Figure 5.1: The detection of and response to opportunities.

a limited number of hours, or because the agent more frequently carries his cash card than his cheque book.

3. An opportunity is only useful if it is likely to be adopted by the agent. For example, a vegetarian will rarely, if ever, satisfy the goal to have mitigated hunger by eating a steak. Similarly, an agent may be more likely to encode the opportunity to use an automatic cash machine than to cash a cheque inside the bank because the agent does not like waiting in long queues. (Note that in this example, the use of a cash machine is both more likely to be available, and more likely to be adopted as an opportunity than cashing a cheque.) Those actions that are more likely to be adopted as opportunities if the situation warrants, are therefore more likely to be encoded as opportunity demons.

4. In general, the more important the goal is to the agent, the more opportunity demons the agent is likely to encode. Depending on the importance that an agent ascribes to the goal, a quota is allocated for opportunities. Then, depending on the criteria itemised above, a number of opportunity demons are encoded, but limited by this quota.

5.1.2 The timely detection of opportunities

An opportunity is a *timely* set of advantageous circumstances. So, the existence of an action that can be used with the minimum of planning effort to satisfy an inactive goal is not an opportunity unless there is some point in achieving the goal at this time. Consider the following example:

> Robby the robot, trundling home from a hard day at the car plant, passed an automatic recharge machine. Recognising this as an opportunity to satisfy its goal to be fully recharged, Robby considers acting on the goal. After choosing to act, and eventually satisfying the goal, a new goal is replenished. Suppose that this replenished goal has a delay time around an hour from now, and a deadline in five hours time. This new goal is encapsulated in an alarm that should trigger attention at some appropriate time in the future. However, as Robby is trundling on his way, the close proximity of the same automatic recharge machine again registers as an opportunity, but to satisfy the newly generated goal. The alarm encapsulating that goal directs Robby's attention to this new goal in the light of this opportunity.

The second opportunity in this example is inappropriate because there is no point in Robby recharging within an hour from the last time the goal to have recharged was satisfied. The fact that there is no point in Robby achieving the goal to have recharged the second time is reflected in the zero intensity of the associated alarm. Only if the intensity of an alarm is above zero will the

goal be relevant to the agent. In the same way that the alarms heuristic is used to direct planning attention, it is also used to direct the attention of the agent to opportunities. (Note, the notation used in this chapter is summarised in appendix A.)

Definition 5.1 If the goal is worth achieving (i.e. the intensity of the alarm is greater than zero), then if an action that was anticipated as a potential opportunity can be performed in the present state, it is an opportunity.

$$intensity(\alpha, t_{now}) > 0 \Rightarrow$$
$$opportunities(\alpha) = \{a : anticipated(\alpha) \mid pre(a) \subseteq D\}$$

where D is the present state of the domain, and the function $intensity(\alpha, t_{now})$ is specified in definition 7.6.

5.1.3 The effect of a detected opportunity

Opportunity demons provide the alarms heuristic with the flexibility to respond opportunistically to changing circumstances. However, this flexibility cannot compromise the robust nature of the alarms heuristic in directing and limiting reasoning attention. It is not acceptable for the detection of a situation that may give the agent an opportunity to satisfy a goal, to unconditionally demand attention; cf. Birnbaum (1986), Patalano et al. (1993), and Simina & Kolodner (1995). If the agent is pursuing a highly important and urgent goal and there exists a potential opportunity to satisfy a less important goal, it may not be reasonable for reasoning resources to be consumed by considering the opportunity at that critical time. Consider the following example:

> Robby the robot is racing home from a hard day at the car plant being pursued by a manic scrap merchant. On passing an automatic recharge machine, Robby detects the opportunity and immediately considers stopping to satisfy its goal to have recharged. Unfortunately, while Robby is distracted by the opportunity, he wraps himself around a lamp post and is taken away for scrap.

It is inappropriate for Robby to consider the opportunity in such a situation. Robby should direct all attention to escaping the manic scrap merchant. If the existence of an opportunity does trigger the attention of the agent, the intensity of the associated alarm must increase above the threshold; i.e. $intensity(\alpha, t_{now}) > \tau$ (τ being the current threshold level). However, if the threshold is greater than i_{max}, for the agent to become aware of the opportunity at this time, the intensity of the alarm must be increased above i_{max}. If the existence of an opportunity can increase the intensity of an alarm above i_{max}, the opportunity has effectively changed the importance of the goal to the agent; this is not a sensible effect. The existence of an opportunity

simply means that the goal is relevant to the agent now; i.e. it changes the temporal relevance of the goal, not its importance.

In the presence of an opportunity the temporal relevance of an alarm increases to maximum. So, the existence of an opportunity simply makes it *more likely* for the agent to be reminded of that goal. (The agent will only be reminded of the goal if the importance of the goal, i_{max}, is greater than the current threshold level, τ.) Reconsider Robby's predicament, but assume that the existence of a potential opportunity simply increases the intensity of the associated goal. While being chased by the scrap merchant, Robby is pursuing the important and urgent goal to escape, and hence will be relatively insensitive to distraction; represented by an increased threshold, see section 6.1. In such a situation, it is unlikely that Robby will be reminded of the goal to be fully charged. (Compare this with the work of Birnbaum (1986), Patalano et al. (1993) and Hammond (1989b) discussed in section 5.4.1.) Therefore, the more important an inactive goal is to the agent (i.e. the greater i_{max}), the more impact a detected opportunity to achieve that goal has on the agent, and hence the more likely it is that the goal will be considered in the light of that opportunity.

The mechanism by which opportunities are registered is by adding the actions that are opportunities to achieve a goal, to the associated alarm structure once they are detected. Then, if the opportunity passes without the agent considering the goal, the opportunity is removed from this list (see appendix B.2.5). Refer to page 115 for a discussion on the processing of goals that trigger in the light of an opportunity, and to sections C.3 and C.7 and the paragraph below for examples of the effects of opportunities on alarm triggering in the warehouse simulation.

Example from simulation

Figure 5.2 (produced using data from the simulation of the alarm processing machinery, see appendix C) illustrates the behaviour of a number of goals that are generated in the service of the motive to have the warehouse fully stocked. An opportunity to satisfy one of these inactive goals occurs if the agent has a plan to order stock. For example, if a goal to have restocked the warehouse with ice cream is activated and the agent is constructing a plan to satisfy this goal, this constitutes an opportunity to satisfy other restock goals because other commodities may be added to the order. This is exactly what occurs between 6am and 7am in the simulation. The alarm encapsulating the goal to have restocked the warehouse with ice cream triggers and is activated. The alarms are then sampled at five minutes past six, and all the alarms encapsulating goals to have restocked the warehouse with some commodity that already have an intensity above zero, trigger. In the simulation the agent considers activating the goals to have restocked the warehouse with yoghurt, salsa and coffee, but only yoghurt is ordered, the other alarms are mitigated

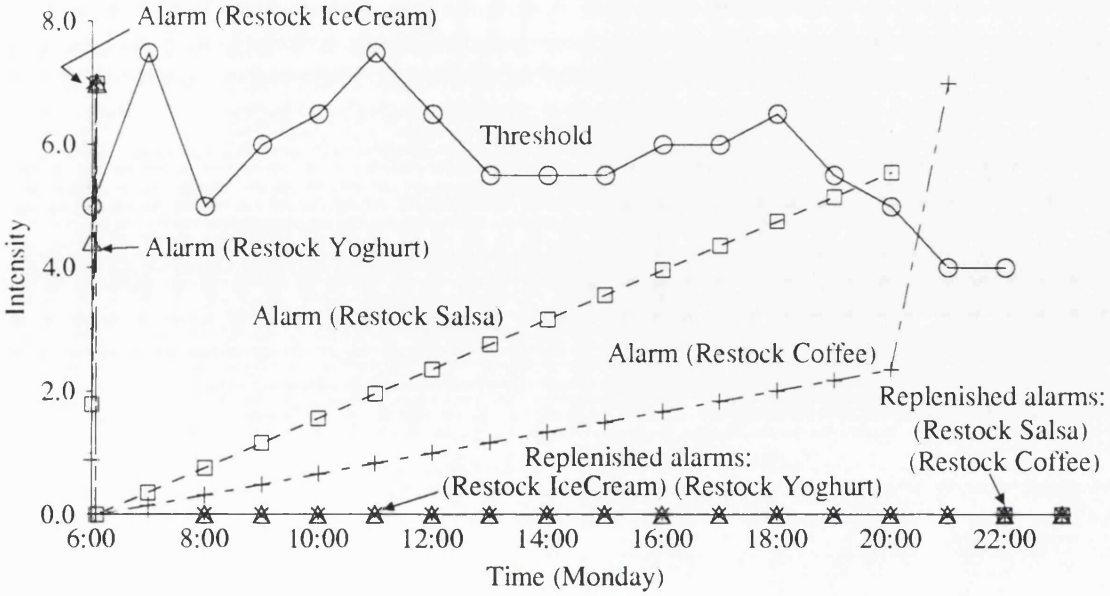


Figure 5.2: The behaviour of alarms encapsulating various goals generated in the service of the motive to have the warehouse fully stocked.

(see section 6.2.3 for a description of the effects of mitigation). Later, at 8pm the alarm encapsulating the goal to have restocked the warehouse with salsa (one of mitigated alarms) triggers. The activation of this goal again constitutes an opportunity to satisfy the goal to have restocked the warehouse with coffee. This goal is also activated. Note that at 8am two new goals to have restocked the warehouse with ice cream and yoghurt are replenished (all these goals are cyclical replenishment goals). However, the alarms encapsulating these goals remain at zero intensity, and so the activation of the goal to have restocked with salsa at 8pm does not constitute and opportunity to satisfy either of them. (There would be little point in even considering these goals so soon after they have been achieved and then replenished.)

5.2 Dangers to the timely satisfaction of goals

This section is concerned with what constitutes a danger to the timely satisfaction of a goal. If a goal that is encapsulated within an alarm is to be satisfied in time, the alarm must trigger so that the agent becomes aware of that goal at some appropriate time between t_{dt} and t_{max}, but at least before t_{max} if the agent's predictions are accurate (see section 4.3). For example, if the warehouse agent accepts an order from a customer, it needs to be reminded of the goal in sufficient time for the order to be prepared before the customer arrives. So, an alarm function is defined so that it reaches maximum intensity at the time that the customer is expected to arrive minus the estimated time required to prepare the order. (Note, if the threshold is lower than i_{max},

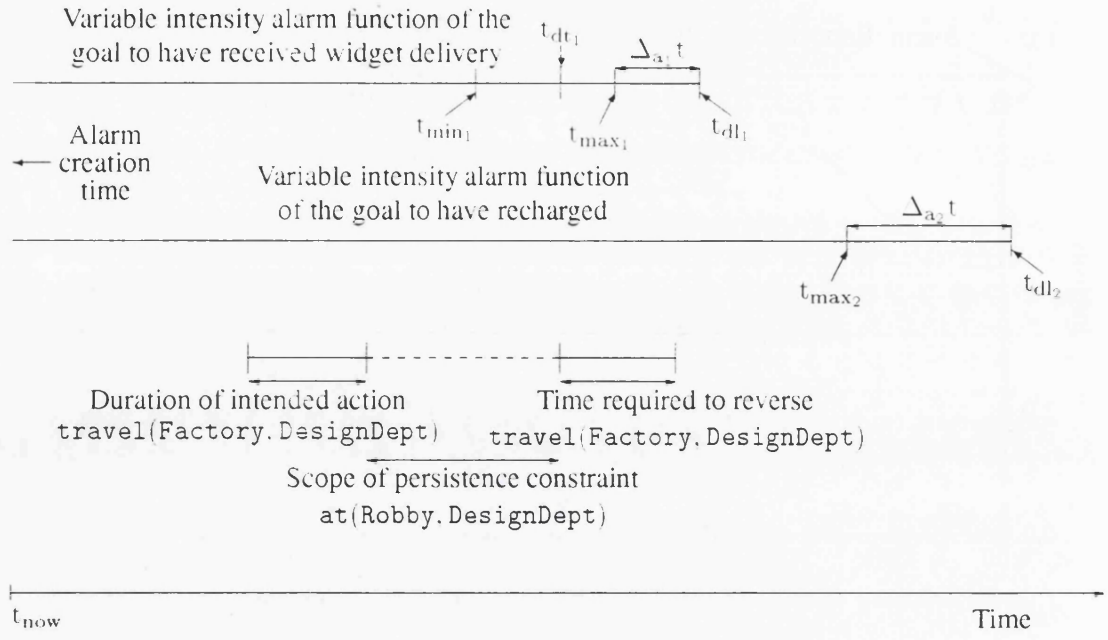


Figure 5.3: An example interaction between an intended action and an inactive goal.

then the alarm will trigger before t_{max}, but if the threshold is greater than i_{max}, the alarm will not trigger until the threshold drops back below i_{max}, see chapter 6.) If, because of changing circumstances, the time required to achieve an inactive goal is significantly increased, there is a danger to the timely satisfaction of the goal. Consider the following example, illustrated in figure 5.3:

> Robby the rebuilt robot is working at the car plant when he receives instructions from the boss to attend a design meeting. The meeting is soon, so Robby directs attention towards this goal, and proceeds to construct a plan to achieve it. The plan involves travelling from the plant to the design department building on the other side of town. Robby has a number of other inactive goals. One of these is to receive a delivery of widgets scheduled for later that day (widgets are an essential element in the manufacturing process). The delivery truck with these widgets is expected to arrive just after the design meeting, but Robby is unaware of this because the goal is inactive. Later, during the design meeting Robby is reminded of the widget delivery. Robby will not have enough time to get back to the plant after the meeting to meet the delivery truck. Robby has the choice of either leaving the meeting to return to receive the delivery, or risk the car plant being short of widgets: either way the boss will sack him.

If Robby was reminded of the inactive goal to receive the widget delivery before leaving for the design meeting, the delivery could have been rescheduled or the boss could have been informed of this conflict. The conflict can only be resolved if Robby is aware of it. If there is such a conflict between an intended action and an inactive goal, the alarm encapsulating the goal is no longer appropriate to the timely reminding of the agent (see section 5.2.2). Furthermore, if an agent finds itself in a situation where the time required to achieve an inactive goal is significantly increased through the actions of some other agent or process, this also constitutes a danger to the timely satisfaction of that goal (see section 5.2.3).

5.2.1 Alarm appropriateness conditions

The definition of an alarm function is typically made under certain assumptions about the normal behaviour of the agent (see section 4.3). For example, if Robby generates a goal to receive the widget delivery at the plant at 4pm, then he should remember that goal at least five or ten minutes before this estimated arrival time of the truck. This will generally give Robby sufficient time to suspend his present activities and meet the delivery truck at the entrance to the plant. So, Robby suspends the goal by encapsulating it in an alarm function that is defined so that the alarm reaches maximum about ten minutes before the expected arrival time. However, remembering the delivery five or ten minutes to four o'clock is only appropriate if Robby is at the car plant. In this example, the alarm is only appropriate to the agent if the condition `at(Robby, Plant)` holds. If Robby subsequently intends to be somewhere else at the time, there is a danger to the timely satisfaction of this inactive goal.

In the example illustrated in figure 5.3, it takes about an hour for Robby to travel from the design meeting back to the plant. If the design meeting was scheduled an hour earlier (i.e. 3.30pm rather than 4 o'clock), Robby would have one and a half hours to get back from the meeting. In this case, Robby would have sufficient time to get back to the plant to meet the delivery truck in time, and there is no danger to the timely satisfaction of the inactive goal. Therefore, the time that it takes to reverse the dangerous effects of an action is critical in determining whether or not an inactive goal is in danger. Furthermore if the agent detects that such a condition on the appropriateness of an alarm does not hold in the domain, the time required to reinstate that condition is critical in determining the effect of this danger on the intensity of an alarm (see section 5.2.3). Suppose that a particular alarm is only appropriate if the agent has sufficient money. The agent may have a simple rule of thumb for how long it generally takes to mitigate this need. However, if an alarm is only appropriate when the agent is at a particular location, the time required for the agent to get to that location depends on where it is travelling from. For example, if Robby has a goal to receive a widget delivery at the plant, then if Robby is at the plant it may

take 10 minutes to get to the loading bay, but if Robby is not at the plant, but in the same town it may take an hour, and if Robby is not in the same town, but in the same country it may take a day, and so on.

An appropriateness condition is a list of tuples, each tuple comprising of a proposition and a period of time. For example, the appropriateness condition for Robby's alarm to receive the widget delivery may be:

```
{(at (Robby, Plant), 1 hour),
 (at (Robby, Roboville), 1 day),
 (at (Robby, Roboland), 1 week)}
```

Then if Robby plans not to be or is not at the plant, but is in Roboville it will take about an hour to reinstate the appropriateness condition. If Robby plans not to be or is neither at the plant, or in Roboville, but is in Roboland, it will take about a day to reinstate the appropriateness condition.

In general, an agent should be reminded of an inactive goal in time if the appropriateness conditions of the alarm encapsulating that goal hold. The detection of dangers to the timely satisfaction of inactive goals relies on the agent encoding relevant appropriateness conditions as danger detectors. The appropriateness conditions of an alarm are selected and encoded when the alarm is generated and the opportunity demons for that alarm are encoded (see figure 5.4).

5.2.2 Dangers due to conflicts between intended actions and inactive goals

The planning attention of an alarm-driven agent is restricted to a small number of salient goals: the agent's active goals. In this way, the planning problem is simplified. However, it is possible that the goals the agent is attending to will interact with other goals that the planner is presently unaware of (i.e. inactive goals). Some action in the agent's plan may change the domain such that an inactive goal may no longer be satisfied in time. The planner is unaware of the inactive goal, it is unaware of the interaction, and so does not see that there is a conflict to resolve.

If an agent intends to perform an action that will delete one of the alarm appropriateness conditions, then this effect must be reversed before the goal can be satisfied. This adds to the time required to satisfy the goal. Whether or not the agent intends this effect of its action, there is a danger to the timely satisfaction of the goal.

If there is insufficient time for the agent to reverse a dangerous effect after the action that produced the effect is completed and before t_{max}, then the agent must be made aware of the inactive goal at least before the action is performed. However, if the agent *intends* the dangerous effect of such an action, a persistence constraint (sometimes referred to as a causal link (Penberthy & Weld, 1992)) may be posted on this effect, protecting it from being reversed for some

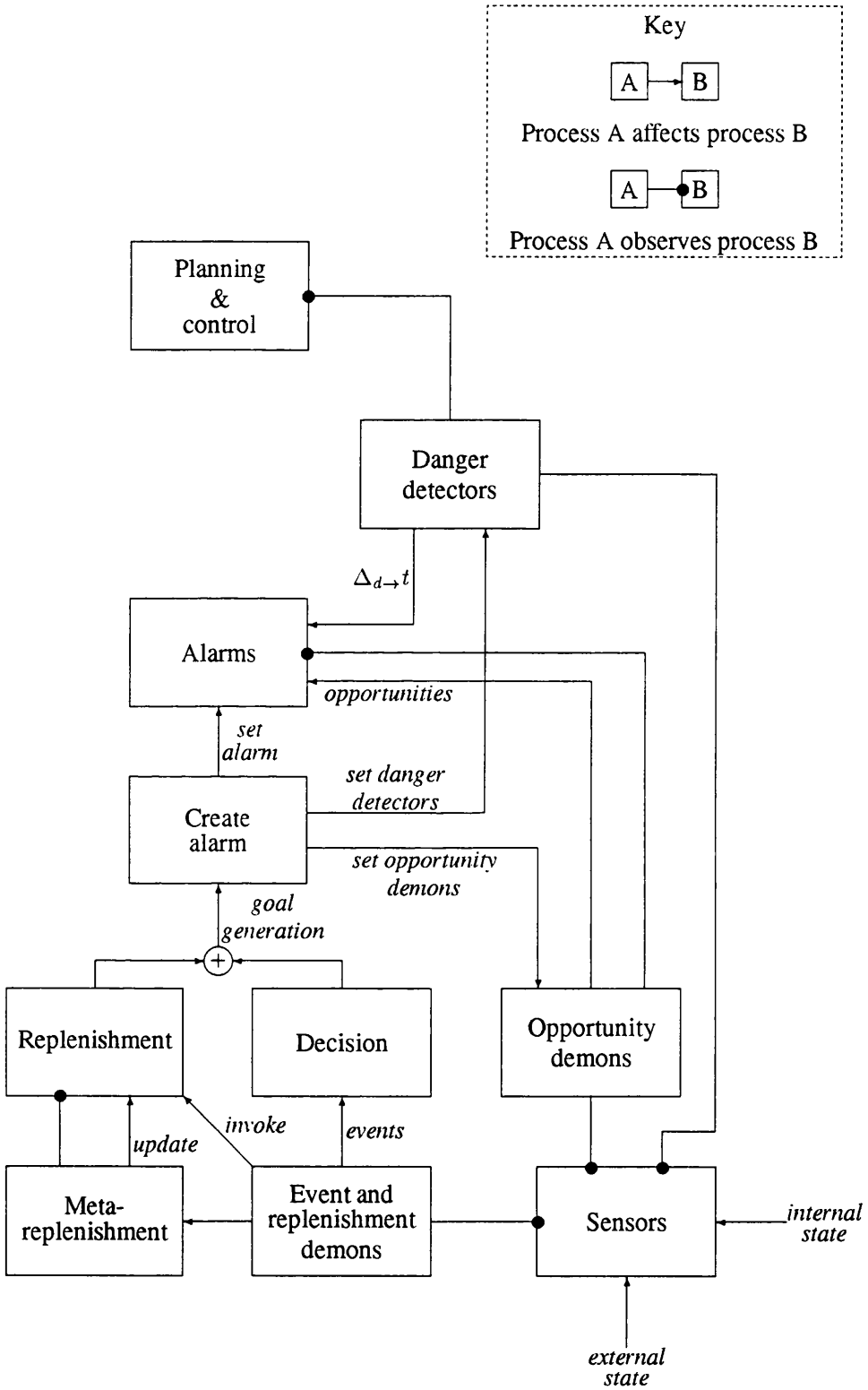


Figure 5.4: The detection of and response to dangers.

time.[2] This is the case in the example illustrated in figure 5.3. Robby plans to attend the meeting, and in doing so a constraint is posted on the proposition `at(Robby, DesignDept)`. This commits Robby to remaining at the location of the meeting for the duration of the meeting. The dangerous effect can only be reversed after this constraint is lifted, but there is insufficient time for Robby to return to the car plant after the meeting and before the delivery truck is expected to arrive. The combination of an action that produces a dangerous effect on an inactive goal, a persistence constraint on that effect, and the time required to reverse the effect, create a conflict between the intended action and the goal. The effect of a dangerous action in the agent's plan is to shift the alarm function by an appropriate period of time so that the function reaches maximum at an earlier time; i.e. some time before a time $\Delta_a t$ before the dangerous action is to be performed. (Note, the notation used in this chapter is summarised in appendix A.)

Definition 5.2 If there exists a persistence constraint in the agent's plan that constrains a proposition p_{pcs} to hold until t_{pcs},[3] p_{pcs} conflicts with the jth appropriateness condition $(p_j, \Delta_j t)$ of the alarm α, and the constraint holds at some time between $(t_{max} \ominus \Delta_j t)$ and t_{dl}, then the alarm should be shifted so that the alarm reaches maximum a period $\Delta_a t$ before the endangering action a is performed (remember $\Delta_a t$ is an estimate of how long the agent will take to achieve the inactive goal), i.e. the alarm reaches maximum at $(end(a) \ominus duration(a) \ominus \Delta_a t)$. (This p_{pcs} is an *intended* consequence of the planned action, a.)

$$(a, p_{pcs}, t_{pcs}) \in pcs(\pi) \wedge (p_j, \Delta_j t) \in apps(\alpha) \wedge conflict(p_j, p_{pcs}) \wedge$$
$$\neg(before(t_{pcs}, (t_{max} \ominus \Delta_j t)) \vee before(t_{dl}, (end(a) \ominus duration(a)))) \Rightarrow$$
$$\Delta_{d \rightarrow} t = \begin{cases} (t_{dl} \ominus end(a)) \oplus duration(a) & \textbf{if } before(end(a), t_{dl}) \\ duration(a) \ominus (end(a) \ominus t_{dl}) & \textbf{if } before(t_{dl}, end(a)) \\ duration(a) & \textbf{otherwise} \end{cases}$$

Else if the postcondition of an action in the agent's plan conflicts with the jth appropriateness condition $(p_j, \Delta_j t)$ of the alarm α, and the interval in which that action is to be performed overlaps the interval $(t_{max} \ominus \Delta_j t)$ to t_{dl}, then the alarm should be shifted so that it reaches maximum at $(end(a) \ominus duration(a) \ominus \Delta_a t)$. (This p is an *unintended* consequence of a.)

$$a \in acts(\pi) \wedge p \in post(a) \wedge (p_j, \Delta_j t) \in apps(\alpha) \wedge conflict(p_j, p) \wedge$$
$$\neg(before(end(a), (t_{max} \ominus \Delta_j t)) \vee before(t_{dl}, (end(a) \ominus duration(a)))) \Rightarrow$$

[2] The precise definition of what constitutes a danger to the timely satisfaction of an inactive goal depends on the planning algorithm used by a particular agent design. In this discussion it is assumed that the planner uses causal-link-style constraint-posting (Penberthy & Weld, 1992; Weld, 1994). However, this does not imply that the alarm processing machinery is restricted to this or any other type of planning algorithm.

$$\Delta_{d\rightarrow t} = \begin{cases} (t_{dl} \ominus end(a)) \div duration(a) & \textbf{if } before(end(a), t_{dl}) \\ duration(a) \div (end(a) \ominus t_{dl}) & \textbf{if } before(t_{dl}, end(a)) \\ duration(a) & \textbf{otherwise} \end{cases}$$

Shifting the alarm function in this way ensures that the function of intensity associated with the inactive goal increases to maximum so that it can be achieved before the action is performed. Consider the example where Robby has the inactive goal to have received the widget delivery encapsulated in an alarm that reaches maximum at 3.50pm (i.e. 10 minutes before the arrival time of the delivery truck at 4pm), and it generally takes about an hour to unload a truck. At 12 noon, Robby is instructed to attend and report on the design meeting which is scheduled for 2.30pm; this meeting is expected to take about an hour. To comply with instructions, Robby must remain at the meeting until it finishes. So, there is a persistence constraint posted on the proposition `at(Robby, DesignDept)` until the meeting is finished, which is expected to be at 3.30pm. Robby knows that it takes an hour to get the design department building from the car plant. In constructing the plan, the agent intends to travel from the plant to the design department building; this action is recognised as a danger to the inactive goal to have received the widget delivery, g_w, because:

$$(\texttt{travel(Plant, DesignDept)}, \texttt{at(Robby, DesignDept)}, 3.30\text{pm}) \in pcs(\pi) \wedge$$
$$(\texttt{at(Robby, Plant)}, 1\text{hr}) \in apps(\alpha_w) \wedge$$
$$conflict(\texttt{at(Robby, DesignDept)}, \texttt{at(Robby, Plant)}) \wedge$$
$$\neg before(3.30\text{pm}, (3.50\text{pm} \ominus 1\text{hr})) \wedge$$
$$\neg before(5\text{pm}, (2.30\text{pm} \ominus 1\text{hr}))$$

Therefore, the alarm function encapsulating the goal to receive the widget delivery is shifted by $((5\text{pm}-2.30\text{pm})+1\text{hr}) = 3\text{hrs}\,30\text{mins}$. So, this alarm function will now reach maximum at 12.20pm, reminding Robby of the conflict before the dangerous action is performed. Note that there is no point in Robby acting on this goal now, but he is now aware of this goal, and so the planner will attempt to resolve the conflict.

5.2.3 Dangers due to changes in the domain

Not only is it possible for the agent to intend to perform an action that will be a danger to the timely satisfaction of an inactive goal, but other agents or processes may endanger goals. The

[3]The proposition p_{pcs} is typically a precondition of another action in the plan. Thus, a persistence constraint (or causal link) normally consists of the action that satisfies the proposition, the proposition itself, and the action that consumes this uses this proposition as a precondition. Here, a persistence constraint consists of the action that satisfies the proposition, the proposition and a time that the proposition should hold until. This choice of representation is used to minimise the complexity of the definition.

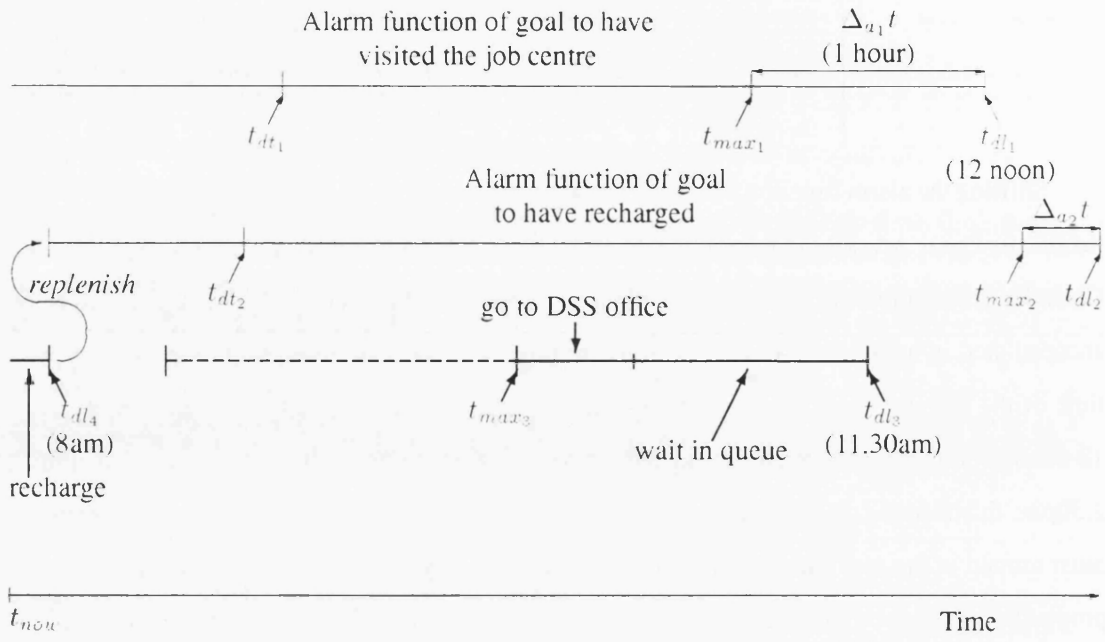


Figure 5.5: An example interaction between a time commitment and a goal.

domain may change either spontaneously, or through the action of another agent, such that an alarm encapsulating an inactive goal is no longer appropriate.

So, if an alarm appropriateness condition ceases to hold due to the influence of some other agent or process, then there is a danger to the timely satisfaction of the inactive goal. In this circumstance, the agent has no warning that this change was to occur, so all that can be done is to shift the alarm function by time enough for the danger to be removed before t_{max}.

Definition 5.3 If a proposition holds in the domain which conflicts with the jth appropriateness condition $(p_j, \Delta_j t)$ of α, then the alarm should be shifted so that it reaches maximum at $(t_{max} - \Delta_j t)$. (This p is either an unanticipated consequence of the agent's action, or due to the action of some other agent.)

$$p \in D \wedge (p_j, \Delta_j t) \in apps(\alpha) \wedge conflict(p_j, p) \Rightarrow \Delta_{d\rightarrow} t = \Delta_j t$$

Such a danger to the timely satisfaction of an inactive goal will be detected by employing a danger demon which fires when an alarm appropriateness condition no longer holds in the domain, see figure 5.4 and appendix B.2.5.

5.3 Time commitments

In the construction of a plan, the agent commits itself to activity at certain times in the pursuit of its goals. These commitments will reduce the time available for the agent to act to satisfy other

goals. The effect of a prior commitment to action by the agent is different from a danger to the timely satisfaction of a goal. A danger increases the time required to satisfy the goal, where a time commitment reduces the free time available for action in pursuit of the goal. Consider the following example, illustrated in figure 5.5.

> Robby the redundant robot has a plan of action that involves collecting his benefit from the department of social security (DSS) before 11.30am. (Note, Robby can start to act on this goal any time between 8.30am and 10am because the goal must be achieved before 11.30, and the DSS office does not open until 9am.) In planning to achieve this goal, Robby commits to the time required to get to the DSS office and about an hour waiting time. Robby also has the intention to recharge before 8am, and the inactive goal to have gone to the job centre. Later, while waiting in a queue at the DSS office Robby remembers that he had wanted to go to the job centre before lunch. Both the job centre and the DSS office are in Roboville, and the appropriateness conditions of the alarm encapsulating the goal to have visited the job centre are not corrupted as long as Robby remains in Roboville. So, no dangers will be detected; it will take no longer for Robby to achieve the goal for having visited the job centre. However, the commitments made in planning to collect the benefit from the DSS office reduces the time that Robby will have at the job centre.

If Robby was reminded of the goal to visit the job centre earlier, his plans could have changed so that he is not presented with this dilemma. For example, Robby may decide to go to the job centre after recharging and before going to the DSS office. The following subsections discuss the processes by which a plan is interpreted to determine the agent's time commitments, the detection of conflicts between inactive goals and time commitments, and the effect of these conflicts on alarms. If an agent is able to compensate for the time commitments made in the process of planning, it may consider inactive goals early enough to have sufficient time to satisfy them before their deadlines.

5.3.1 Plan interpretation

A time commitment is an interval of time in which the agent is committed to activity. So, a plan can be viewed as a number of intervals of time commitments; i.e. the intervals in which the agent intends to act. A typical plan is viewed as a partially ordered sequence of actions with one or more time constraints. For example, if Robby must collect his benefit before 11.30am, any plan to achieve that goal will have 11.30am as a time constraint on the final action in the plan. The plan to achieve Robby's goal to have recharged (see figure 5.5) also has at least one

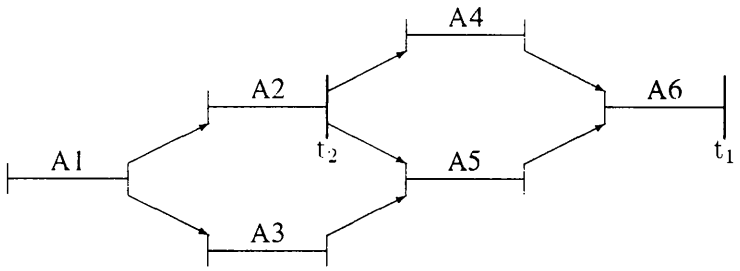


Figure 5.6: A partially ordered plan with two external time constraints, t_1 and t_2.

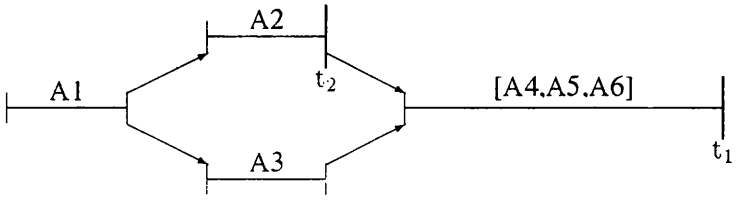
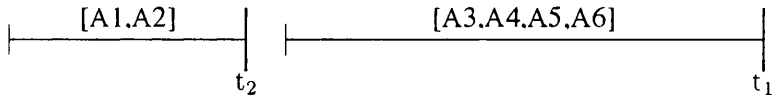


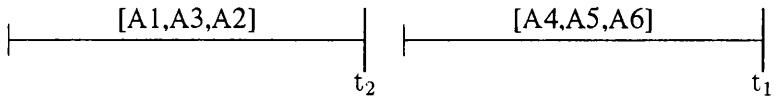
Figure 5.7: Abstraction of A4, A5, and A6.

time constraint. In this case Robby wishes to have achieved the goal by 8am. However, if the goal is not achieved by this deadline, the goal can still be achieved and Robby will wish it to be achieved as soon as possible. So, if it is 8.20am, Robby has not completed the plan to have recharged and it will take about ten more minutes to complete the plan, the time constraint in the plan will change to around 8.30am to reflect the fact that Robby wishes to have recharged as soon as possible. So, some deadlines are more flexible than others, but *all* goals have a limited temporal relevance. Therefore, all plans have a minimum of a single time constraint, before which all actions in the plan should be performed.

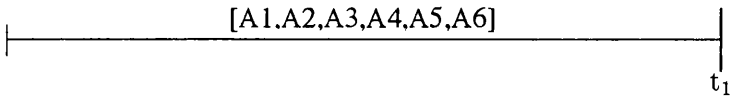
The example illustrated in figure 5.6 is a plan consisting of six actions (A1, ..., A6) and two time constraints, t_1 and t_2, associated with the actions A6 and A2 respectively. The time commitments made by the agent in constructing this plan can be calculated by simply propagating these constraints through the plan and collecting together periods of activity. In this example, action A6 must be performed before t_1, and both A4 and A5 must be performed before A6 (the order of A4 and A5 is unimportant). The combination of these three actions is the single time commitment of duration $(duration(A4) \div duration(A5) \oplus duration(A6))$ (figure 5.7). The three actions A4, A5, and A6 are abstracted into a single time commitment [A4,A5,A6].



(a): Two time commitments [A1,A2] and [A3,A4,A5,A6].



(b): Two time commitments [A1,A3,A2] and [A4,A5,A6].



(c): The single time commitment [A1,A2,A3,A4,A5,A6].

Figure 5.8: Abstracted time commitments.

In this process of plan interpretation, it is assumed that the agent does not do more than one thing at a time. If an agent is able to perform two actions at once, the time taken to complete both actions will be less than the sum of their expected durations. So, this procedure will tend to produce overestimates of the time that the agent will actually take to execute the plan. However, if the agent can predict that there will be an opportunity to perform more than one action at once, this should be reflected in the expected durations of the actions within the plan. Furthermore, an estimate of the length of time required to perform an elemental action will typically be better than for a more abstract action with an hierarchical plan structure. The time commitments generated through such a plan interpretation procedure represent the "best yet" estimates of how long it will take for the agent to perform the actions in the current plan. Generally, as planning progresses and more information becomes available to the agent, these estimates will improve.

In the example, if the action A3 can be performed after t_2 and before the start of the time commitment [A4,A5,A6] then it can be added to this time commitment; figure 5.8(a). If A3 cannot be fit in this space, it is combined with A2 and A1 to produce a second time commitment [A1,A3,A2]; figure 5.8(b). Furthermore, if t_2 is not before the start of the time commitment [A4,A5,A6], then both time commitments are combine into one; figure 5.8(c).

5.3.2 The detection of conflicts between time commitments and inactive goals

In transforming a plan into a list of time commitments, it is assumed that the agent will leave all activity to the last moment in achieving its goals before their deadlines. This is justified because the objective of the alarm mechanism is to remind the agent of goals in sufficient time for them to

be achieved in time. Therefore, the agent must detect situations in which the agent plans activity that *may* result in there being insufficient time available to achieve another goal.

Refer to the example illustrated in figure 5.5. Suppose that Robby goes to the DSS office at the last possible moment to achieve the goal before 11.30am. If the interval of time t_{max_2} to t_{dl_2} overlaps the interval t_{max_1} to t_{dl_2}, and the agent is reminded of the inactive goal at t_{max_1} (worst case) then one of the goals must be sacrificed. Now, if the goal to have visited the job centre has a deadline, t_{dl_1}, of 9.50am instead of 12 noon, there is no conflict between the time commitments made in planning to achieve the active goal (assuming action at the last minute) and the interval t_{max_1} to t_{dl_1}. Hence, there will be no effect on the alarm encapsulating the goal to have visited the job centre due to these agent's time commitments. Suppose that Robby starts to act on his plan to have collected his benefit before t_{max_2}; e.g. Robby could leave for the DSS office at 8.30am. Even if the inactive goal does not trigger attention until t_{max_1}, there is sufficient time to suspend the current plan, go to the job centre before 9.50am and then return to achieve the original goal (i.e. collect the benefit) before 11.30. Therefore, if the effects of the agent's time commitments on its inactive goals are determined using the assumption that everything will be left to the last minute, then all possible situations in which an inactive goal cannot be satisfied in time because of prior commitments can be detected.

5.3.3 The effect of conflicting time commitments on alarms

If the agent is committed to activity during the period t_{max} to t_{dl}, then there is a possibility that the agent will not be reminded of an inactive goal in time for it to be achieved before the deadline. The existence of a conflicting time commitment causes the alarm function to be shifted so that it reaches maximum at an earlier time. (Note, the notation used in this chapter is summarised in appendix A.)

Definition 5.4 In the simplest terms, a time commitment, $(\Delta_{tc}t, t_{tc})$, conflicts with an alarm if the interval $(t_{tc} \ominus \Delta_{tc}t)$ to t_{tc} overlaps the interval t_{max} to t_{dl}. If this is the case, the alarm must be shifted so that it reaches maximum at an appropriate time before $(t_{tc} \ominus \Delta_{tc}t)$; the alarm is shifted by $\Delta_{tc\rightarrow}t$.

$$(\Delta_{tc}t, t_{tc}) \in tcs(\pi) \wedge \neg\,(before(t_{tc}, t_{max}) \vee before((t_{tc} \ominus \Delta_{tc}t), t_{dl}))$$

$$\Rightarrow \Delta_{tc\rightarrow}t = \begin{cases} (t_{dl} \ominus t_{tc}) \dotplus \Delta_{tc}t & \text{if } before(t_{tc}, t_{dl}) \\ \Delta_{tc}t \ominus (t_{tc} \ominus t_{dl}) & \text{if } before(t_{dl}, t_{tc}) \\ \Delta_{tc}t & \text{otherwise} \end{cases}$$

However, after the alarm is shifted, it may conflict with another time commitment interval; i.e. if there are two time commitments $\{(\Delta_{tc_1}t, t_{tc_1}), (\Delta_{tc_2}t, t_{tc_2})\} \subseteq tcs(\pi)$ such that the

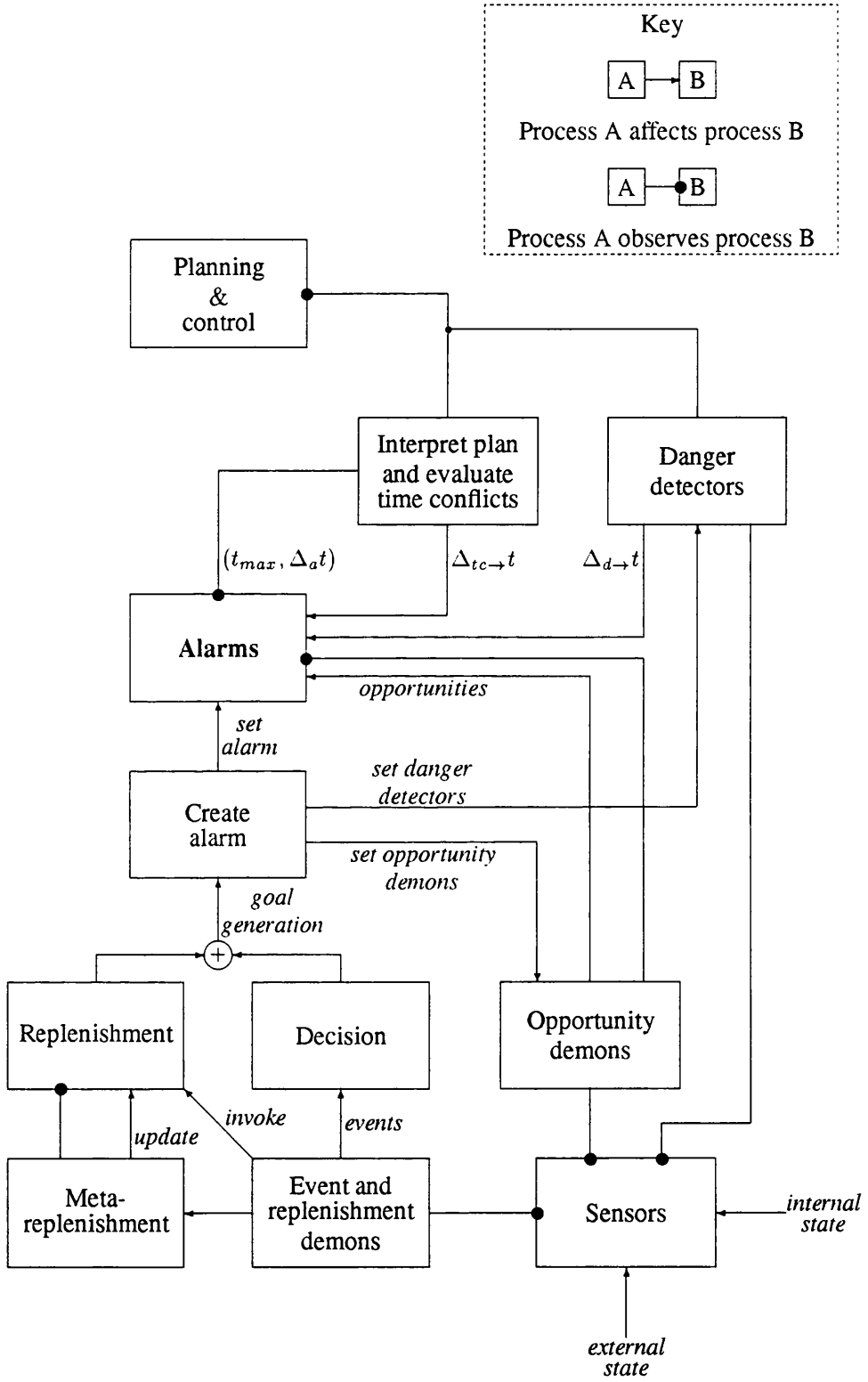


Figure 5.9: The influences of opportunities, dangers and time commitments.

interval $(t_{tc_1} \doteq \Delta_{tc_1} t)$ to t_{tc_1} overlaps the interval t_{max} to t_{dl} and the interval $(t_{tc_2} \doteq t_{tc_2})$ to t_{tc_2} overlaps $(t_{max} \doteq \Delta_{tc_1 \rightarrow} t)$ to $(t_{dl} \doteq \Delta_{tc_1 \rightarrow} t)$, then the alarm must be again shifted so that it reaches maximum before $(t_{tc_2} \doteq \Delta_{tc_2} t)$. Therefore, the alarm is shifted so that the interval $(t_{max} \doteq \Delta_{tc_j \rightarrow} t)$ to $(t_{dl} \doteq \Delta_{tc_j \rightarrow} t)$ does not overlap any time commitment $(\Delta_{tc_k} t, t_{tc_k}) \in tcs(\pi)$, where $\Delta_{tc_j \rightarrow} t$ is the time shift due to the time commitments $(\Delta_{tc_1} t, t_{tc_1})$ to $(\Delta_{tc_j} t, t_{tc_j})$ inclusive. This algorithm is given in appendix B.2.4. (A number of examples of the effect of time commitments on the triggering of an alarm can be seen in the warehouse domain simulation in appendix C, and specifically sections C.2, C.7 and C.8.)

Note that it does not matter if the only time slot that can be found to fit in the inactive goal is either before the delay time or before the time now; i.e. $before(t_{max} \doteq \Delta_{tc \rightarrow} t, t_{now})$ may be true. The purpose is *not* to schedule the planning activity of the agent, but to schedule when the agent considers its goals. A decision about whether the goal must be abandoned due to lack of time is the responsibility of the planner.

Generally, this heuristic procedure underestimates the time available to the agent. It may be possible for the agent to pursue an inactive goal at the same time as certain actions in its plan. For example, I can read a paper while travelling to work if I take the train, but not if I ride a bicycle. If such opportunities to achieve two goals at the same time are considered, a better estimate of the effects of the agent's time commitments may be made. However, the cost involved (this process will almost certainly require deliberation for it to be effective) will, in general, outweigh any small benefit gained from a better estimate.

It is often possible for the agent to partially achieve a goal, perform an unrelated action in the service of some other goal, and then return to complete it. Suppose the warehouse agent has an inactive goal to have tidied the loading bay. It is possible for the agent to partially tidy this area of the warehouse, pause to recharge its batteries or place an order for new stock, and then return to this goal without affecting the outcome. If it is known that achieving a goal can be interleaved with other activities to some extent, then a better estimate of the time available to the agent for action on this goal can be found. For example, the agent may estimate that it takes an hour to achieve the goal to have tidied the loading bay, and it may have two half hour periods of time in which it has no commitment. In this case, the use of the knowledge that the time required to achieve the goal (i.e. $\Delta_a t$) can be interleaved with other activities will result in a better estimate of the effect of the agent's time commitments on that particular alarm. The benefits gained from using such knowledge in determining $\Delta_{tc \rightarrow} t$ are more likely to outweigh the small increase in the computational requirements of the alarm processing machinery.

5.4 Discussion

5.4.1 Related work

The reminding of suspended goals in the presence of opportunities is widely recognised as a requirement in the design of an agent that must make timely decisions with an incomplete and imperfect model of its environment (Schank & Abelson, 1977; Hayes-Roth & Hayes-Roth, 1979; Birnbaum, 1986; Hammond, 1989b). Birnbaum & Collins (1984) and Hammond (1989b) present two distinct views on the detection of an opportunity to satisfy a goal that is suspended because it does not fit into the agent's current on-going plan. Birnbaum & Collins (1984) argue that, to explain the form of opportunism exhibited in Freudian slips, the indexing of suspended goals in memory described by Birnbaum (1986, chapter 8) is not sufficient. To explain this phenomenon, a suspended goal is viewed as an independent processing entity that uses inference to recognise the existence of opportunities. However, the use of inference is not a computationally feasible solution to the recognition of opportunities to satisfy a suspended goal (Patalano et al., 1993). The route scheduling system, TRUCKER, and the errand planner, RUNNER, (Hammond, 1989b) suspend goals that cannot fit into the current on-going plan by indexing them in an associative memory. These same memory structures parse the agent's sensory data to generate its model of the environment, and as the elements of memory that are associated with the suspended goal are activated, the agent will be reminded of the goal. The indexing of goals in associative memory is one of the mechanisms discussed by Birnbaum (1986) and referred to as "predictive encoding" by Patalano et al. (1993) (also see Simina & Kolodner (1995)). Before a goal is suspended, a limited number of features, the existence if which are most likely to constitute opportunities, are selected and the goal indexed in associative memory on the basis of these predictions. The reminding of a suspended goal is therefore achieved through the normal inferential processes of an associative memory. The structure of memory is not considered in this thesis, but the selection of a limited number of potential opportunities which are encoded as opportunity demons has a similar function. The difficulty in this approach is to select indexes that are sufficiently general to enable the goal to be activated in a wide range of opportunistic situations, but sufficiently specific to minimise the number of inappropriate recurrences of the suspended goal and to minimise the deliberation required once the agent is reminded of the goal.

Birnbaum (1986, pp. 143–154) discusses this "mental notes" approach to the encoding of opportunity demons in detail, and identifies an important trade-off that should be considered when using such a mechanism. "If the conditions which constitute an opportunity to pursue some goal have been characterized too specifically — if, in other words, the goal has been indexed in terms of overly specific features — then there will be many cases in which it will not be aroused

even though an opportunity for its satisfaction is present. On the other hand, characterizing the opportunity to pursue a goal too abstractly — that is, indexing the goal in terms of highly abstract features — makes it more difficult to retrieve the goal in *any* circumstances: Such abstract features will tend to be more difficult and expensive to recognize, and so it is less likely that the processing necessary to do so will be carried in any given situation." This problem is addressed in the design of the alarm processing machinery by restricting the conditions that may be used to characterise an opportunity to preconditions of known actions. These conditions are more likely to occur in the domain because the agent may only use the actions available to it to achieve its active goals. If the agent is acting on a goal that is somewhat similar to that associated with an opportunity, the conditions that characterise that opportunity are more likely to occur. Furthermore, taking the action, instansiated in this particular circumstance, is itself the opportunity, and so the reasoning involved in considering this opportunity is minimised.

The important difference between the opportunistic reminding approach discussed by Birnbaum (1986, chapter 8) and Patalano et al. (1993) and the mechanisms that have been discussed in this chapter is in the conditions under which an agent is reminded of a suspended goal. After predictive encoding (Patalano et al., 1993), the agent will be reminded of a suspended goal if it is associated with the present state of the associative memory which may include a number of temporal indexes (Birnbaum, 1986). (Note, the implications of the use of temporal indexes are not discussed, something that is central to this thesis.) In contrast, a motivated agent that is presented with an opportunity to achieve a suspended goal will only be reminded of that goal if the existence of the opportunity causes the intensity of the associated alarm to exceed the threshold. An opportunity simply causes the alarm that encapsulates the relevant suspended goal to increase in intensity to maximum. If the agent is pursuing other, more important and urgent goals the threshold may be set above the maximum intensity of the alarm, and so the agent will not be reminded of the goal in the light of this opportunity. Furthermore, an opportunity is defined here as a *timely* set of advantageous circumstances, and so an agent will not respond to a situation that may allow an inactive goal to be achieved with the minimum of planning effort if the time now is before the delay time of that goal.

Pryor & Collins (1992b) use a system of indexes, or "reference features", as heuristics to focus the process of deciding whether some situation does in fact present an opportunity to satisfy a goal. The system PARETO (Planning and Acting in Realistic Environments by Thinking about Opportunities) uses these reference features to focus deliberation in determining whether a situation that has been triggered as possibly constituting an opportunity conflicts with other goals that the agent is pursuing. Inactive goals are indexed in some way in terms of sets of conditions that

may constitute opportunities to satisfy the goal. (Note that neither the choice of conditions that characterise an opportunity nor the mechanisms that bring the agent's attention to those conditions are considered.) If a goal is considered in the light of an opportunity, the reference features that characterise the objects and constraints associated with that opportunity are compared with the reference features of the agent's active goals. If there is a conflict between the features of the opportunity (e.g. the opportunity for a truck agent to pick up some heavy cinder blocks) and the features of the currently active goals (e.g. to travel over a rickety bridge) then the agent's reasoning is focussed on that conflict. For example, the agent may decide that the cinder blocks should not be picked up because this will mean the weight of the truck will exceed the limit for the rickety bridge. If there are no conflicting reference features, the opportunity is assumed to be valid (goals are assumed to be "essentially independent"). Reference features provide useful heuristic mechanisms for the agent to reason about opportunities once it is reminded of a suspended goal in the light of an opportunity.

Once goals are to be achieved in a timely manner, opportunism is not sufficient for an agent to present appropriate behaviour in all circumstances. For this reason the influences of dangers and time commitments have also been investigated. The detection of dangers to the timely satisfaction of an inactive goal is similar to the detection of conflicts between actions in a non-linear plan. In such planning algorithms persistence constraints (or causal links) are posted between actions within the plan tying together a postcondition of one action to a precondition of another (Rich & Knight, 1991; Penberthy & Weld, 1992; Weld, 1994). Then, if a further action conflicts with this constraint, the conflict must be resolved; the use of persistence constraints provides a mechanism for recognising such conflicts. A particular alarm specification is associated with a number of "appropriateness conditions". For the goal to be successfully achieved in time, these conditions must hold between the interval t_{max} to t_{dl}. Therefore, if an action in the agent's plan endangers one of these appropriateness conditions, the planner should be made aware of the goal in time so that this conflict may be resolved. The combination of the effects of opportunities, dangers and time commitments provide a more complete set of influences on the timely reminding of suspended goals.

5.4.2 Conclusion

This chapter has described a novel approach to dealing with opportunities to satisfy inactive goals, and combined this influence with those of dangers and time commitments (issues that have not been considered in this context before). The agent may be reminded of goals in situations that present opportunities to satisfy those goals, situations in which the goal will take longer to satisfy (dangers), and the effect of decreased available time for the agent to satisfy its

goals. These influences on an alarm are heuristic: taking an opportunity may conflict with other active goals, or it may be quicker to achieve two goals together than one after the other, and hence reducing the time conflicts between two goals. Such limitations are an undesirable, but inevitable consequence of fast and cheap heuristics.

Chapter 6

The direction of planning attention

This chapter discusses the processes by which the focus of an agent's planning attention is directed towards its goals. The agent will be reminded of a suspended goal if and only if the intensity of the alarm associated with that goal exceeds some threshold; this is referred to as alarm triggering. If the threshold is increased, the intensity of an alarm must be higher for it to trigger and vice versa. It is through the manipulation of this threshold that the agent can control the goals that are considered at any one time (see section 6.1). Once an alarm triggers, the goal associated with that alarm is considered for further processing (see section 6.2). The primary purpose of these goal management mechanisms is to maintain a limited set of goals that are most appropriate for the agent to act on, leaving the planner to decide how to achieve them (see chapter 1).

6.1 Threshold

The mechanism by which a goal is brought to the attention of the agent is essentially simple. If the intensity of an alarm is greater than a certain current threshold value, the alarm triggers, and the goal encapsulated in that alarm is considered. Therefore, the threshold represents the sensitivity of the agent to considering its goals. If the threshold is high, the agent is relatively insensitive; i.e. the intensity of an alarm must be greater for it to exceed the threshold. If the threshold is low, the agent is relatively sensitive; i.e. alarms with lower intensities may trigger. By manipulating the threshold the agent can limit the number of alarms that trigger, and hence the number of goals that are considered for action.

6.1.1 The effects of a changing threshold

The success of the alarms mechanism relies heavily on an appropriate threshold level. If the threshold is set too high, the agent will tend to pursue its active goals to the detriment of other inactive goals that may be more important. The agent will only be reminded of other inactive goals when their associated alarm intensities exceed this exaggerated threshold, so it will tend to "leave everything to the last minute". If the agent has a number of important and urgent tasks

to pursue (i.e. if it is "very busy", cf. Beaudoin (1994)), then it may be appropriate to increase the threshold and hence consider fewer alternative goals. However, if the agent is not so busy it is less relevant to have such an insensitivity to other inactive goals. For example, it may be apt for a warehouse agent to delay considering the need to tidy the warehouse while it is busy preparing an urgent order, but not to delay considering the need to prepare an urgent order while it is tidying the warehouse. In the second case, the agent is not reminded of an urgent goal while pursuing another less important goal because the threshold is set inappropriately high.

The threshold value effects the rate at which alarms trigger. Typically, as the threshold decreases, there is an increase in the number of alarms that trigger, and goals will tend to be considered earlier. If the threshold is set too low, the agent will tend to be overly distracted by alternative goals. For example, it is generally not appropriate for the warehouse agent to be distracted by a goal to have tidied the warehouse while it is in the process of preparing an urgent order. A reduction in the threshold will tend to increase the number of goals that must be considered, increasing the computational resources consumed in this activity. So, an inappropriately low threshold can lead to the agent spending too much time thinking about alternative and inappropriate goals to pursue, and too little time pursuing its active and more appropriate tasks.

6.1.2 Threshold control

The function of the threshold is to control the rate at which alarms trigger, and hence the number of goals that are considered for activation in every cycle (see sections 3.1.5 and B.2.9). If the agent is busy pursuing important and urgent goals, then there will be little point in resources being directed towards considering alternatives unless the alternative is sufficiently relevant. If the agent has few goals to pursue, or is pursuing unimportant or non-urgent goals, then it can afford to invest more resources in considering alternative courses of action. Exactly how the threshold is controlled such that these requirements are met is the problem that will be addressed in the remainder of this section (i.e. subsections 6.1.2–6.1.5).

The threshold is set through the use of feedback control where the actual alarm triggering rate is controlled to the required rate by modifying the threshold (see figure 6.1). However, there is an important difference between this threshold control mechanism and those used in classical control theory. The dynamic control of the actual triggering rate to the required triggering rate through the manipulation of the threshold is limited by a threshold ceiling, above which the threshold cannot be set. This is discussed in more detail in subsections 6.1.4 and 6.1.5.

The required triggering rate (r_{reqd}) is compared to the actual triggering rate (r_{actl}). If the actual rate is lower than the required rate, the threshold is lowered. This lowering of the threshold tends to increase the alarm triggering rate, and hence compensate for the discrepancy. If the

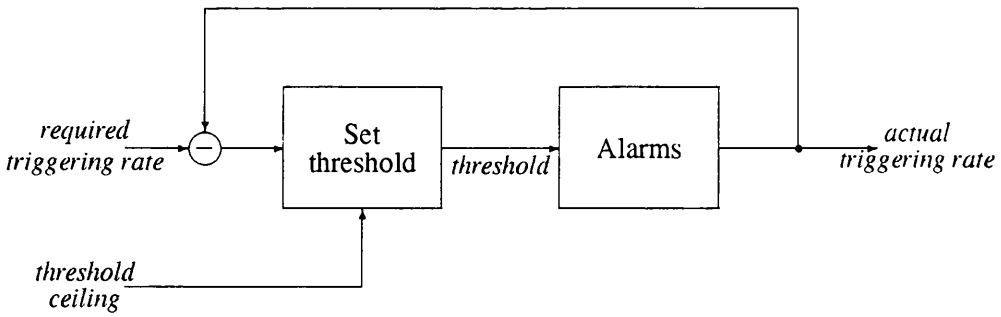


Figure 6.1: The control of alarm triggering rate.

actual rate is higher than the required rate, the threshold is raised. This tends to have the opposite effect and decrease the alarm triggering rate. A thermostat behaves in a similar way. The temperature of the room is equivalent to the actual triggering rate and the temperature set on the thermostat is equivalent to the required triggering rate. The thermostat switches on a heating system if the temperature of the room is less than that required and off if the room is too hot.

This use of feedback ensures that the rate of alarm triggering is effectively controlled independent from the total number of alarms influencing the agent. If the total number of alarms increases, the number of alarms that are above a particular threshold level at any one time will tend to increase. For example, assume that 10% of the total number of alarms are above some intensity I at any time. (Note that this analysis assumes that the agent has a steady flow of goals increasing in intensity; i.e. the deadlines of the agent's goals are evenly distributed.) Then, if the threshold is set to I and there are 20 alarms in total, an average of 2 alarms will trigger in every cycle. However, if the total number of alarms is doubled, the alarm triggering rate will double (in this case from 2 to 4 per cycle). So, the alarm triggering rate will increase as the total number of alarms increases if the threshold remains the same. However, if a triggering rate of 2 per cycle is required, there are a total of 40 alarms, and the threshold is initially at I, then it will be increased so that only around 5% of the alarms exceed it.

Furthermore, it cannot be assumed that the agent's goals are evenly distributed so that 2 alarms *do* trigger in every cycle, as in the above example. The times that alarms trigger depend on the times at which the agent wishes its goals to be satisfied. There may be some periods of time in which a large number of goals need to be satisfied, and some in which very few do. So, the concentration of high intensity alarms is not necessarily constant over time. If feedback is used to control the rate of alarm triggering, the threshold will tend to increase at times where there is a high concentration of high intensity alarms and decrease where the concentration is low. This will keep the rate of alarm triggering relatively constant, and at or near r_{reqd}. However,

it is possible that a burst of alarms trigger, temporarily swamping the system. Each goal will be considered, but typically, few activated (see section 6.2).

6.1.3 Determining the required triggering rate

Typically, as the number of active goals increases, the load on the cognitive resources of the agent increases. This means that the agent consumes a greater degree of its available time in pursuing its active goals. There is limited time available for planning and action, so as the cognitive load increases the rate at which alarms trigger should decrease, decreasing the number of goals considered for activation. If the triggering rate is low, the agent will be more focussed in its behaviour, and will be less likely to be distracted from its current goals by other tasks. Conversely, if cognitive load decreases, the triggering rate should increase, and the agent become more easily distracted.

In principle the triggering rate can be reduced to a lower limit of zero by increasing the threshold above the highest i_{max} for any alarm, preventing any distraction. However, in practice there is a limit to the level at which the threshold can be sensibly set. Even if the load on the agent is high, it may still consider a goal that is sufficiently important. For example, if the warehouse agent is busy (i.e. high cognitive load) with tidying the warehouse, it will still be reminded of more important goals such as the preparation of a customer's order. The procedure for determining the threshold ceiling is discussed in section 6.1.4. At the other extreme, a maximum triggering rate for any set of alarms can be achieved by setting the threshold to zero. However, this means that the agent has so few goals that they can all be considered without overloading the agent's planning and acting capabilities. In this situation, which will arise occasionally, there is no advantage in using the alarms mechanism. However, the machinery has low overheads, so there is little penalty associated with "wasting" resources on the threshold management system.

The cognitive load on an agent due to a particular set of active goals will depend on how long it takes to generate and execute plans. All other things being equal, the slower the agent is, the fewer goals it can satisfy in any given interval of time. However, cognitive load cannot be determined by simply counting the number of goals that are active because some goals may be far easier to plan for and achieve than others. For example, it may be easier for the warehouse agent to satisfy the goal to have recharged than to prepare a large order for a customer. Furthermore, the same goal may be easier to achieve in different circumstances. For example, if the agent takes an opportunity to satisfy a normally difficult goal, it will take far less time to plan for and often less time to achieve. So, it is not the nature of the goal, but the nature of the plan to achieve the goal that will give some indication of how much work the agent is committed to.

It is possible, and in some ways desirable for operators (or actions) within a plan to include

estimates of how long it will take to successfully execute the operator (Fox, 1993; Fox & Long, 1995). However, in abstraction-based planners, operators may not be directly executable. For example, an abstraction-based planner may select an operator such as eat at Joe's restaurant to satisfy the goal to have mitigated hunger. This operator requires refinement; e.g. the agent may need to decide how to get to Joe's and how the meal is to be paid for. It takes time for the agent to refine this abstract operator; time that is interleaved with execution in some hierarchical planners (e.g. FORBIN (Dean et al., 1988), PRS (Georgeff & Lansky, 1987), and HELP (Aylett et al., 1991)). Generally, planning takes far less time than execution, so in many cases planning time is an insignificant factor. For example, if I recognise that walking past Joe's is an opportunity to satisfy my goal of mitigating hunger and decide to take the opportunity, little planning effort will be involved.[1] However, the time taken to refine an abstract operator into executable actions may, in some cases, need to be taken into account to give a sufficiently accurate estimate of the work involved. Fox & Long (1995) proposes that an abstract operator in a hierarchical plan should include an estimate of how long it will take for (or is allocated by) the planner to refine it into primitive operators that may be executed. The combination of planning and execution time will give a good estimate of the time that the agent will invest in that operator. The more time the agent invests in its active goals, the less time the agent has available for the pursuit of other goals. Therefore, as the proportion of the time committed to active goals increases, the fewer alternatives the agent will be prepared to consider over the same time period. Conversely, if the agent's active goals require little effort, it will more readily consider other tasks.

A plan represents not only what actions the agent intends to perform and how much time must be invested in those actions, but also *when* the agent intends to perform them. For example, as part of its plan an agent may wish to attend a meeting in an hour, and reply to an electronic mail message within the next few minutes. The meeting may be more important than composing an Email, but the performance of the latter action is more urgent. If an agent is acting or intends to act in the immediate future, the time that is committed to this action will have a greater effect on the current cognitive load than an action to be performed an hour from now. If the agent is committed to performing an action an hour from now, this will have a greater effect on the cognitive load than one to be performed tomorrow, and so on. So, the time that is to be invested in attending the meeting will have an effect on the cognitive load from the time the action is inserted in the agent's plan until the time at which it is completed, but the magnitude of this effect depends on how soon the agent intends to act. As the time of the meeting approaches, the

[1] Little planning effort is involved because the definition of a opportunity given in section 5.1 is a timely set of advantageous circumstances that will allow the agent to achieve a goal with the minimum of planning effort.

effect of this intended action on the cognitive load increases.

Following this specification, the cognitive load, and hence the required triggering rate, can be estimated through a simple procedure. If the periods in which the agent is committed to the actions $\{a_1, a_2, \ldots, a_n\}$ are $\{(\Delta_1 t, t_1), (\Delta_2 t, t_2), \ldots, (\Delta_n t, t_n)\}$ where $\Delta_j t$ is the duration of interval j and t_j is its end, and $bf(t)$ is some biasing function, the required triggering rate may be estimated in the following way: (Note, the notation used in this chapter is summarised in appendix A.)

Definition 6.1 The required triggering rate is governed by the ratio of the area under the biasing function $bf(t)$ within the intervals in which the agent is committed to action, and the area under the function within the intervals in which it is free from commitment. (Note that the cognitive load on the planner and hence the required triggering rate is a discrete function, and hence is discretely computed.)

$$\frac{\sum_{j=1}^{n} \int_{(t_j \dot{-} \Delta_j t)}^{t_j} bf(t)}{\left(\int_0^{\infty} bf(t) - \sum_{j=1}^{n} \int_{(t_j \dot{-} \Delta_j t)}^{t_j} bf(t) \right)}$$

The function $bf(t)$ serves to bias the effect of a commitment interval $(\Delta_j t, t_j)$ on the cognitive load. So, if $bf(t)$ is some function that decays from some constant k at $t = t_{now}$ with some half life $\Delta_{1/2} t$, then the effect of a commitment interval $(\Delta_j t, t_j)$ on the cognitive load will decrease with both a decrease in the duration of that interval, $\Delta_j t$, and an increase in the time interval from t_{now} to $(t_j \dot{-} \Delta_j t)$. For example, the commitment intervals $\{(\Delta_1 t, t_1), (\Delta_2 t, t_2), (\Delta_3 t, t_3)\}$ of the actions $\{a_1, a_2, a_3\}$ shown in figure 6.2 all have different effects on the cognitive load. The performance of action a_1 is estimated to consume more time than a_2 and less time than a_3, but it is to be performed first and so has the greatest effect on the cognitive load. The actions a_2 and a_3 have similar influences on the cognitive load despite their different durations.

6.1.4 Determining the threshold ceiling

Ideally, the required triggering rate is determined on the basis of the cognitive load, and the threshold manipulated to control the actual triggering rate to this level. Then the agent will be less likely to consider other tasks when it is busy with its active goals, and more easily distracted if it is intending to use a small proportion of the time available. However, as mentioned above, there is a limit to the level at which the threshold can be sensibly set: the threshold ceiling. If the agent is busy, it should remain sensitive to considering goals that are sufficiently important relative to the goals it is busy with. It may not be possible for the threshold control mechanism to provide the required triggering rate because the threshold cannot be increased above the threshold ceiling to sufficiently reduce the triggering rate (see definition 6.3). (This is due to the agent's

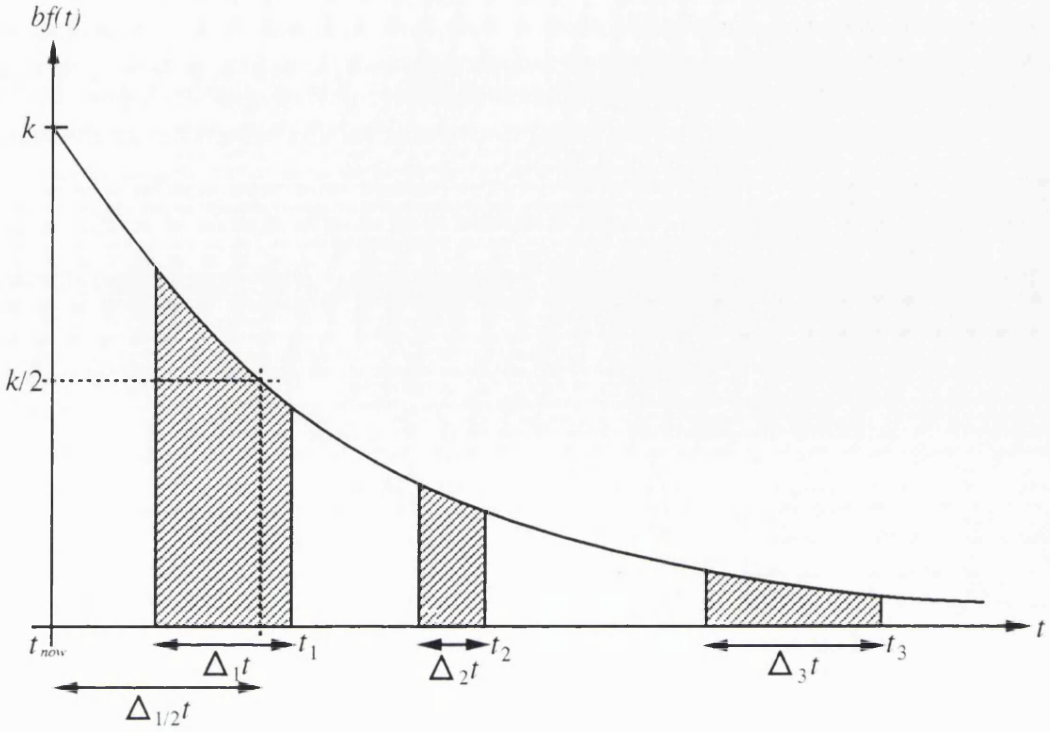


Figure 6.2: The effect of actions with different temporal characteristics on cognitive load.

finite capability for work and is discussed in more detail in chapter 7.) The agent will continue to consider and possibly activate goals that have an associated alarm intensity which exceeds the threshold ceiling, and activate those goals that are sufficiently important and urgent to pursue (see section 6.2.1). The activation of this new goal may mean that the agent can no longer achieve all its active goals within the time available. In such a situation the agent must decide between its active goals; this may involve deciding between a less important goal in which a certain amount of effort has been invested and a newly activated but relatively more important goal, for example.

Consider an agent that intends to attend a meeting an hour from now, and to compose an Email message in the next few minutes. The agent is busy with a relatively unimportant task at the moment (the Email message), so it should remain sensitive to goals that are more important than this one, but not necessarily as important as attending the meeting. Instead, if the agent intends to attend a meeting in the next few minutes, and compose and Email after the meeting, the threshold ceiling will be higher to reflect the fact that the more important goal is more urgent. So, it is both the importance of an action in the plan, and the time at which the agent intends to perform the action that influence the threshold ceiling.

Following this specification, the threshold ceiling is determined by taking the weighted av-

erage of the importances of the actions within the plan. The function $bf(t)$ is the same function that is used in the estimation of cognitive load (definition 6.1); it decays from some constant k at $t = t_{now}$ with some half life $\Delta_{1/2}t$. The use of this biasing function ensures that the threshold ceiling is influenced more by those actions that the agent intends to perform earlier.

Definition 6.2 The threshold ceiling, $\hat{\tau}$, is the weighted average of the importances of the actions in the agent's plan.

$$\hat{\tau} = \sum_{j=1}^{n} \left(\frac{\int_{(t_j - \Delta_j t)}^{t_j} bf(t)}{\sum_{k=1}^{n} \int_{(t_k - \Delta_k t)}^{t_k} bf(t)} \right) \times imp_{a_j}$$

6.1.5 Summary

The threshold is determined by evaluating the cognitive load on the agent and the threshold ceiling. The cognitive load is used to define the required triggering rate, but the exact relationship between the cognitive load and the required triggering rate will depend on the characteristics of the agents planning and acting capabilities. With the required triggering rate, r_{reqd}, the threshold ceiling, $\hat{\tau}$, the rate at which alarms are triggering, r_{actl}, and the current threshold, τ, the updated threshold can be determined in the following way:

Definition 6.3 The updated threshold, τ', is reduced by $\delta\tau$ if the actual triggering rate exceeds $(r_{reqd} + \delta r)$ and is increased by the same amount, up to some threshold ceiling $\hat{\tau}$, if r_{actl} is less than $(r_{reqd} - \delta r)$. If the actual triggering rate lies between $(r_{reqd} - \delta r)$ and $(r_{reqd} + \delta r)$, there is no change to the threshold level.

$$\tau' = \begin{cases} \tau - \delta\tau & \text{if } r_{actl} > (r_{reqd} + \delta r) \\ \tau + \delta\tau & \text{if } r_{actl} < (r_{reqd} - \delta r) \wedge \hat{\tau} > (\tau + \delta\tau) \\ \hat{\tau} & \text{if } r_{actl} < (r_{reqd} - \delta r) \wedge \hat{\tau} \leq (\tau + \delta\tau) \\ \tau & \text{otherwise} \end{cases}$$

The threshold is manipulated so that the triggering rate is controlled to within a limited range of the required triggering rate. The size of this range is determined by the variable δr. This variable is required to prevent the threshold from fluctuating by $\pm\delta\tau$ when the actual triggering rate is being controlled close to the required rate. The size of the step $\delta\tau$ by which the threshold changed must be small enough to enable accurate manipulation of the threshold, and large enough so that the threshold can be altered sufficiently quickly for the system to respond to changes in its cognitive loading. These parameters will depend on the particular implementation of the alarms mechanism.

6.2 Goal processing

Once the agent is reminded of a goal, it must make some decision about how to further process the goal. The agent may process such a goal in only three different ways. The goal can be activated and its associated alarm deleted, the alarm can be deleted without the goal being activated, or it can be left to be reconsidered later.

Goal activation and alarm deletion. If the agent decides that it is sensible to act on the goal, the goal will be activated. This means that the goal is added to the focus of planning attention (figure 6.3). Then, the means by which this goal is achieved is the responsibility of the planner. At this point, the alarm associated with the goal has served its purpose and is deleted.

Alarm deletion without goal activation. If a goal has not been activated and the agent no longer wishes it to be satisfied, then the alarm associated with that goal is deleted. Now, the agent will no longer be influenced by the alarm.

Alarm mitigation. The fact that the agent has considered the goal means that it is no longer relevant for consideration at the present time. So, if the alarm is not deleted, after the associated goal has been considered, the alarm is mitigated. The effect of this mitigation is to reduce the intensity of the alarm to a minimum, which subsequently increases to maximum again so that the agent reconsiders the goal at some appropriate time in the future. An additional effect of this action is that the opportunities list of the alarm is reset; the fact that the alarm has triggered in the presence of an opportunity means that the opportunity has been considered (see section 6.2.3).

Notice that the planning and control and the goal management mechanisms are independent. The goal management mechanisms monitor the planner and the planner uses the focus of planning attention to direct its behaviour, deleting goals when appropriate (see figure 6.3).

6.2.1 Goal activation and alarm deletion

The agent will attempt to achieve a goal only if the goal is activated. On activation, the achievement of the goal is the sole responsibility of the planner. So, if the agent decides to activate the goal, its alarm is deleted. When an alarm triggers, the associated goal will be activated if it is to the advantage of the agent to achieve that goal in the present situation. In deciding whether or not it is to its advantage to activate a goal, the agent will be guided by the following principles:

1. The agent will attempt to achieve as many goals as it can in the time available. The more goals that the agent can achieve, the greater the benefit the agent can gain from its activi-

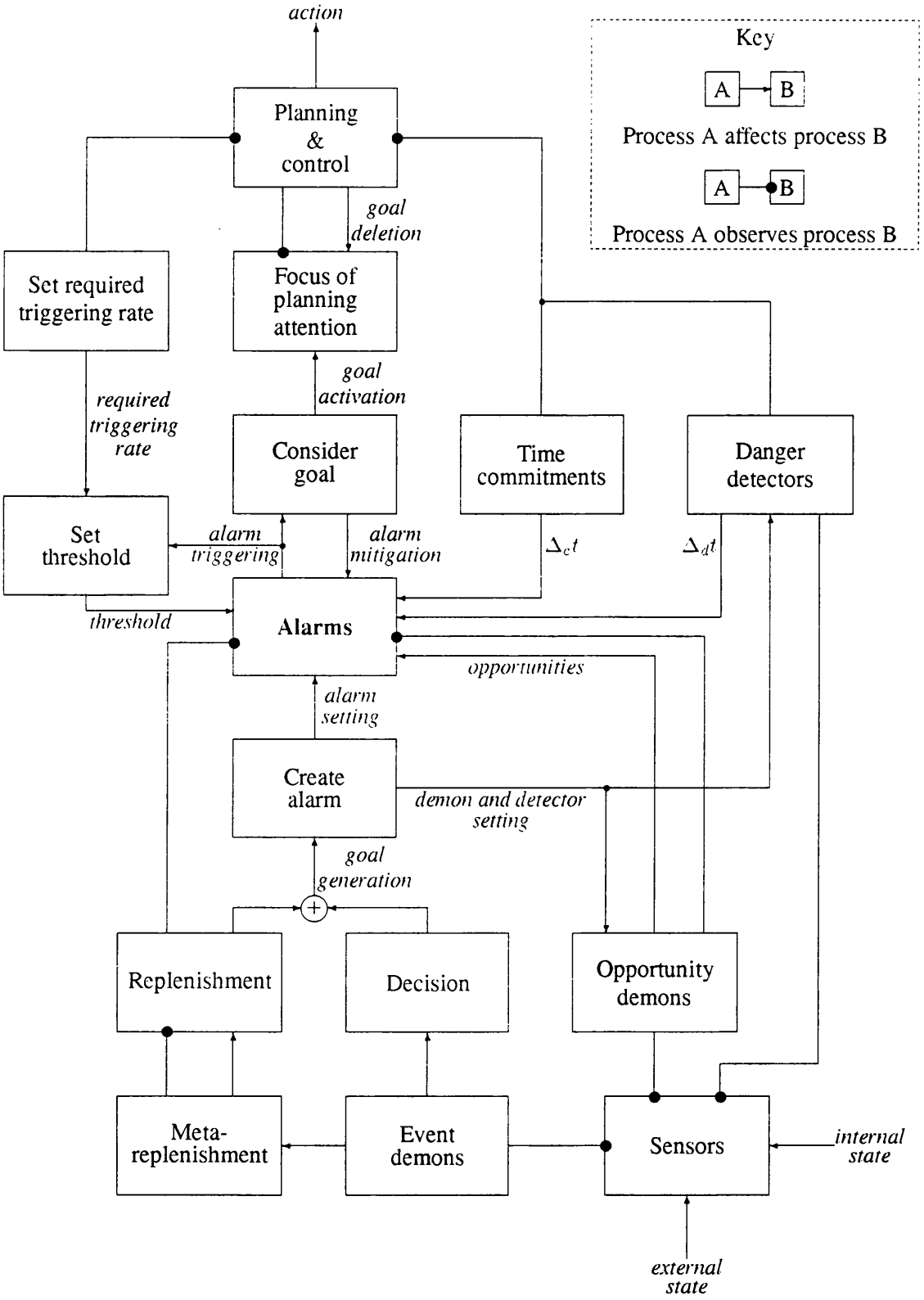


Figure 6.3: An architecture for goal generation and management.

ties.

2. The agent will only activate a goal if it can be achieved within a reasonable period of time. This serves the primary function of the goal management mechanism: i.e. to limit the agent's planning attention to those goals that are relevant, avoiding unnecessary reasoning.

3. The agent will use the time available in the pursuit of its most important goals. The greater the importance of a goal achieved, the greater the benefit the agent will gain from its activities. (This scheduling problem is being solved heuristically; given time, and goals with varying importance and time requirements, the best schedule, determined on the basis of some utility function, will be NP-hard to compute.)

Principle 1 ensures that if the agent can act, it will. So, the agent will not delay the pursuit of a goal if it is possible to do so, unless there are other tasks that are more relevant to pursue now. Principle 2 concerns the limiting of the number of goals that are visible to the planner. Consider a goal that is estimated to take about ten minutes to achieve. If there is insufficient time within the next hour or so for the goal to be achieved, then the goal will not be activated. The associated alarm will be mitigated to be reconsidered later (see section 6.2.3). The goal is only activated if there is time available for it to be achieved within a reasonable time period *relative to the time required to achieve the goal.* So, if the goal is estimated to take a couple of minutes, it will be activated if there is sufficient time available in the next half hour, and if it estimated to take a couple of hours, there must be sufficient time today, and so on.

Consider an agent that conforms only to principles 1 and 2. If such an agent is managing a warehouse, it may have planned a busy schedule of tidying and restocking the warehouse leaving little free time for other goals. Now suppose that the agent is reminded of an order that it had agreed to prepare for a reliable customer due to an alarm exceeding the threshold. This goal will be considered for activation. Following principle 1, the agent has planned to use its available time to maximum benefit in the pursuit of its currently active goals. However, there is now insufficient time available (principle 2) for the agent to achieve the goal under consideration as well as its active goals. The goal will not be activated. Therefore, an agent that conforms only to principles 1 and 2 will treat its goals on a "first come, first served" basis. If the agent also conforms to principle 3, goals that the agent considers more important will be given priority.[2]

[2]This best-first heuristic makes sense if it is possible for time estimates to be flawed. If the agent chooses to act on a less important goal first at this stage, there may be insufficient time left for more important goals later if its predictions prove to be inaccurate.

Consider an agent that conforms to principles 1–3 is managing a warehouse. following principle 1 it may plan a busy schedule of tidying and restocking activities. Suppose the agent is again reminded of the relatively important goal of meeting the customer's order. Now. following principles 2 and 3. if the goal under consideration can be achieved in a reasonable period of time along with those planned activities that have an equivalent or greater importance to the agent. then the goal is activated. In this case the goal is activated because meeting the customer's order is more important than tidying and restocking the warehouse. Therefore, the agent will continue to use the time available in the pursuit of its active goals. but if it is reminded of a sufficiently important alternative this goal will also be added to the focus of planning attention.

If the alarm triggers on the basis of an opportunity. the goal is processed in the same way. with one exception. The estimate of how long it will take to satisfy the goal will have changed. Typically. if the agent is presented with an opportunity to satisfy a goal and decides to take the opportunity. it will take less time to achieve the goal. An opportunity to achieve a goal is defined (definition 5.1) as the recognition of an action that can be performed in the present state which satisfies the goal as one of its postconditions. So. if the alarm triggers in the presence of an opportunity. the time estimate for the associated goal is the length of time that the the action is estimated to take. If there is no opportunity. the time estimate for the goal is $\Delta_a t$. where $\Delta_a t$ is the estimate of how long it will take to satisfy the goal made at the time that the goal is generated and encapsulated in an alarm (section 4.3).

6.2.2 Alarm deletion without goal activation

This section considers the conditions under which alarms are deleted without the associated goal being activated. (Note that this will occur *after* the alarm has triggered; alarms are not deleted automatically.) In principle. a goal will be deleted if: (1) the goal has been satisfied; (2) the goal cannot be satisfied; or (3) the agent no longer wishes to satisfy the goal (cf. Cohen & Levesque (1990)). Intuitively, the goal can only be satisfied if the agent activates the goal and acts on it. However. it is possible for a goal to be satisfied by another agent, or for the agent to have satisfied the goal through its own actions without being aware of it. So. if a goal under consideration has been achieved, the alarm associated with that goal will be deleted without the goal being activated.

Furthermore. it may not be necessary for the agent to fail to find a plan that will achieve a goal for the agent to know that a goal cannot be achieved. Consider a warehouse agent that receives a request from a potential customer for an order to be satisfied at some specific time. If. in response to this request. the agent generates a goal to have prepared the order. then the agent will set an alarm so that it is reminded of the goal at an appropriate time before the customer

is scheduled to arrive. However, if the agent is busy with other tasks and is not reminded of this goal until a significant time after the deadline, it will no longer be possible for the agent to achieve the goal; the customer will have left. This is an example of a goal with a hard deadline; i.e. if the goal is not achieved before the deadline there is no possibility that the agent will gain anything from continuing to act on that goal. In contrast, consider the goal to have ordered new stocks of widgets. If the agent is not reminded of this goal before the deadline, it will still be important for the agent to achieve the goal. This deadline is a time before which the agent wishes to achieve the goal, not a time before the agent must achieve the goal. This type of deadline is referred to as a soft deadline (see section 4.1). However, this does simplify the problem a great deal. It may be possible that the agent can gain something from achieving the goal after the deadline, but this will decrease as time passes. If the agent knows that there is some point in time after which nothing can be gained from achieving a goal under consideration and that time has passed, the alarm associated with that goal will be deleted without the goal being activated.

The third condition under which an agent will delete an alarm without activating the goal is if that goal is no longer relevant to the agent. But what constitutes the relevance of a goal? Consider a warehouse agent that receives a request from a potential customer for an order to be satisfied at some specified time. In response to this request, the agent must decide whether to accept or reject the order. Now suppose that the warehouse agent makes this decision on the basis of two criteria:

1. The customer must be reliable, or at least not known to be unreliable. In other words, the agent expects the customer to collect the order, and pay for it on time.

2. The order must be profitable for the agent. For example, if the prices that the agent charges for the commodities in the warehouse are low, orders may need to be large for the agent to make a sufficient profit.

So, the agent will generate a goal to have satisfied the customer's order if the customer requires the order to be satisfied, satisfying the goal is profitable for the agent, and the customer is reliable. However, the agent will generate a goal to have rejected the request if the customer requires the order to be satisfied, and either the order is unprofitable, or the agent considers the customer to be unreliable (see figure 6.4). The decisions made by the agent in response to the request, even in this simple example, are critical in determining the goal that will be generated. The initiating change in the domain and the decisions made in this process are the *reasons* for the existence of the goal that is generated, whatever that goal is.

For example, if the agent generates a goal to satisfy the order as requested, the reasons for

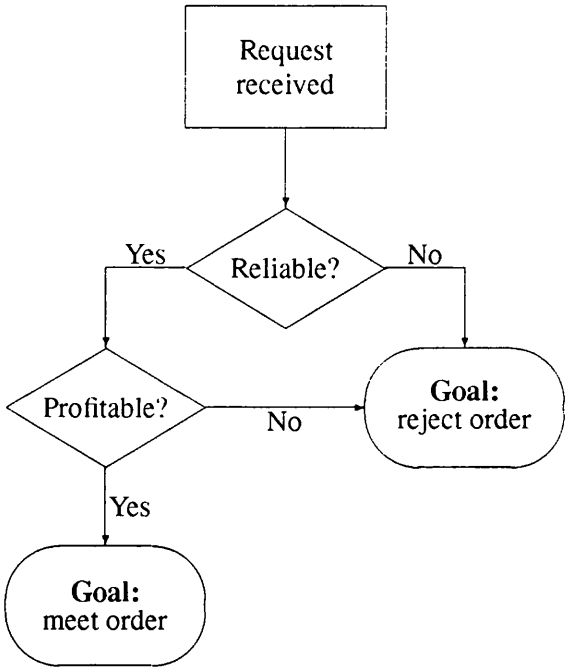


Figure 6.4: An example decision process in the generation of a goal.

the existence of that goal are: (1) the customer requires the order; (2) the order is profitable (this could be a decision made after negotiation (Müller, 1996; Norman et al., 1996; Rosenschein & Zlotkin, 1994); and (3) the customer is reliable. Every time a goal is generated through decision, the agent will evaluate the event that has triggered this process, decide whether to generate a goal, and if so, what goal and when it must be satisfied (see section 4.4). For goals that must be achieved periodically or due to some timetable, a replenishment process is adopted for the automatic generation of goals. The generation of goals in this way by-passes this decision process, and hence serves to avoid unnecessary reasoning (see section 4.5). For example, a reliable customer who requires the same order every Tuesday may negotiate a contract with the warehouse agent. Under this contract, the agent agrees to prepare orders for collection on Tuesday and the customer agrees to certain payment conditions. This contract removes the need for the customer to request an order, and for the agent to decide whether or not to adopt the order every week; the warehouse agent assumes that the customer requires the order. In this case, the reasons for the replenishment of this goal are: (1) the customer requires the regular order; and (2) the customer does not break the conditions of the contract. These may include the conditions that the customer collects every order, and pays by a certain time each week, for example.

Between the time that a goal is generated, either through decision or replenishment, and the associated alarm triggers, the situation may have changed. The customer may have failed to

collect a different order, the agent may receive a message cancelling the order, or payment may not be received, for example. If any one of the reasons for the generation of the goal no longer hold, then the agent may no longer wish the goal to be satisfied. For example, if the agent has the goal to satisfy an order placed by a customer because the customer was considered reliable (among other reasons), and that customer has subsequently proved to be unreliable, then the agent will no longer wish to achieve this goal. This does not mean that the agent will never satisfy this or any subsequent orders from that customer, but a new agreement must be reached with that customer.

When an alarm triggers, the reasons for the existence of the goal associated with that alarm are checked. Then, the alarm will be deleted without the goal being activated if any one of the reasons for the existence of the goal no longer holds. This action of deleting an alarm may cause the agent to generate other goals. For example, if the alarm encapsulating a goal to have satisfied a customer's order is deleted because the customer is no longer considered reliable, the agent will generate a goal to inform the customer that the order has been rejected. The customer is then free to commence negotiation with the agent if the order is still required.

6.2.3 Alarm mitigation

The intensity of an alarm reflects the relevance of the goal encapsulated in that alarm for consideration by the agent. Once an alarm has triggered and the associated goal has been considered, the goal is no longer relevant for consideration at the present time. So, if the alarm is not deleted, the fact that the goal has been considered causes the intensity of the alarm to be mitigated. Then, as the time since the goal was last considered increases, the goal becomes more relevant for reconsideration. For example, if the warehouse agent considered the goal to have checked the order book and decided not to activate the goal at that time, then as time passes, the intensity of the alarm (i.e. the relevance of the goal for consideration) will again approach i_{max}. So, even though the agent decided not to activate the goal, it will be forced to reconsider the goal at an appropriate time later unless the threshold has subsequently increased above i_{max}. The mitigation of an alarm is modelled by constructing another function of intensity that is added to the alarm function, which along with the effects of opportunities, dangers and time commitments, gives the intensity of the alarm. If the intensity of the alarm function at the time of triggering is $f(t_{now})$ and the time at which the agent wishes the mitigation to stop is t_{off}, then the mitigation function is given by definition 6.4. The effect of this mitigation function on an alarm function is illustrated in figure 6.5. (Note, the notation used in this chapter is summarised in appendix A.)

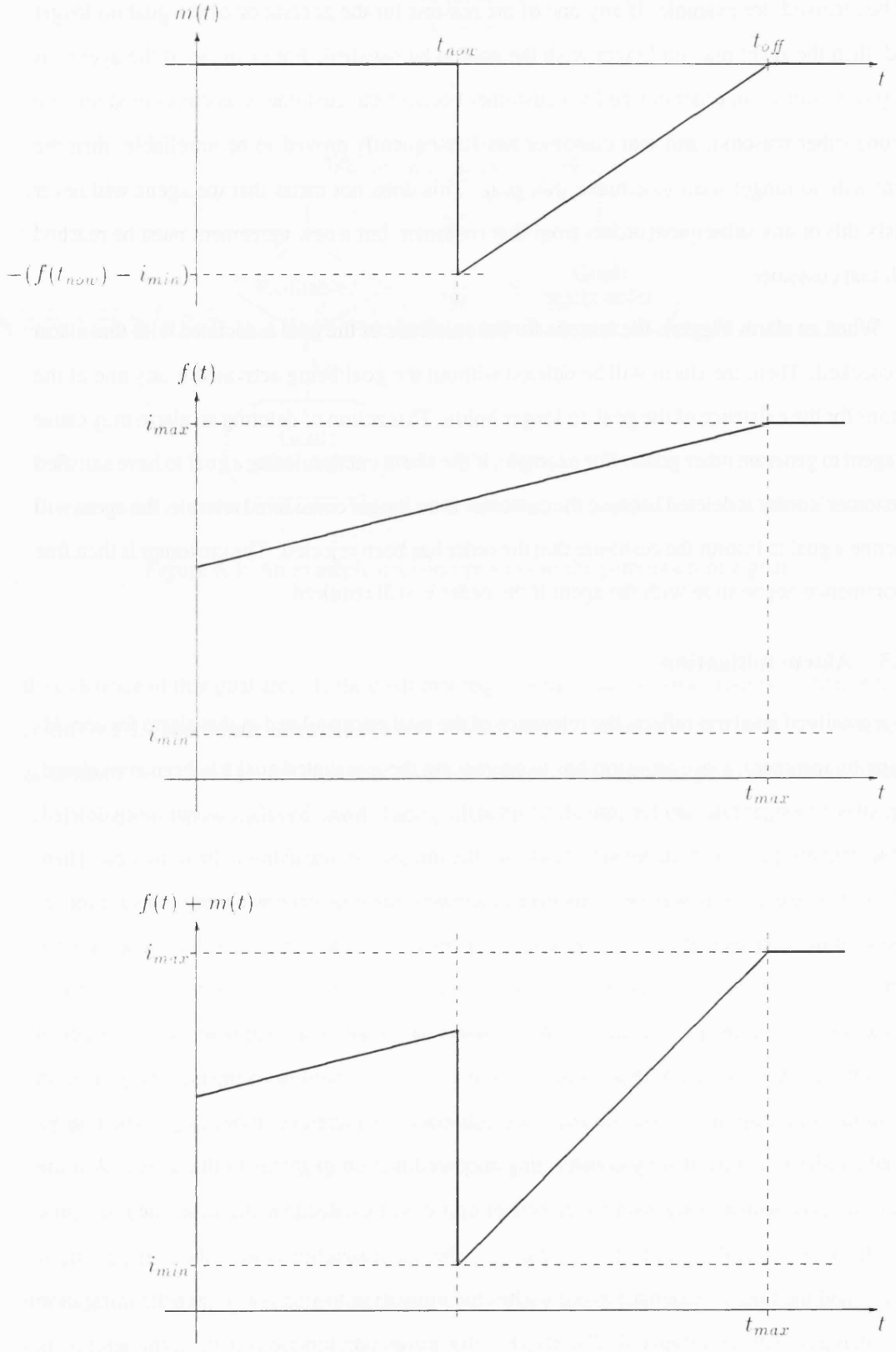


Figure 6.5: The effect of mitigation on an alarm function.

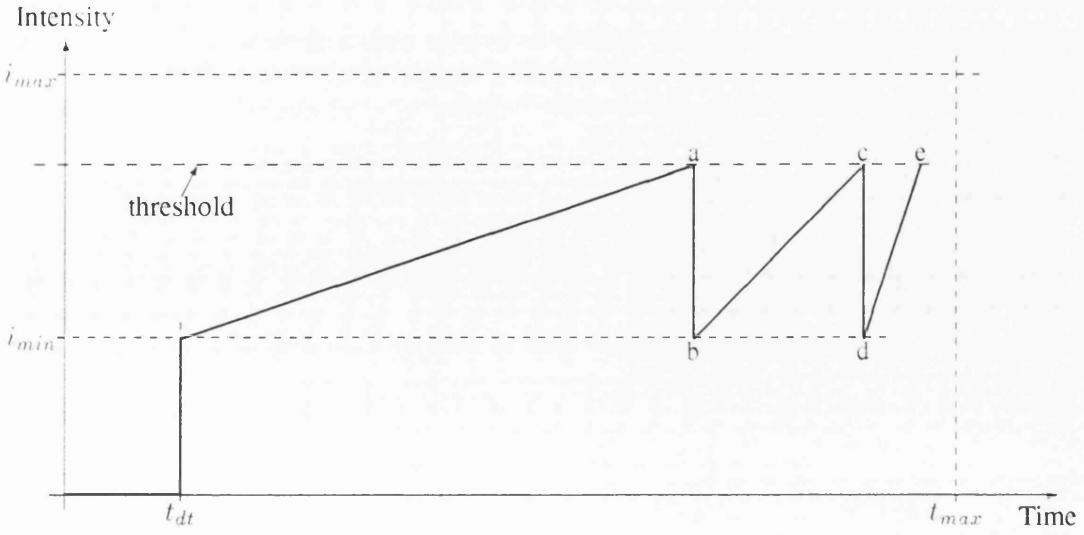


Figure 6.6: The effect of repeated mitigations on an alarm (1).

Definition 6.4 The mitigation function is zero before t_{now} and after t_{off}, and increases from $-(f(t_{now} - i_{min})$ at t_{now} to zero at t_{off}.

$$m(t) = \begin{cases} 0 & \text{if } before(t, t_{now}) \vee \neg before(t, t_{off}) \\ -(f(t_{now}) - i_{min}) \times \frac{t_{off} - t}{t_{off} - t_{now}} & \textbf{otherwise} \end{cases}$$

where the ratio $\frac{t_{off}-t}{t_{off}-t_{now}}$ is expressed as a real number, and t_{off} decreases to $\Delta_{min}t$ as t_{max} approaches and increases to $\Delta_{max}t$ after t_{max}.

Suppose that the warehouse agent has the goal to check the order book, the alarm encapsulating this goal has triggered, and the agent has decided not to activate the goal at this time. This decision does not change the fact that the agent must commence action before t_{max} to ensure that the goal is satisfied in time. So, the agent will set t_{off} to t_{max} so that it is forced to reconsider the goal before this critical time unless the threshold has subsequently increased above i_{max} (this is the case in figure 6.5). In fact, as t_{max} approaches, the agent will tend to consider activating the goal more frequently if the goal is not activated. The effect of the agent considering the goal for activation, deciding not to, and mitigating the alarm function is illustrated in figure 6.6.[3] This alarm function increases to i_{min} at t_{dt} and then continues to rise linearly towards i_{max}. When the intensity of the alarm exceeds the threshold (point a), the goal encapsulated in the alarm is considered and a decision is made about the further processing of that goal. Suppose that the

[3]This is a simplified picture of how the alarm will behave. The threshold is assumed to be constant and below t_{max}, and no changes in the intensity of the alarm due to opportunities, dangers or time commitments are considered.

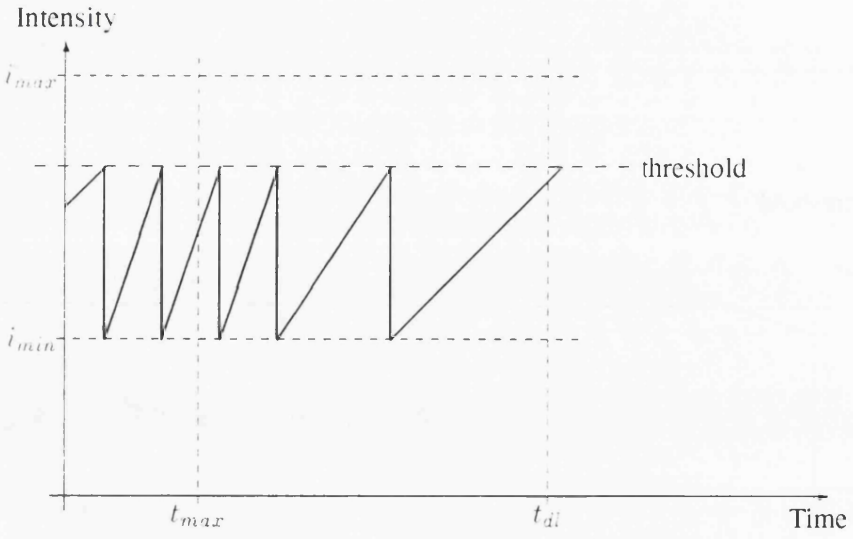


Figure 6.7: The effect of repeated mitigations on an alarm (2).

goal is to have checked the order book, and at this point in time the agent decides not to activate the goal. The agent has considered the goal, and so the alarm is mitigated to i_{min} (point b). The intensity of the alarm increases back towards i_{max} until it again exceeds the threshold and the goal is reconsidered (point c). At this point the agent again decides not to activate the goal, again the alarm is mitigated (point d), it increases over time, and exceeds the threshold a third time (point e). So, the goal is considered more frequently as t_{max} approaches if it has not yet been activated. (The time t_{max} is a prediction of the last point in time that the agent must commence action for it to successfully achieve the goal before the deadline.)

Suppose that the goal is not activated before t_{max}. If the goal is activated at any time after t_{max}, the agent may not have sufficient time to satisfy the goal before its deadline. Furthermore, as time passes the chances that the goal can be satisfied before the deadline if the goal is activated will decrease. As the time now approaches t_{max}, the frequency that the alarm triggers increases to a maximum.[4] Then as the time now proceeds beyond t_{max}, the triggering frequency will decrease to some minimum (see figure 6.7). So, the frequency at which the alarm will trigger and the agent considers the goal (assuming that the threshold is constant and below i_{max}) is related to the modulus of the difference between the t_{now} and t_{max}. If the goal is never activated, the agent will continue to be reminded of the goal around some minimum frequency until the agent decides to delete the alarm. Typically, the goal will be activated at some stage and the planner

[4] If a maximum triggering frequency (i.e. a minimum duration of mitigation) is not defined, around t_{max} the goal will be considered every cycle (i.e. every time the alarms are evaluated); this is an impractical solution.

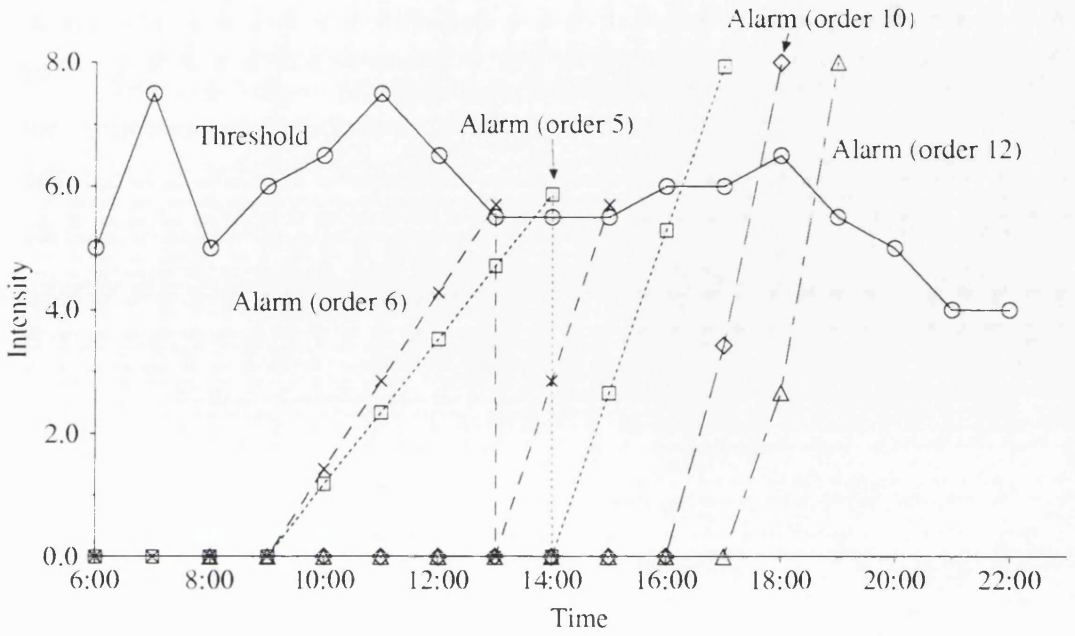


Figure 6.8: The behaviour of mitigated alarms in the warehouse simulation.

will attempt to achieve it.

Definition 6.5 The difference between t_{now} and the time at which mitigation should stop, t_{off}, decreases to $\Delta_{min}t$ as t_{max} approaches, and increases to $\Delta_{max}t$ after t_{max}.

$$t_{off} = \begin{cases} t_{now} + \Delta_{min}t & \textbf{if } before(t_{now}, t_{max}) \wedge |t_{max} - t_{now}| \leq \Delta_{min}t \\ t_{now} + \Delta_{max}t & \textbf{if } before(t_{max}t_{now}) \wedge |t_{max} - t_{now}| \geq \Delta_{max}t \\ t_{now} + |t_{max} - t_{now}| & \textbf{otherwise} \end{cases}$$

where $|t_1 - t_2|$ is $(t_1 - t_2)$ if $t_1 \geq t_2$ and $(t_2 - t_1)$ otherwise.

In addition to the mitigation of an alarm associated with a goal that is not activated after consideration, any opportunities associated with that alarm no longer influence the intensity of the alarm. The mechanism by which this effect is achieved is for an opportunity demon to insert a detected opportunity into the alarm structure. Then once the opportunity ceases to exist, the demon removes the opportunity from the alarm structure (section 5.1.3). However, if the alarm triggers and the associated goal is not activated, the opportunity is removed after the consideration process.

Example from simulation

Figure 6.8 (produced using data from the simulation of the alarm processing machinery, see appendix C) illustrates the behaviour of four alarms encapsulating goals to have prepared orders for customers. The alarms associated with orders 5 and 6 are both mitigated before they are

activated. The alarm encapsulating the goal to have prepared order 6 increases with a steeper gradient than order 5 and exceeds the threshold at 1pm. However, at this time the agent is not prepared to act on the goal at this time, and so the alarm is mitigated. This again increases, but with a steeper gradient to exceed the threshold again at 3pm. This time the goal is activated. The alarm encapsulating the goal to have prepared order 5 behaves in a similar way. It exceeds the threshold for the first time at 2pm, is mitigated, increases again at a faster rate and is activated at 5pm. (Note that the alarm function actually crosses the threshold at around 4:15pm, but it is only considered at 5pm due to the low frequency at which the alarms are sampled.

6.3 Discussion

6.3.1 Related work

The use of thresholding mechanisms to direct the attention of an agent to salient goals or other types of control states is not entirely new. Although, the mechanism which exploits its use as described here is a novel research contribution. Using the filter penetration theory (Sloman & Croucher, 1981; Sloman, 1992) as a starting point, Beaudoin (1994, pp. 80–90) proposes a filter that serves to limit attention by passing only those goals that are sufficiently insistent to exceed some threshold. The level of this threshold is primarily dependent on the "busyness" of the situation. However, Beaudoin (1994) distinguishes between a "busy" and an "acute" situation: an acute situation being one in which "a split second distraction could have drastic consequences" (p. 85). The "busyness" of a situation is discussed in terms of: (1) the importance of current (or active) goals; (2) the urgency of active goals; and (3) the total number of goals, both active and inactive. However, no concrete specification is provided. In the alarm processing machinery, the alarm triggering threshold depends only on the importance and urgency of the actions that the agent intends to perform; i.e. it depends only on those goals that consume cognitive resources. More importantly, this chapter contributes by presenting an operational specification of this filtering mechanism, indicating how a practical implemention may be produced.

The processes involved in the recollection of an agent's suspended goals between their generation and enactment is also of interest to psychologists interested in the study of "prospective memory" (Brandimonte et al., 1996; Harris & Wilkins, 1982; Reason, 1984; Ellis & Nimmo-Smith, 1993; Ellis, 1994). The alarm processing machinery presented here is not intended to model the behaviour of human prospective memory, but it is interesting to compare the behaviour of these processes to the behaviour reported in the psychological literature. For example, Harris & Wilkins (1982) have reported that human subjects tend to recall their goals with increasing frequency as the time to act approaches. In the alarm processing machinery, if a

goal is not activated once the alarm encapsulating that goal has triggered, the alarm is mitigated. This mitigation causes the intensity of the alarm to increase to maximum before t_{max} (i.e. the time at which the agent has predicted that it should act). Therefore, if the goal is not activated when the associated alarm triggers (i.e. if the agent does not decide to act on the goal), the agent will tend to be reminded of the goal more frequently as t_{max} approaches. The diary study presented by Ellis & Nimmo-Smith (1993) indicates that the likelihood of a subject reporting the recollection of a previously formulated intention tends to increase when that subject is involved with tasks that require less attention (see also Kvavilashvili (1987)). In the alarm processing machinery, the likelihood of a goal being considered for activation (i.e. being recalled) will tend to increase as the threshold level decreases. This threshold will tend to decrease as the load on the agent's reasoning resources decreases. This cognitive load tends to decrease as the number of urgent and important actions decreases. However, Ellis & Nimmo-Smith (1993) also indicate that subjects tend to be more likely to be reminded of intentions with shorter "performance intervals" (i.e. the interval of time within which it is sensible for the agent to act on the intention). This behaviour is not exhibited by the alarm processing machinery. However, there may be some correlation between the subjective importance attributed to intentions and the size of their performance intervals. An intention with a short performance interval may be considered more importance purely on this basis; possibly because the agent is at greater risk of missing this interval. This is pure conjecture, but in the alarm processing machinery an alarm encapsulating a goal with higher importance will have a greater chance of being considered.

6.3.2 Conclusion

The alarm triggering and goal consideration processes together act as a double filter. An alarm will trigger if its intensity exceeds some current threshold level: a coarse heuristic. Then, the associated goal is checked to determine whether it is worth acting on at the present time; this is decision is made on the basis of a best-first heuristic strategy. If the goal is activated, it becomes the responsibility of the planning machinery. Furthermore, these goal filtering processes are modulated by the load on the agent's cognitive resources. A primary contribution of this thesis is that this, and the previous two chapters provide an operational account of these goal management processes. Only through providing such a concrete specification can such a system be critically examined and improved upon.

Chapter 7

Summary and Discussion

7.1 Introduction

Chapter 3 gave a high level overview of the motivated agent architecture, and chapters 4, 5 and 6 have described the alarm processing solution to the problem of directing and limiting planning attention in the three stages: alarm generation, modification, triggering and goal consideration respectively. This chapter provides a summary of the previous four chapters and a detailed discussion of the alarm processing machinery. Section 7.2 is a dense, functional specification of the alarm processing machinery in fairly precise terms. It brings together the most important definitions from chapters 4–6, and explains in more detail how they are combined. (It may be useful for the reader to refer to Appendix B while reading this section; this appendix presents the implementation of a prototype motivated agent following this specification.) In section 7.3 the function of the planner in a motivated agent is discussed in broad terms. Finally, section 7.4 examines the alarm processing machinery from a more critical perspective, and a number of open issues concerning planner and goal management interaction are identified for future research.

7.2 Summary of the alarms heuristic

An alarm is a structure that associates a goal with some heuristic measure of intensity. This is essentially an indexing scheme for the suspension and subsequent reminding of goals (cf. Birnbaum (1986), Ellis & Nimmo-Smith (1993), Ellis (1994), Patalano et al. (1993), Hammond (1989b), and Simina & Kolodner (1995)). The alarm processing machinery presented in the previous three chapters and summarised here serves to manage the generation and activation of goals within the motivated agent architecture presented in chapter 3.

An agent will generate a goal either because it has decided to do so in response to some detected event in the domain, or automatically (i.e. goal replenishment) in response to a similar goal being deleted (chapter 4). A generated goal is then encapsulated within an alarm structure. The intensity of the alarm can then be heuristically evaluated from the information contained in

this alarm structure. When the goal is in this state, it is not visible to the planning and reasoning processes of the agent, and so consumes a small amount of computational resources (refer to section 7.4.4 for a discussion). As the intensity of an alarm increases, the agent is more likely to be reminded of the goal encapsulated in that alarm structure, and possibly act to satisfy it. If an alarm does trigger, i.e. the intensity of the alarm exceeds some current threshold value, the associated goal is considered for activation. If the goal is then activated, the alarm is deleted and the further processing of that goal becomes the responsibility of the planner. If the agent decides that the goal is no longer required, the alarm is deleted without the goal being activated. If the alarm is not deleted, it is mitigated. This mitigation effectively re-suspends the goal until the alarm again triggers some time later and the goal is again considered for activation. The following sections serve to summarise the various aspects of this alarm processing machinery.

7.2.1 Alarm generation

An alarm is generated on the basis of an initial prediction of when the goal is relevant for consideration and its importance. This information is necessary for the specification of an alarm function, which, if there are no other influences on the intensity of an alarm, and if the threshold level remains below i_{max}, causes the agent to be reminded of the goal at an appropriate time before t_{dl}, where $t_{max} = t_{dl} - \Delta_a t$. The information consists of:

1. The time before which the agent should not act on the goal, the delay time, t_{dt};
2. The time before which the agent wishes the goal to be achieved, the deadline, t_{dl};
3. An estimate of the time that it will take to satisfy the goal, $\Delta_a t$;[1]
4. The minimum intensity of the alarm after the delay time, i_{min}; and
5. The maximum intensity of the alarm, i_{max}, which is the importance that the agent gives to the satisfaction of the goal.

If a goal is unique or must be achieved at unpredictable times, the variables t_{dt}, t_{dl}, $\Delta_a t$, i_{min} and i_{max} are determined through prediction at the time that the goal is generated and an alarm set (alarm generation through decision, section 4.4). However, if the agent wishes to pursue a goal cyclically or a particular times of the day or week, for example, then the variables are determined through a pre-defined alarm specification (alarm replenishment, section 4.5). (Note, the notation used in this chapter is summarised in appendix A.)

[1] It is possible that this estimate may be refined, but typically only once a plan of action to satisfy the goal is being generated; i.e. once the goal is activated and a plan generated.

Definition 7.1 The alarm function of an alarm α is zero before t_{dt}, increases from i_{min} to i_{max} from t_{dt} to t_{max}, and then remains at i_{max}.

$$f(t) = \begin{cases} 0 & \text{if } before(t, t_{dt}) \\ i_{max} & \text{if } \neg before(t, t_{max}) \\ i_{min} + \left(\frac{t \ominus t_{dt}}{t_{max} \ominus t_{dt}} \times (i_{max} - i_{min})\right) & \text{otherwise} \end{cases}$$

where $t_{max} = t_{dl} \ominus \Delta_a t$ and the ratio $\frac{t \ominus t_{dt}}{t_{max} \ominus t_{dt}}$ is expressed as a real number.

7.2.2 Opportunities, dangers and time commitments

The transient effects of an uncertain and changing domain, i.e. opportunities, dangers and time commitments, may also affect the intensity of an alarm.

Opportunities

An action is an opportunity to satisfy a suspended goal if the action can be performed with the minimum of planning effort, the goal is worth achieving, the preconditions of the action hold in the domain, and the goal is a postcondition of that action. An opportunity will only be detected if the action is encoded as an opportunity demon at the time that the alarm is set.

Definition 7.2 If the goal is worth achieving (i.e. the intensity of the alarm is greater than zero), then if an action that was anticipated as a potential opportunity can be performed in the present state, it is an opportunity.

$$intensity(\alpha, t_{now}) > 0 \Rightarrow opportunities(\alpha) = \{a : anticipated(\alpha) \mid pre(a) \subseteq D\}$$

where D is the present state of the domain.

The effect of a detected opportunity is to increase the intensity of the alarm to i_{max}. This effect will remain either for the duration of the opportunity or if the alarm triggers and the goal is not activated. If the goal is not activated and the alarm is mitigated (see below), the effect of all opportunities is removed.

Dangers

The initial predictions that define the alarm function are only appropriate under a number of conditions: the appropriateness conditions of the alarm, $apps(\alpha)$. Suppose that an agent has a goal to attend a meeting at a particular location and a particular time, and predicts that it will take about half-an-hour to get to this meeting. However, suppose that this prediction is only appropriate if the agent is in Roboville. If the agent subsequently plans to be somewhere else around the time of the meeting, this intention constitutes a danger to the timely satisfaction of the goal; it will take more than half-an-hour to get to the meeting if it is not in Roboville. So, if an agent has a set of

appropriateness conditions $\{(at(\text{Agent}, \text{Roboville}), 1\ \text{Day}), (at(\text{Agent}, \text{Roboland}), 2\ \text{Days})\}$ and if $\neg at(\text{Agent}, \text{Roboville}) \wedge at(\text{Agent}, \text{Roboland})$ holds in the domain within a day before the meeting, this constitutes a danger to the timely satisfaction of the goal. Furthermore, if the agent plans to be outside Roboville, but in Roboland within two days before the meeting this also constitutes a danger to the goal. If there is a danger to the goal encapsulated in an alarm, the alarm will be shifted appropriately so that the alarm reaches maximum early enough for the agent to have enough time to achieve it.

Definition 7.3 If there exists a persistence constraint in the agent's plan that constrains a proposition p_{pcs}[2] to hold until t_{pcs}, p_{pcs} conflicts with the jth appropriateness condition $(p_j, \Delta_j t)$ of the alarm α, and the constraint holds during the interval $(t_{max} \ominus \Delta_j t)$ to t_{dl}, then the alarm should be shifted so that it reaches maximum at $(end(a) \ominus duration(a) \ominus \Delta_a t)$. (This p_{pcs} is an *intended* consequence of the planned action, a.)

$$(a, p_{pcs}, t_{pcs}) \in pcs(\pi) \wedge (p_j, \Delta_j t) \in apps(\alpha) \wedge conflict(p_j, p_{pcs}) \wedge$$
$$\neg(before(t_{pcs}, (t_{max} \ominus \Delta_j t)) \vee before(t_{dl}, (end(a) \ominus duration(a))))$$
$$\Rightarrow \Delta_{a\_} t = \begin{cases} (t_{dl} \ominus end(a)) \div duration(a) & \text{if } before(end(a), t_{dl}) \\ duration(a) \div (end(a) \div t_{dl}) & \text{if } before(t_{dl}, end(a)) \\ duration(a) & \text{otherwise} \end{cases}$$

Else if the postcondition of an action in the agent's plan conflicts with the jth appropriateness condition $(p_j, \Delta_j t)$ of the alarm α, and the interval in which that action is to be performed overlaps the interval $(t_{max} \ominus \Delta_j t)$ to t_{dl}, then the alarm should be shifted so that it reaches maximum at $(end(a) \ominus duration(a) \ominus \Delta_a t)$. (This p is an *unintended* consequence of a.)

$$a \in acts(\pi) \wedge p \in post(a) \wedge (p_j, \Delta_j t) \in apps(\alpha) \wedge conflict(p_j, p) \wedge$$
$$\neg(before(end(a), (t_{max} \ominus \Delta_j t)) \vee before(t_{dl}, (end(a) \ominus duration(a))))$$
$$\Rightarrow \Delta_{a\rightarrow} t = \begin{cases} (t_{dl} \ominus end(a)) \div duration(a) & \text{if } before(end(a), t_{dl}) \\ duration(a) \div (end(a) \ominus t_{dl}) & \text{if } before(t_{dl}, end(a)) \\ duration(a) & \text{otherwise} \end{cases}$$

Else if a proposition holds in the domain which conflicts with the jth appropriateness condition $(p_j, \Delta_j t)$ of α then the alarm should be shifted so that it reaches maximum at $(t_{max} \ominus \Delta_j t)$. (This p is either an unanticipated consequence of the agent's action, or due to the action of some

[2] This proposition is typically a precondition of another action in the plan, but this is not stated explicitly in this definition to minimise its complexity. Refer to Rich & Knight (1991) or Weld (1994) for a discussion of the use of constraint posting in non-linear planning algorithms.

other agent.)

$$p \in D \wedge (p_j, \Delta_j t) \in apps(\alpha) \wedge conflict(p_j, p) \Rightarrow \Delta_{i \rightarrow} t = \Delta_j t$$

The effect of a detected danger is to shift the alarm function by some appropriate period, $\Delta_{i \rightarrow} t$. This is achieved by evaluating the alarm intensity at $t_{now} + \Delta_{i \rightarrow} t$. If there is more than one danger to the timely satisfaction of a single goal, the associated alarm function is shifted by the maximum $\Delta_{i \rightarrow} t$ required.

Time commitments

In the construction of a plan, the agent commits itself to activity at certain times in the pursuit of its goals. These commitments reduce the time available for the agent to act to satisfy other goals that are not currently active. If the agent is committed to action at the same time that it needs to satisfy an inactive goal (i.e. between t_{max} and t_{dl}), then it is possible that the agent will not be reminded of the goal early enough for successful satisfaction. To determine the time commitments that the agent has made, its plan is interpreted in the way described in section 5.3.1 (the implementation of this algorithm is given in appendix B.2.4).

Definition 7.4 In the simplest terms, a time commitment conflicts with an alarm if the interval $(t_{tc} \ominus \Delta_{tc} t)$ to t_{tc} overlaps the interval t_{max} to t_{dl}. If this is the case, the alarm must be shifted so that it reaches maximum at an appropriate time before $(t_{tc} \ominus \Delta_{tc} t)$; the alarm is shifted by $\Delta_{tc \rightarrow} t$.

$$(\Delta_{tc} t, t_{tc}) \in tcs(\pi) \wedge \neg (before(t_{tc}, t_{max}) \vee before((t_{tc} \ominus \Delta_{tc} t), t_{dl}))$$

$$\Rightarrow \Delta_{tc \rightarrow} t = \begin{cases} (t_{dl} \ominus t_{tc}) \oplus \Delta_{tc} t & \text{if } before(t_{tc}, t_{dl}) \\ \Delta_{tc} t \ominus (t_{tc} \ominus t_{dl}) & \text{if } before(t_{dl}, t_{tc}) \\ \Delta_{tc} t & \text{otherwise} \end{cases}$$

However, if the alarm is shifted this may bring it into conflict with another time commitment interval. So, this procedure is repeated until the alarm is shifted sufficiently so that the interval $(t_{max} \ominus \Delta_{tc_j \rightarrow} t)$ to $(t_{dl} \ominus \Delta_{tc_j \rightarrow} t)$ does not overlap any time commitment $(\Delta_{tc_k} t, t_{tc_k}) \in tcs(\pi)$. This algorithm is given in appendix B.2.4.

7.2.3 Alarm mitigation

Once an alarm has triggered and the associated goal has been considered, the goal is no longer relevant for consideration at the present time. So, if the alarm is not deleted, the fact that the goal has been considered causes the intensity of the alarm to be mitigated. This mitigation is temporary and reduces to zero at some time t_{off}.

Definition 7.5 The mitigation function is zero before t_{now} and after t_{off}, and increases from $-(f(t_{now}) - i_{min})$ at t_{now} to zero at t_{off}.

$$m(t) = \begin{cases} 0 & \text{if } before(t, t_{now}) \vee \neg before(t, t_{off}) \\ -(f(t_{now}) - i_{min}) \times \frac{t_{off} \dot{-} t}{t_{off} \dot{-} t_{now}} & \text{otherwise} \end{cases}$$

where the ratio $\frac{t_{off} \dot{-} t}{t_{off} \dot{-} t_{now}}$ is expressed as a real number, and t_{off} decreases to $\Delta_{min}t$ as t_{max} approaches and increases to $\Delta_{max}t$ after t_{max}.

$$t_{off} = \begin{cases} t_{now} + \Delta_{min}t & \text{if } before(t_{now}, t_{max}) \wedge |t_{max} - t_{now}| \leq \Delta_{min}t \\ t_{now} + \Delta_{max}t & \text{if } before(t_{max}t_{now}) \wedge |t_{max} - t_{now}| \geq \Delta_{max}t \\ t_{now} + |t_{max} - t_{now}| & \text{otherwise} \end{cases}$$

where $|t_1 - t_2|$ is $(t_1 - t_2)$ if $t_1 \geq t_2$ and $(t_2 - t_1)$ otherwise.

7.2.4 Alarm triggering

An alarm is a structure that contains:

1. A goal, g;
2. An alarm function, $f(t)$;
3. A set of detected opportunities, $opps(\alpha)$;
4. A set of appropriateness conditions, $apps(\alpha)$;
5. A, possibly zero, time shift due to detected dangers, $\Delta_{d\rightarrow}t$;
6. A, possibly zero, time shift due to time commitments, $\Delta_{tc\rightarrow}t$; and
7. A mitigation function, $m(t)$, which is by default defined as: $m(t) = 0$.

Definition 7.6 The intensity of an alarm, α at t_{now} is given by:

$$intensity(\alpha, t_{now}) = \begin{cases} i_{max} & \text{if } opps(\alpha) \neq \emptyset \\ f(t_{eval}) + m(t_{eval}) & \text{otherwise} \end{cases}$$

where $t_{eval} = t_{now} \dot{+} \Delta_{d\rightarrow}t \dot{+} \Delta_{tc\rightarrow}t$

An alarm, α, will trigger if $intensity(\alpha, t_{now})$ exceeds the threshold, τ. The threshold is used to control the rate at which the alarms trigger to some required rate which depends on the load on the planning and control processes of the agent. The updated threshold level (i.e. the threshold level at $t_{now} \dot{+} \Delta_{cycle}t$, where the alarms are evaluated at a cycle period of $\Delta_{cycle}t$, section 3.1.5), τ', depends on:

1. The current threshold level, τ;
2. The rate at which alarms are triggering, r_{actl};
3. The required triggering rate, r_{reqd}; and
4. The maximum threshold level (the threshold ceiling), $\hat{\tau}$.

Definition 7.7 The updated threshold, τ', is reduced by $\delta\tau$ if the actual triggering rate exceeds $(r_{reqd} + \delta r)$ and is increased by the same amount, up to some threshold ceiling $\hat{\tau}$, if r_{actl} is less than $(r_{reqd} - \delta r)$. If the actual triggering rate lies between $(r_{reqd} - \delta r)$ and $(r_{reqd} + \delta r)$, there is no change to the threshold level.

$$\tau' = \begin{cases} \tau - \delta\tau & \textbf{if } r_{actl} > (r_{reqd} + \delta r) \\ \tau + \delta\tau & \textbf{if } r_{actl} < (r_{reqd} - \delta r) \wedge \hat{\tau} > (\tau + \delta\tau) \\ \hat{\tau} & \textbf{if } r_{actl} < (r_{reqd} - \delta r) \wedge \hat{\tau} \leq (\tau + \delta\tau) \\ \tau & \textbf{otherwise} \end{cases}$$

Definition 7.8 The required triggering rate is governed by the ratio of the area under the biasing function $bf(t)$ within the intervals in which the agent is committed to action, and the area under the function within the intervals in which it is free from commitment.

$$\frac{\sum_{j=1}^{n} \int_{(t_j \ominus \Delta_j t)}^{t_j} bf(t)}{\left(\int_0^{\infty} bf(t) - \sum_{j=1}^{n} \int_{(t_j \ominus \Delta_j t)}^{t_j} bf(t) \right)}$$

where the interval in which the jth action in the plan is to be performed has a duration $\Delta_j t$ and end point t_j.

Definition 7.9 The threshold ceiling, $\hat{\tau}$, is the weighted average of the importances of the actions in the agent's plan.

$$\hat{\tau} = \sum_{j=1}^{n} \left(\frac{\int_{(t_j \ominus \Delta_j t)}^{t_j} bf(t)}{\sum_{k=1}^{n} \int_{(t_k \ominus \Delta_k t)}^{t_k} bf(t)} \right) \times imp(a_j)$$

where $imp(a_j)$ is the importance assigned to the jth action in the plan.

7.3 The function of the planner

Once a goal is activated, the further processing of this goal becomes the responsibility of the planner. The planning process is provided with a list of goals by the alarm processing machinery, the "focus of planning attention", that are relevant for action at the present time. However, this focus of planning attention is not prescriptive; it is up to the planner which goal (or goals) to pursue. So, the planner must be able to alter its focus of activity (see figure 7.1) as the goals

within the focus of planning attention, and the priorities of those goals change. Suppose that the warehouse agent presently has a single goal within its focus of planning attention: the goal to have tidied away some perished stock. Typically, the agent will act on this goal, but it is not particularly important and so the current threshold level will not be set high, it being limited by the threshold ceiling. So, it is likely that a goal that is presently inactive and somewhat relevant to the present situation will trigger, then be activated by the agent. Suppose that the agent is reminded of a previously generated goal to have prepared an order for a customer who is expected to arrive in about half-an-hour. If this goal is activated, the planner must decide whether to suspend its present goal and start planning the preparation of this order or to continue tidying the warehouse and pursue the newly activated goal some time later. In making such a decision the agent must take into account the relative importances of these two goals, and the effort that has already been invested in the goal that is currently in the focus of activity along with the characteristics of their respective deadlines and any other relevant information. In the example, the agent must achieve the goal to have prepared the order before the customer arrives to be sure that it will be successful in selling the stock, but tidying away the perished stock is not as important, can be done at any time, and can be easily interrupted with little loss of efficiency. So, it may be prudent for the agent to suspend the goal it is presently acting on and prepare the order.

This scenario gives some indication of the problems involved in deciding between goals within the focus of planning attention. This thesis has concentrated on mechanisms for the control of goal activation, but this does not entirely solve the difficult problem of deciding what goal or goals should be the focus of the agent's activity (see figure 7.1). There is some work in this area using the decision-theoretic planning paradigm (Wellman & Doyle, 1991; Haddawy & Hanks, 1992), and the notion of intentions organised in temporally and structurally partial plans (Bratman et al., 1988; Rao & Georgeff, 1992).

The planner is also responsible for the deletion of goals from the focus of planning attention. A goal will be deleted once it is satisfied. Suppose the warehouse agent detects a request on the order book for it to satisfy an order at 3pm on Tuesday for some customer. The agent decides that this customer is reliable and the order is profitable and so generates a goal to have prepared this order for collection at the time specified. This goal is encapsulated in an alarm structure, and at some appropriate time later the alarm triggers and the agent considers the goal for activation. Suppose that the agent is reminded of this goal at 2.30pm, and at this time the goal is activated. The agent does not decide to act on the goal immediately because it estimates that it will take about 15 minutes to achieve the goal and there are other more urgent goals to pursue. At about twenty minutes to 3 o'clock, the agent focuses its activity on this goal and commences planning

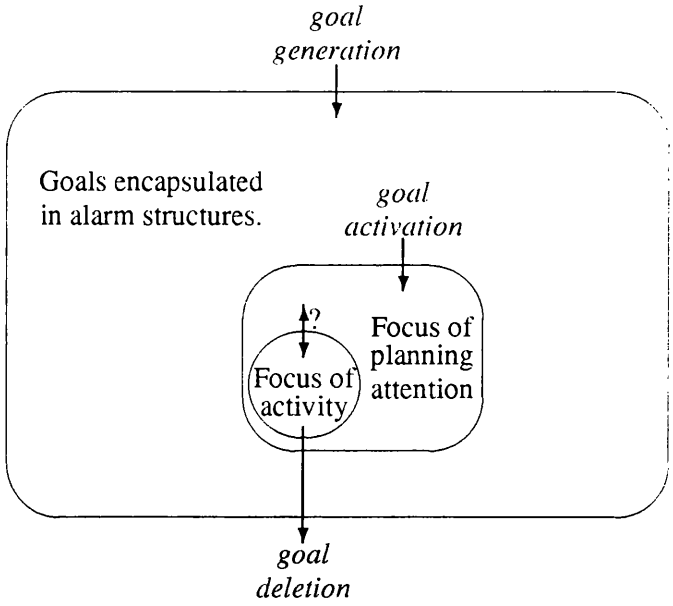


Figure 7.1: Focus of attention: 2.

to achieve it. The order is ready in time and is collected by the customer. At this point the agent recognises that the goal is satisfied and is therefore deleted by the agent.

A goal will also be deleted if the agent cannot satisfy it. Suppose that the warehouse agent activates the goal to have prepared the order for the customer at 2.45pm. The agent attends to the goal immediately, but in planning to achieve this goal, it discovers that due to a previous order for an unusually large number of widgets there is insufficient stock in the warehouse to satisfy the order. It will take too long to restock the warehouse with widgets. So, if the agent is unable to negotiate with the customer for more time, the goal cannot be satisfied, and so it will be deleted.

Cohen & Levesque (1990) amongst others recognise that if these are the only conditions under which an agent ceases to pursue a particular goal, its behaviour will be "fanatical". Suppose that after the warehouse agent has generated and activated a goal to have prepared the order requested by the customer, it receives a message from the customer cancelling the order. The goal is not satisfied and the agent can still achieve the goal (the agent can still prepare the order), but there is no longer any reason for the agent to achieve that goal (the customer will not collect it). Cohen & Levesque (1990) modify their formal model of intention by adding a number of "propositions q [which] form a background that justifies the agent's intention". Then, if the agent believes $\neg q$, the goal is no longer relevant, and so the agent no longer intends to achieve the goal; i.e. the goal is deleted. What constitutes the relevance of a goal? Cohen & Levesque (1990) suggest that this "relativisation" of an intention can be used to represent the

inter-dependency between intentions and beliefs, but they only explain that there may exist inter-dependencies between sub-goals and super-goals within an hierarchical plan structure. A more complete answer to this problem may be found by using the "reasons" for the generation of a goal (see section 6.2.2) from the alarm processing machinery as a basis for defining the inter-dependency between the agent's beliefs and intentions. For example, if one of the reasons for the generation of the goal to have prepared the order for a customer is that the customer requires the order, then if the agent is informed that the order is no longer required, the goal will be deleted.

7.4 A critical evaluation of the alarm processing machinery

The alarm processing machinery is a mechanism that schedules the consideration of goals heuristically. As with all heuristics its worth depends on whether or not its use is, on the whole, advantageous. The question that is addressed in this section is whether the advantages that are gained from the alarm processing machinery in managing a potentially large and varying number of goals out-weighs any disadvantage.

7.4.1 Consequences of a large number of relevant goals

If the intensity of an alarm does not exceed the current threshold value, the alarm will not trigger, and hence the goal encapsulated in that alarm structure will not be considered. Furthermore, it is not possible for an alarm to exceed its maximum intensity: i_{max}. Therefore, if the threshold is greater than i_{max} for a particular alarm, it will not trigger unless and until the threshold drops back below this maximum intensity. If the threshold is above this level, opportunities to satisfy the goal will be missed, and if the threshold does not decrease in time, then the deadline of the goal may pass without the agent even considering the goal.

This behaviour can be useful if the goal that is not considered would not be acted on in the current situation even if it is considered. However, because the intensity of an alarm is evaluated heuristically, there is always the possibility that either a goal that should be acted on is not considered or a goal that should not be acted on is considered (cf. Sloman (1987), Bratman (1992) and Smith (1992)). Therefore, it is important to evaluate the alarm processing machinery to determine the conditions under which such behaviour can and does occur.

Initially, suppose that threshold never reaches the threshold ceiling, and the current threshold level is set to limit the triggering rate of alarms. If the rate at which alarms are triggering exceeds the required rate the threshold is raised, and if alarms are not triggering at a sufficiently high rate it is lowered. Suppose that the rate at which alarms are triggering at the present time significantly exceeds the required rate: this may be due to: (1) an increase in the total number of alarms that influence the agent; (2) a high concentration of high intensity alarms at the present

time; or (3) an increase in the cognitive load on the agent reflected in a decreased required triggering rate. This difference in the required an actual triggering rates will cause the threshold to be immediately increased to compensate. So, the intensity of a particular alarm must increase to a higher level for that alarm to trigger and the associated goal to be considered. If this increase causes the threshold to be set at a level that is above the maximum intensity of a particular alarm, then this alarm will not trigger until either the actual triggering rate has decreased or the required triggering rate has increased, and hence the threshold is lowered. The fact that the maximum intensity of this alarm is so low that it is less than the threshold means that there are too many other alarms encapsulating goals of greater importance (i.e. i_{max}) for this goal to be worth considering in the present situation. Only when either the agent has time available to consider more alternatives, or the average intensity of the existing alarms decreases, will the threshold be lowered below the maximum intensity of this alarm.

Consider a modified version of the little Nell problem (McDermott, 1982). The heroine, Nell, is tied to the tracks, a train is approaching and the hero, Dudley, wishes to save her. The modification is that Dudley may act on a number of different goals. Suppose that the only goal that is active is the goal to have saved Nell, but Dudley also has an inactive goal to have sated his hunger, among others. Furthermore, it is lunch time so that the temporal relevance of the goal to have sated hunger is high, and so the intensity of the alarm encapsulating this goal is near i_{max}. The train is approaching and in response to this perceived event Dudley predicts that Nell will get mashed soon, so the urgency of saving Nell is also high. Therefore, Dudley will commit to action immediately in pursuit of this urgent goal. The cognitive load will increase, causing a decrease in the required triggering rate, and a consequent increase in the threshold. This high threshold reduces Dudley's sensitivity to considering alternative courses of action. The goal to have sated hunger is not particularly important, so despite the fact that this goal is relevant for action now, the intensity of the associated alarm is not high. Therefore, the goal to have sated hunger will typically not come to Dudley's attention while he is busy ensuring Little Nell's safety.

Now consider the reverse situation. The hero, Dudley, is presently busy acting on the goal to have sated his hunger, and is unaware of Nell's predicament. Then, when Dudley notices that Nell is in imminent danger, an alarm will be generated that reaches maximum intensity immediately; the goal is immediately relevant. In this situation, despite the fact that Dudley is busy with the goal that is presently his focus of activity, he will still be reminded of the inactive, but highly relevant goal to have saved Nell. This is because of the difference in importance between these two goals. The relatively low importance that Dudley associates with the goal to have sated hunger is reflected in a low threshold ceiling. So, despite the fact that the load on the cognitive

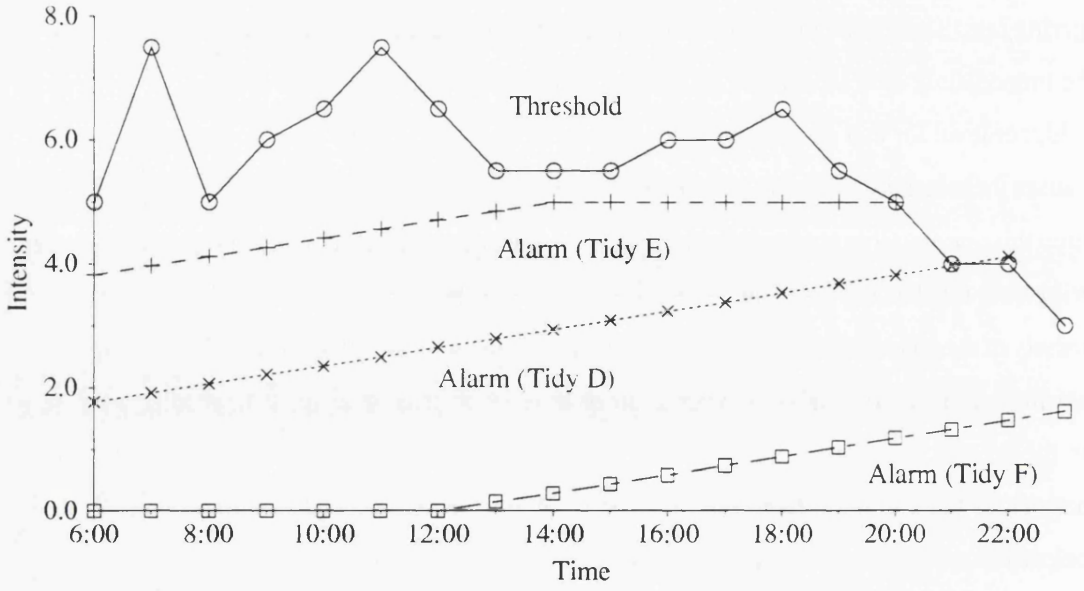


Figure 7.2: The behaviour of the alarm encapsulating the goal to have tidied room E.

resources due to the active goal to have sated hunger is high, the actual threshold level will be constrained so that a more important goal will still be considered. So, the alarm encapsulating the goal to have saved Nell will trigger, and Dudley will redirect his attention to the pursuit of this more important activity.

So, in general, the alarm processing machinery will only prevent an alarm triggering and hence a goal from being considered if: (1) A high proportion of the agent's cognitive resources are presently being invested in activities that are more important and urgent than the goal encapsulated within an alarm; and/or (2) There are a significantly high number of alarms with a greater intensity, and hence a greater number of goals with higher combined temporal relevance and importance. If there is a significant number of relevant goals, the most important inactive goals are given precedence, and if there are more important inactive goals than those the agent is attending to, these will always be considered.

Example from simulation

Figure 7.2 (produced using data from the simulation of the alarm processing machinery, see appendix C) illustrates the suppression of an alarm that is at maximum intensity due to a high threshold. This figure shows the behaviour of the alarms encapsulating the goals to have tidied rooms D, E and F in the warehouse domain and the threshold from 6am to 11pm. During the period of time between 8am and 10pm the threshold is above 5; the maximum intensities of these three alarms shown are also 5. The alarm encapsulating the goal to have tidied room E reaches maximum at 2pm, but due to this high threshold it does not trigger until 8pm when the thresh-

old drops back to 5. This relatively low intensity alarm is prevented from triggering because the agent is directed towards other higher priority goals during this period of time. Only after the orders for the day have been completed, and other alarms with higher intensities have been processed (i.e. when the threshold has dropped) will the agent consider these lower priority goals.

An inherent property of such heuristics is that alarms may be suppressed due to a high threshold when it may be important for the agent to consider that goal, even briefly. Suppose that the agent is pursuing some goal, but the actions it intends to perform conflict with the achievement of the inactive goal. The effect of these dangers is to shift the alarm function so that it is a maximum at the present time. However, the threshold is too high for this alarm to trigger, and so this interaction is not considered, and cannot be prevented. Later, when the alarm does trigger the agent will realise that its prior action prevented it from achieving this goal. This could have been prevented if the agent was able to simply consider the danger rather than decide whether or not to activate the goal.

7.4.2 Consequences of a small number of relevant goals

If the number of inactive goals drops, the average number of alarms with intensities above a particular level will tend to decrease. So, for the same triggering rate to be achieved, typically, the threshold will have to be lowered. In reality, the agent will tend to have periods of time where there are high concentrations of goals and times when there are few goals that are relevant for action. For example, in the warehouse domain there may be few orders to prepare during the period of time between 6pm and 8am. So, how does the agent behave during periods of time when there are few relevant goals to pursue?

A small number of relevant goals will tend to cause the threshold to be reduced. Therefore, the agent will tend to consider goals earlier and will be less likely to fail to meet deadlines, whatever the importance of the goal. Consider the warehouse agent during a period where there a few relevant goals to pursue. Suppose that there are no orders that are worth preparing at the present time and the warehouse is fully stocked with a wide range of goods. The only activities that have a non-zero influence on the agent's actions are those to check the order book, tidy the warehouse, recharge, and check the state of the refrigeration devices in rooms A-D (see figure 2.2). However, the agent does not expect many orders at this time as it is 2am, all the rooms of the warehouse have been recently tidied and the temperature control devices in rooms A-D show no signs of a fault. In such a situation, the agent will tend to reduce the threshold to such a level that any alarm that increases above zero will immediately trigger and the agent will activate and pursue that goal. (This behaviour seems similar to that of boredom.)

It may be advantageous for the agent to have access to goals that are of no relevance for

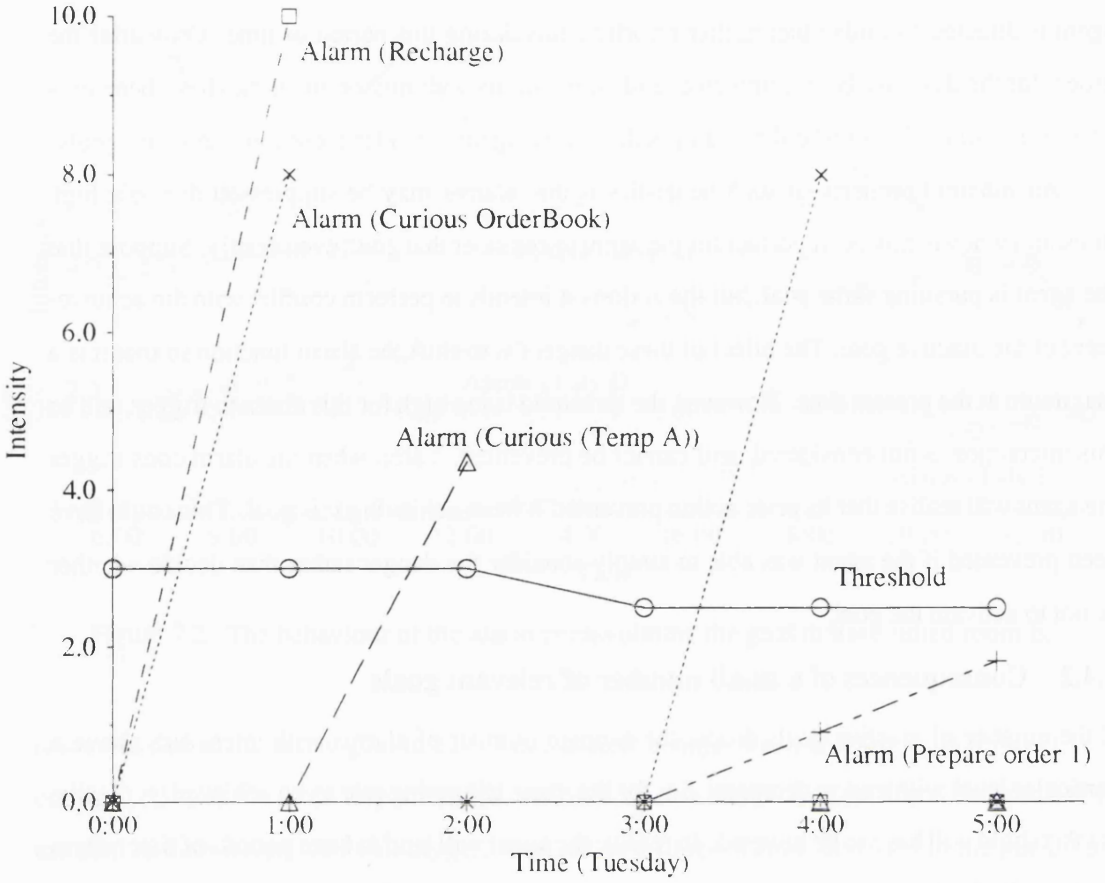


Figure 7.3: The behaviour of alarms between 12 midnight to 5am (1).

action at this time. In other words, the agent may benefit from activating goals for which there is no point in it acting. For example, the agent may benefit from activating the orders that are scheduled to be prepared during the following day so that it may check the warehouse for the required commodities without acting to prepare the order itself. This behaviour is not exhibited by the alarm processing machinery, but it is discussed by Ellis & Nimmo-Smith (1993), (Ellis, 1994) where human subjects reported that they considered tasks relevant for action later at points of low activity, or while performing routine activities during the day.

Example from simulation

Figures 7.3 and 7.4 (produced using data from the simulation of the alarm processing machinery, see appendix C) illustrates the behaviour of this simulation of the alarm processing machinery during a period of low activity. These figures show the behaviour of all the non-zero alarms that are influencing the agent during this time. The alarms illustrated in figure 7.3 increase in intensity relatively quickly. These tend to be alarms to have prepared orders and alarms encapsulating goals with a high frequency of replenishment. For example, the goal to have checked

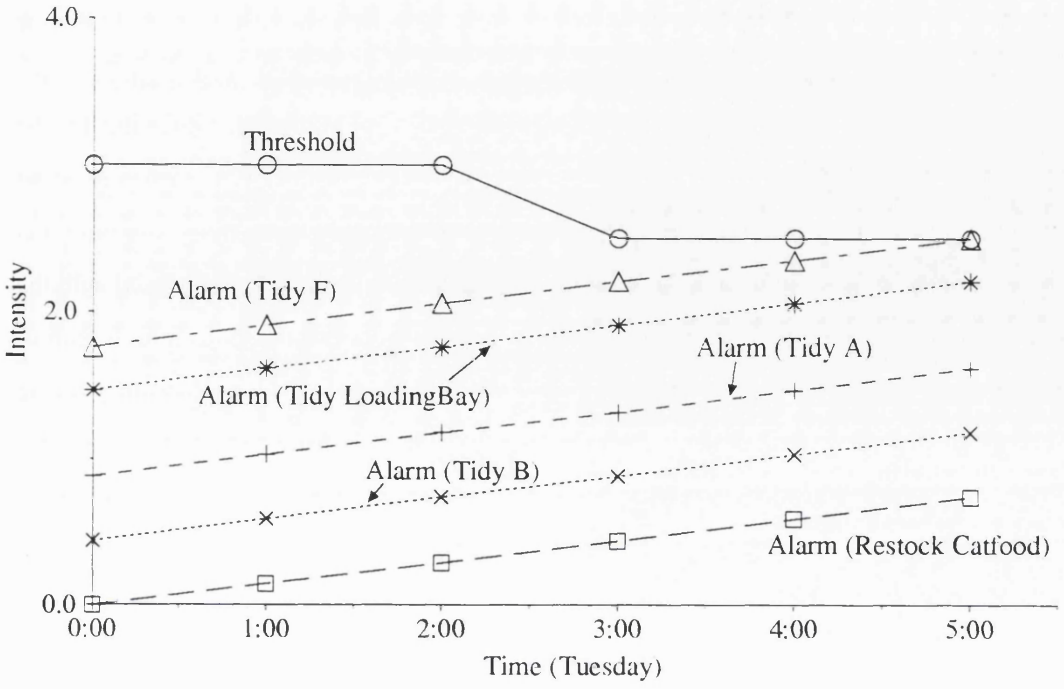


Figure 7.4: The behaviour of alarms between 12 midnight to 5am (2).

the order book is considered at 1am and activated, and then at 2am a new goal of the same type is replenished which triggers again at 4am, and then replenished again at 5am. The alarm function that represents the goal to have checked the temperature in room A is identical to those to have checked the temperatures in rooms B, C and D, and so only one is shown. The alarms illustrated in figure 7.4 have relatively shallow alarm functions. These include the alarms encapsulating goals to have tidied various rooms in the warehouse and to have restocked with various commodities. The threshold during this period varies only slightly from 3 to 2.

Notice that due to the low frequency at which the alarm functions are sampled, the alarms shown in figure 7.3 often increase far above the threshold before they are triggered. However, an increased sampling frequency will correct this. The important point here is that alarms tend to trigger long before t_{max} because their intensities exceed the threshold more easily. Furthermore, the more important a goal is, the earlier (relative to their respective t_{max} times) an alarm will trigger. Also, the agent does not need to mitigate any of the triggered alarms during this period; as soon as an alarm triggers, it is acted on.

7.4.3 Potential inefficiencies

The alarm processing machinery in certain situations may produce inefficient behaviour in the scheduling of multiple goals. This manifests itself in the discussion of time commitments in chapter 5.1. In the example discussed in section 5.3, Robby the robot intends to go to the job

centre and collect his benefit among his activities of the day. The goal to have gone to the job centre is inactive and Robby is currently planning his pursuit of the goal to have collected his benefit. In the example, Robby's prior commitments to collecting his benefit reduce the time he has at the job centre. If Robby is reminded of the goal to have gone to the job centre before going to the DSS office, he would have time to satisfy both goals before their deadlines. Suppose that Robby+ (i.e. Robby the robot with alarm processing machinery installed) is presented with the same scenario. Robby+ will be reminded of the goal to have gone to the job centre in time so that both goals can be achieved. However, before Robby+ is reminded of the job centre goal, he invests effort in planning and possibly in acting to satisfy the goal to have collected his benefit. Robby+ must suspend his activities and go to the job centre, and then return to the DSS office to collect his benefit. If Robby+ was aware of both goals earlier, effort would not have been wasted in acting to collect his benefit before being reminded of the goal to have visited the job centre. Therefore, the fact that the alarm processing machinery hides goals from the planner can lead to inefficient solutions. However, such inefficiencies will tend to be bounded due to the design of the alarm processing machinery. In particular, the mechanisms which predict temporal resource requirements ensure that the agent will be reminded of expensive goals (in terms of the resources required to achieve them) with a good margin. Also, the influence of a detected danger to the timely satisfaction of an inactive goal combined with the influence of time commitments provides a good margin.

7.4.4 Overheads in goal generation

In chapter 3, the motivated agent architecture was compared to the motive processing architecture (Beaudoin & Sloman, 1993; Beaudoin, 1994). It was argued that for an agent to generate certain types of goals (those that require a decision by the agent prior to generation) a two-stage filtering process is required. The Beaudoin & Sloman (1993) architecture (called the Motive Processing Architecture in this thesis) provides a mechanism for the filtering of situations that may warrant the generation of such a goal; i.e. the generation of goals to consider generating these goals. Here, a mechanism is presented for the direction and limiting of planning attention to goals that have been generate through decision; i.e. goals that have associated with them a deadline, delay time, estimated travel time and importance. These filtering mechanisms are therefore complementary as both types of deliberation must be limited on the basis of the cognitive resources available. The alarm processing machinery needs to be extended to manage the generation and filtering of goals to have considered generating such goals.

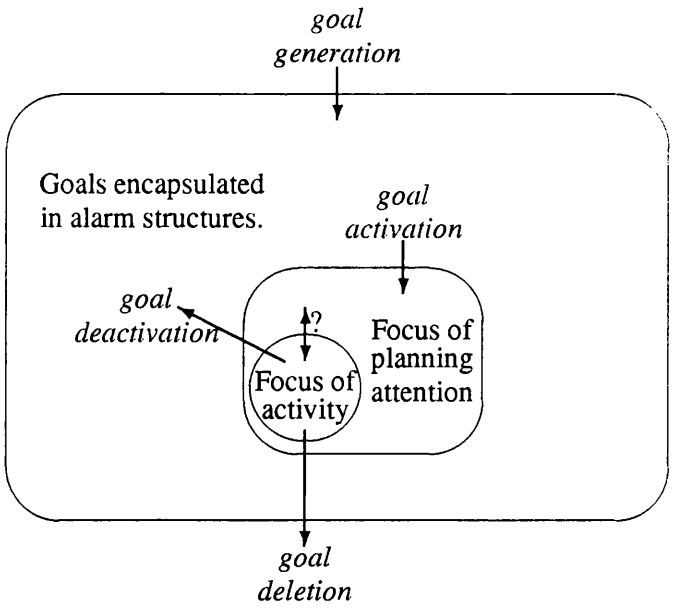


Figure 7.5: Focus of attention: 3.

7.4.5 Suspension of goals by the planner

An issue that has been mentioned in passing, but not investigated in any detail in this thesis is the suspension of goals that are presently within the focus of planning attention; i.e. goal deactivation (figure 7.5). Suppose that a particular alarm triggers early due to the agent detecting a situation that may present an opportunity to satisfy the associated goal. For example, Robby+ the robot passes an automatic recharge point on his way home from the car plant (see section 5.1). If Robby+ is not in a hurry and satisfying the goal to have recharged is relevant at the time, then Robby+ will be reminded of the goal to have recharged in the light of this opportunity. However, the fact that Robby+ is presented with an opportunity does not mean that by taking the opportunity, he is guaranteed to achieve the goal. Suppose that, when the automatic recharge machine is approached, Robby+ notices that it is out of order. In this situation, there is no point in continuing to pursue this goal; it was only the presence of the opportunity that caused the goal to be relevant for consideration in the first place. In such a situation, the agent should have the ability to deactivate the goal and return to its previous activities; e.g. Robby+ should continue his journey home. In this particular example, the goal to have recharged may be deactivated by simply treating it as a newly generated goal. However, this is not always the case.

Suppose that the agent does not have the information or the time to integrate a plan to achieve a particular goal into its current agenda, and the agent wishes to suspend consideration of that goal until some appropriate time later (Hammond, 1989b). Furthermore, some effort has already been invested in constructing a plan to achieve that goal. So, the agent has a greater amount

of information on how this goal can be achieved in the present situation. Suppose that Robby+ detects another recharge point adjacent to the first. In this situation it is better for Robby+ to simply plan to use the second than to abandon the plan altogether; it should not be necessary to regenerate the goal, detect the second opportunity and then re-plan.

In the previous example, the agent suspends the goal in the light of a failed opportunity, but the agent may also wish to suspend a goal generated through planning simply because there is no point in pursuing it at the present time. Consider an agent that has the goal to have gone on holiday. Suppose this goal has been refined into a number of more specific subgoals, all related because they have been generated in the service of the same top level goal. For example, the agent may plan to have a ticket, to have the bags packed, to have arranged a cat sitter and a number of other related goals. However, pursuing the short-term goal to have a ticket is the only activity that is relevant at the present time. It may be advantageous for the agent to suspend consideration of the certain subgoals in this abstract plan of action. However, all these goals are interrelated; there will be at the least a number precedence constraints between the achievement of these goals. If such related goals are to be encapsulated in alarm structures there must be some kind of relationship between the intensities of their alarms. For instance, if an alarm triggers that encapsulates one goal, that may cause the intensities of alarms encapsulating other related goals to increase in intensity and possibly trigger. The investigation of such inter-dependencies will again depend on the particular type of planning machinery used, but this is an avenue for future investigation.

7.4.6 Alarm function characteristics

It may be useful in further study to investigate how different alarm functions, possibly based on different information, affect the behaviour of the agent. Some possibilities are discussed in section 4.3, but exactly how various alarm function shapes will modify the overall behaviour of the agent is uncertain. All alarm functions must be maximal at t_{max} and zero at t_{dt}, must not drop back to zero, and never exceed i_{max}. Therefore, between t_{dt} and t_{max} the extremes are that the alarm function remains at zero until t_{max}, and increases to i_{max} at t_{dt} and remains at that level. An analysis of these extremes may indicate the sensitivity of the mechanism to changes in the alarm function.

In the alarm processing machinery it is assumed that the importance of a goal can be represented as a single real number. However, it is clear from the work of Sloman (1987) and Beaudoin (1994) among others that the importance of a goal does not necessarily map onto the set of real numbers. There may exist only a partial ordering between goals, or the importance of a particular goal may depend of qualitative as well as quantitative factors. These issues require

further, more detailed investigation. However, the impact of this on the core ideas presented in this thesis is restricted to implementation; the foundations remain intact. Furthermore, no explanation is given for how the importance of a goal is computed. In the initial implementation, the importance of each type of goal is assumed to be constant (e.g. all tidy goals have an importance of 5 and all order stock goals have an importance of 7). It is however, feasible that an agent has a dedicated processor that periodically computes the relative importances of different goals as the priorities of the agent changes; changes that will typically be slow.

7.5 Conclusion

This chapter has summarised and evaluated the alarm processing machinery presented in this thesis. The maintenance of a limited focus of planning attention through heuristic means can only be evaluated on whether or not it gives more benefit to an agent control system than takes in possible inefficient behaviour. This chapter has provided a convincing argument that, although the alarm processing machinery is lacking in some areas, is a good initial solution to this important problem.

Chapter 8

Conclusion

The primary objective of this thesis was to provide an operational specification for a mechanism that is capable of limiting and directing the planning attention of an agent with goal autonomy. The premises of the work are, first, real agents are resource-bounded (Simon, 1957), and second, an agent must be goal autonomous for it to successfully interact with a domain that is not entirely predictable. From this it was argued (in section 1.3) that, for such an agent to use planning effectively, it must be capable of directing and limiting its planning attention. The contribution of this detailed specification and initial implementation is significant and represents progress towards an understanding of how to design complete agents to solve problems in real-world domains.

This objective was met by first analysing the sources of goals. This analysis lead to a characterisation of goals in terms of their generation processes rather than their content or type, cf. Schank & Abelson (1977), Ortony et al. (1988) and Slade (1994). The two primary sources of goals identified are a decision to achieve some state of affairs, and the automatic replenishment of a goal, either because of a prior decision or due to the character of that particular agent. Also, it was found useful to further distinguish between the automatic replenishment of goals so that the agent pursues the goal periodically (i.e. depending on how long since the goal was last satisfied), and on the basis of some timetable. The identification of replenishment as a distinct type of goal generator was motivated primarily by the fact that automatic goal generation avoids the reasoning involved in the generation of goals through decision. Furthermore, for an agent to perform certain types of goal-directed behaviour, it is necessary for it to generate goals cyclically rather than simply responding to detected events. A more refined classification must be motivated by necessity, this is an area for future investigation.

From this analysis, the importance of the temporal relevance of a goal to a mechanism that is to bring an agent's attention to its various goals at appropriate times became clear. The various temporal characteristics of goals were then outlined and a number of encoding schemes discussed. The most simple alarm function was selected for an initial analysis. The intensity of

an alarm provides an ordering between an agent's goals. In the specification of an alarm structure the intensity of the alarm depends on the temporal relevance of the goal encapsulated in that alarm and its importance (the temporal relevance of a goal is the primary issue addressed). The agent's initial predictions about when that goal is relevant for consideration are captured in this alarm function. The structure of an alarm function may vary depending on the information available about the agent's goals. However, a simple and uniform function aids the analysis of the alarm processing mechanisms that have been developed.

It was then observed that an agent is required to act in a domain that can neither be completely nor correctly modelled, and so its initial predictions about when a goal will be relevant for consideration may be flawed, or new information may become available. In chapter 5, the various types of unexpected changes in the domain were analysed, and three distinct influences were identified. The thesis contributes by providing a more detailed account of the recollection of goals in the light of opportunities. This is motivated by the observation that if the agent is presented with a situation in which a goal may be achieved with the minimum of effort, it is only an opportunity if the goal is worth achieving. Furthermore, the attention limiting function of the alarm processing machinery will prevent the agent from being reminded of a goal in the light of an opportunity if the consideration of that goal is detrimental to its other activities. This opportunism is combined with the alarm function and the effects of dangers and time commitments to determine the intensity of an agent's alarm. If there exists a danger to the timely satisfaction of an inactive goal, the agent will tend to recall the goal earlier. If the agent is committed to action in the future that may cause a conflict with the time requirements of an inactive goal, the agent will tend to recall the goal earlier. Furthermore, the current threshold level depends on the actions in the agent's plan and when they are to be performed. Together, these influences on the likelihood of an alarm triggering, serve to heuristically schedule the consideration of inactive goals.

Finally, the thesis considers the choices that an agent may make once an alarm triggers. At this point, the alarm may be activated, deleted or considered later. The ability for an alarm to trigger a number of times and to be neither activated not deleted is important because of the heuristic nature of the thresholding mechanism. The effect of mitigation, described in chapter 6 provides an elegant solution to this problem that is consistent with the other effects on the intensity of an alarm. The thesis also provides an in-depth discussion of how the load on the agent's cognitive resources due to it planning for the achievement of its active goals. This thesis constitutes, to the author's knowledge, the only specification for a mechanism that is designed to direct and limit planning attention that is presented in sufficient detail for an implementation to

be feasible. Furthermore, an initial implementation is presented in appendices B and C, and the results from its simulation are discussed at relevant points throughout the thesis.

Appendix A

Notation

The following notation is used throughout the thesis:

- The classical propositional connectives: $\vee$, 'or'; $\neg$, 'not'; $\wedge$, 'and'; $\Rightarrow$, 'if...then...' (material implication).
- From set theory: $\emptyset$, 'empty set'; $\in$, 'in'; $\subset$, 'subset'; $\subseteq$, 'subset or equal'; $\cup$, 'union'; $\cap$, 'intersection'; $\{x : X \mid P(x)\}$, 'the set of all x in X that satisfy the predicate $P(x)$ (set comprehension).

In addition to this, the following system-specific notation is used:

Time:	t	A point in time.
	Δt	A period of time.
	$before(t_j, t_k)$	A boolean function that evaluates to true if t_j is known to be before t_k (a precedence relation).
	$t_j \oplus \Delta_k t$	A point in time which is a period $\Delta_k t$ after t_j.
	$\Delta_j t \oplus \Delta_k t$	A period of time equal to the sum of the periods $\Delta_j t$ and $\Delta_k t$.
	$t_j \ominus \Delta_k t$	A point in time which is a period $\Delta_k t$ before t_j.
	$t_k \ominus t_j$	The period of time between t_j and t_k (note, $before(t_j, t_k)$ must be true).
	$\frac{\Delta_j t}{\Delta_k t}$	A real number representing the ratio of these two periods of time.
Propositions:	p	A proposition.
	$conflict(p_j, p_k)$	The propositions p_j and p_k cannot both hold at the same time.

Actions:	a	an action.
	$pre(a)$	A set of propositions that are the preconditions of a.
	$post(a)$	A set of propositions that are the postconditions of a.
	$end(a)$	The point before which the agent intends to complete a. Note that this is only meaningful if the action is instantiated in a plan.
	$duration(a)$	An estimate of the time required to perform a. (Only meaningful if a is instantiated in a plan.)
	$imp(a)$	The importance that the agent ascribes to the performance of the action. (Only meaningful if a is instantiated in a plan.)
Plans:	π	A plan.
	$acts(\pi)$	The set of actions in π.
	$pcs(\pi)$	The set of persistence constraints in π. (A persistence constraint is a 3-tuple (a, p, t) where $p \in post(a)$ and the proposition p must hold until t.)
	$tcs(\pi)$	The set of time commitments the agent has made in the plan π. (A time commitment is a tuple $(\Delta t, t)$ where Δt is the duration of the time commitment and t is the end of the commitment.)
Goals:	g	A goal.
Intensity:	i	An intensity. Intensity is modelled by real numbers, so the normal arithmetic operators ($+, -, \times, \div, >, <$, etc.) may be used. (Note, there is no difference between the intensity level i_{max} of an alarm and the importance of a goal.)
	τ	The threshold. (The threshold is an intensity.)
	$\delta\tau$	A small change in the threshold.
	$\hat{\tau}$	The threshold ceiling.
Alarms:	α	An alarm.
	$anticipated(\alpha)$	The actions that are anticipated as potential opportunities to satisfy the goal associated with α, and encoded as opportunity demons.
	$opportunities(\alpha)$	The actions that are opportunities to satisfy the goal associated with α in the present state.

	$app(\alpha)$	The conditions under which α is appropriate.
	$intensity(\alpha, t)$	The intensity of α at time t.
Functions:	$f(t)$	An alarm function.
	$m(t)$	A mitigation function.
	$bf(t)$	A biasing function.
Rates:	r_{reqd}	The required triggering rate of alarms.
	r_{actl}	The actual (or measured) triggering rate.
	δr	A small change in the triggering rate.

Appendix B

A prototype motivated agent

B.1 The Miranda syntax

Here the main features of Miranda^{TM}, a particular example of a lazy functional language, are summarised as an aid to the reader interested in the code but unfamiliar with the language. (See Bird & Walder (1988) for a good introduction to functional programming.)

A program written in Miranda consists of a "script", which lists the definitions (in any order) of a set of functions. A function takes one or more input parameters and delivers a single output value. Its result is independent of the context in which it is called, so the function will always return the same value for the same input values - this property is called *referential transparency* and is important in making programs written in the language easy to reason about. One consequence of this definition of functions is that, in contrast to *procedures*, they can have no side-effects. That is, it is always possible to understand the behaviour of a function without reference to anything outside the body of its definition, since it can have no effects outside the scope of that definition, nor depend on anything that is not explicitly passed in as an input parameter.

Miranda, in common with most functional languages, is higher-order. This means that functions are values, so they can be passed as input parameters to other functions and returned as the result of functions. This has a powerful effect, which is to allow a very high degree of abstraction. Entire functional behaviours can be abstracted into higher-order functions, to be tailored to specific tasks by a function offered as an input parameter. For example, the higher-order function `map` which applies a given function to every element in a given list is a powerful functional abstraction. This abstraction is a major contributor to the conciseness of the language: it is estimated that Miranda offers an order of magnitude reduction in the size of source code over more traditional languages for a given task.

Miranda is also a lazy language. This means that values are constructed by need, so that values which are never needed are never evaluated, regardless of the existence of references to

those values. One consequence of this is that it is possible to refer to arbitrarily large structures, such as the list of all natural numbers, even though the computation of the complete structures would not be possible. Laziness offers a powerful way to decompose many problems. For example, search problems can be decomposed into the generation of the search space and, quite separately, the exploration of the elements of that space, where in a non-lazy language the search would have to be interspersed with the construction of the space in order to gain the most efficient behaviour (to avoid constructing parts of the space which are eventually not required).

A Miranda function definition consists of a type-declaration, which is optional since Miranda can infer it from a function definition and which specifies the types of the input parameters and the output value, and the definition of the function behaviour. The latter is given as one or more equations, consisting of a left-hand-side (the name of the function and local names for the input parameters) and the right-hand-side. The right-hand-side defines a result using the input parameters and any other functions (including inbuilt functions, such as arithmetic operators) or constants that are defined in the environment. The function might return different results depending on conditions met by one or more of the input parameters. These conditional functions are expressed using "guards". The condition is placed after a comma, following the value that is returned if that guard is true. Guards are evaluated in order, and the first true guard indicates the value to be returned, even if subsequent guards might also have been true. A default guard of "otherwise" can be used in the last case of a function, which is always true. Conditions can also be expressed using *pattern-matching* which allows the specification of patterns on the left-hand-side in naming input parameters which must be matched by the input value in order to use that function clause. For example:

```
length (x:xs) = 1 + length xs
length [] = 0
```

defines a function `length` which evaluates the length of an input list. The first clause specifies that the input must match `x:xs` in order for it to be used. This means that the input must be a list containing at least one element (the first element in the list) which will be called `x` and the rest of the list (which could be empty) will be called (locally) `xs`. The right-hand-side specifies that the length of this input list can be found by adding one to the length of the remainder of the list once the first element is removed. The second equation (or clause) specifies that the length of the empty list is zero. Again, pattern-matching will ensure that the second clause is used only when the list is empty, so these two clauses could in fact be entered in either order.

Miranda has several basic types available to the programmer: numbers (real and integer), characters and boolean values are offered as basic primitives. In addition lists of elements of any

single type (e.g. `[num]` is a list of numbers), functions and "tuples" are supported. Tuples are vectors of elements (which can have different types) and can contain any number of elements (each different sized vector and vector containing different components being a different type) (e.g. `(num, char)` is a tuple that contains a single number and a single character). In addition, algebraic data types allow new types to be constructed by enumerating the components from which they are built (e.g. the algebraic type `order`, page 156). This allows finite types, type sums and recursive types to be constructed very easily. Abstract data types offer a useful way to modularise code, essentially providing a data-hiding mechanism to separate implementations of concrete types from the abstract type used by the programmer.

B.2 The code

In this section the Miranda^{TM} code which makes up the alarm processing machinery of the prototype motivated agent is presented. The code is written in 'literate' style, which means that everything is a comment except the lines beginning with the symbol '>'.

Although Miranda can infer the types of the functions and data structures involved, they are written into the code explicitly, as an aid to the reader and as good programming practice.

Here is some brief information about basic Miranda functions that are used in the program: `map`, `filter`, `++`, `:`, `member` and `concat` are standard Miranda functions. `map` applies a function to each member of a list. `filter` filters out of a list all the elements that do not satisfy a given predicate. `++` appends two lists together, and is an infix operator. `:` (cons) adds an element to the front of a list, and is also an infix operator. `hd` returns the first element of a list, while `tl` returns a list except its first element. `member` returns `True` if a given element is in a list and `False` otherwise. `concat` appends a list of lists together, producing one big list. Furthermore, the language provides a concise method of manipulating lists through list comprehension. The function `mapfilter` uses list comprehension in its definition. (As the name of this function suggests, it is similar to a combination of the standard functions `map` and `filter`; in fact, the function `mapfilter2` is an alternative definition of the same function.) The function of `mapfilter` is as follows: each element, `x`, in the list `xs` is tested with the predicate function `g`. If this evaluates to the symbol `True`, then the function `f` is called with this `x`, the result of this function then becomes the next element of the resulting list. The two examples below illustrate this behaviour (`Miranda` is the prompt of the Miranda programmimg environment). These two function calls find the list of squares of the domain list, `[1,2,3,4,5,6]`, for each element that is greater than 3.

```
> mapfilter :: (*->**) -> (*->bool) -> [*] -> [**]
> mapfilter f g xs = [f x | x<-xs ; g x]
```

```
> mapfilter2 :: (*->**) -> (*->bool) -> [*] -> [**]
> mapfilter2 f g xs = map f (filter g xs)

Miranda mapfilter (^2) (>3) [1,2,3,4,5,6]
[16,25,36]
Miranda mapfilter2 (^2) (>3) [1,2,3,4,5,6]
[16,25,36]
```

The implementation is designed around a number of abstract types: `time`, `goal`, `action`, `plan`, `alarm` and `alarms`. The specification of the warehouse domain is not provided, but it is assumed that a full implementation includes some representation of the domain physics and specifications of a particular agent's motives; see section 2.2 for a description of the warehouse domain. Furthermore, the definitions of a number of functions are omitted; these are either trivial, or provide unnecessary detail. Those functions that are omitted from this appendix, but are worth mentioning are functions that overload the standard Miranda function `show`. This function is overloaded because it can be defined so that an object of any type may be transformed into a list of characters if a function exists to do so, and if the type that is to be transformed is clear in the context that this function is being used. For example, it is possible to `show` an object of type `num`, `bool`, `(num, char)`, etc. Furthermore, if an abstract object of type `time` is defined, a function may be included in the signature of that abstract object called `showtime`. This further overloads the function `show` so that abstract objects of type `time` may also be transformed into lists of characters for display. This function is worth noting because the show function is overloaded in such a way that instances of any of the abstract objects defined here may be displayed. What this means is that through the simulation described in appendix C, the output provided at each step is the output of the overloaded show function called with every alarm that is triggered, mitigated and not considered at every time step. Thus, the information displayed is the actual output of the simulation, and not some sanitised interpretation of the output.

The structure of this appendix is as follows. Sections B.2.1, B.2.2, B.2.3, and B.2.4 present the `time`, `goal`, `action` and `plan` abstract type specifications respectively, along with relevant type and function definitions. The structure and triggering of demons are specified in section B.2.5, the alarm and mitigation functions are specified in section B.2.6, and in section B.2.7 a prototype goal replenishment process is presented.

B.2.1 Temporal representation

The type `time` is implemented as an abstract type. A time point or interval is represented as a tuple of five tuples. The first five tuple represents the estimate of the time point or interval,

ε, and the second represents the accuracy of that estimate, λ (see section 4.2). The fields of a single five tuple represent a number of weeks, days, hours, tens of minutes, and minutes in that order. (The definition of `zerotime` below is referred to a number of times in this appendix, and denotes the time referred to as t_0 throughout the thesis.)

```
> abstype time
> with zerotime  :: time
>      plus      :: time -> time -> time
>      minus     :: time -> time -> time
>      divide    :: time -> time -> num
>      before    :: time -> time -> bool
>      shorter   :: time -> time -> bool
>      mintime   :: [time] -> time
>      maxtime   :: [time] -> time
>      infertime :: time -> [time] -> time
>      maketime  :: (timetuple, timetuple) -> time

> time == (timetuple, timetuple)
> timetuple == (num, num, num, num, num)

> zerotime == ((0,0,0,0,0),(0,0,0,0,0))
```

The addition, $\oplus$, and subtraction, $\ominus$, of two time points or intervals are performed by the functions `plus` and `minus`. The addition of two points in time, the addition of a time point to an interval and the subtraction of a time point from a time interval are meaningless (section 4.2.2), so it is assumed that these operators are never used in these circumstances. The subtraction of a time point, t_k or interval, $\Delta_k t$, from a time point, t_j, is only meaningful if the interval being subtracted (i.e. $(t_k - t_0)$ or $\Delta_k t$ respectively) is smaller than the interval $(t_j - t_0)$. Therefore, it is necessary to check for the legal use of the function `minus` with the error guard indicating a negative result. (It is assumed that $t = t - t_0$, hence the definition of `zerotime`.) To maintain the accuracy of the result of an addition or subtraction of two times, the accuracy of the result is always the sum of the accuracy of the two arguments (see section 4.2.2).

```
> plus (e1, l1) (e2, l2) = (add e1 e2, add l1 l2)

> minus (e1, l1) (e2, l2)
> = error "minus: negative result", less_than e1 e2
> = (subtract e1 e2, add l1 l2),    otherwise
```

To support these functions, the functions `add`, `subtract` and `less_than` are required. The representation of a timetuple necessitates more involved addition and subtraction functions; there are ten minutes in a single ten minute period, six ten minute periods in an hour, twenty four hours in a day, and seven days in a week. The function `carry_and_remainder` returns a tuple with either a 1 or 0 as the first field (indicating whether or not there is a carry), and the result of the addition of the two timetuple fields passed to that function, minus the base if there is a carry (`borrow_and_remainder` performs a similar function).

```
> add :: timetuple -> timetuple -> timetuple
> add (w1,d1,h1,t1,m1) (w2,d2,h2,t2,m2)
> = ((w1 + w2 + cw),rd,rh,rt,rm)
>   where
>   (cw,rd) = carry_and_remainder d1 d2 7 cd
>   (cd,rh) = carry_and_remainder h1 h2 24 ch
>   (ch,rt) = carry_and_remainder t1 t2 6 ct
>   (ct,rm) = carry_and_remainder m1 m2 10 0

> carry_and_remainder :: num -> num -> num -> num -> (num, num)
> carry_and_remainder x y base carry
> = (0, sum),          sum < base
> = (1, sum - base), otherwise
>   where
>   sum = x + y + carry

> subtract :: timetuple -> timetuple -> timetuple
> subtract (w1,d1,h1,t1,m1) (w2,d2,h2,t2,m2)
> = ((w1 - w2 - bw),rd,rh,rt,rm)
>   where
>   (bw,rd) = borrow_and_remainder d1 d2 7 bd
>   (bd,rh) = borrow_and_remainder h1 h2 24 bh
>   (bh,rt) = borrow_and_remainder t1 t2 6 bt
>   (bt,rm) = borrow_and_remainder m1 m2 10 0

> borrow_and_remainder :: num -> num -> num -> num -> (num, num)
> borrow_and_remainder x y base borrow
> = (1, difference + base), difference < 0
> = (0, difference),         otherwise
>   where
>   difference = x - y - borrow
```

The function `less_than` evaluates to `True` if the first argument is less than the second where each argument is of type `timetuple`. The first argument is less than the second if the number of weeks in the first, `w1`, is less than the number of weeks in the second, `w2`, or if the number of weeks are equal and the number of days in the first, `d1`, is less than the number of days in the second, `d2`, etc.

```
> less_than :: timetuple -> timetuple -> bool
> less_than (w1,d1,h1,t1,m1) (w2,d2,h2,t2,m2)
> = (w1<w2) \/ (ew & d1<d2) \/ (ewd & h1<h2) \/
>   (ewdh & t1<t2) \/ (ewdht & m1<m2)
>   where
>   ew = w1=w2
>   ewd = ew & d1=d2
>   ewdh = ewd & h1=h2
>   ewdht = ewdh & t1=t2
```

The function `divide` and its support function `implode` are used to generate an estimate of the division of one time into intervals of another. (This is only meaningful if both arguements are intervals of time.)

```
> divide (e1, l1) (e2, l2)
```

```
> = implode e1 / implode e2

> implode :: timetuple -> num
> implode (w,d,h,t,m) = m+(10*t)+(60*h)+(1440*d)+(10080*w)
```

The function `before` is used to compare two time points to detect whether the first is known to be before the second, and as a time point is defined as an interval after `zerotime`, the function can also be used to determine whether one interval of time is shorter than another. However, to use a function named `before` for this purpose may lead to confusion, and so the function `shorter` is added to the signature of the abstract type time. One time, $t_1 = (\varepsilon_1, \lambda_1)$, is only known to occur before another time point, $t_2 = (\varepsilon_2, \lambda_2)$, if $add(\varepsilon_1, \lambda_1)$ is less than $subtract(\varepsilon_2, \lambda_2)$; see definition 4.6.

```
> before (e1, l1) (e2, l2)
> = less_than (add e1 l1) (subtract e2 l2)

> shorter = before
```

The functions `mintime` and `maxtime` evaluate to the minimum interval (or earliest time point) and maximum interval (or latest time point) respectively in a list of intervals or time points.

```
> mintime [t] = t
> mintime (t1:t2:ts)
> = mintime (t1:ts), before t1 t2
> = mintime (t2:ts), otherwise

> maxtime [t] = t
> maxtime (t1:t2:ts)
> = maxtime (t1:ts), before t2 t1
> = maxtime (t2:ts), otherwise
```

The function `infertime` evaluates to the next point in time that conforms to certain characteristics after some point in time. For example, if the current time is 9am on a Monday, and the next either Wednesday at 5pm or Saturday at 11am is required, the time 5pm on the coming Wednesday will be returned. This function is used in determining the deadline of a replenished timetabled R-Goal (see section B.2.7). For every element of the list of time characteristics, `ts`, passed to this function, the function `nextpoint` is used to determine the next absolute time with these characteristics (i.e. from the current time `now`). This is a list of next possible deadlines. Suppose that the temporal characteristics of the replenishment process are 5pm on Wednesdays and 11am on Saturdays, and the process is invoked at 9am on some Monday. The function `nextpoint` is used to find the times 5pm this coming Wednesday and 11am this coming Saturday. Then the function `mintime` is used to find the earliest of these times; i.e. 5pm this coming Wednesday.

```
> infertime now ts = mintime (map (nextpoint now) ts)

> nextpoint :: time -> time -> time
> nextpoint ((w1,d1,h1,t1,m1),l1) ((0,0,0,0,m2),l2)
> = np,               before ((w1,d1,h1,t1,m1),l1) np
> = (plus np tenmin), otherwise
>   where
>   np = ((w1,d1,h1,t1,m2), (add l1 l2))
>   tenmin = ((0,0,0,1,0),(0,0,0,0,0))

> nextpoint ((w1,d1,h1,t1,m1),l1) ((0,0,0,t2,m2),l2)
> = np,             before ((w1,d1,h1,t1,m1),l1) np
> = (plus np hour), otherwise
>   where
>   np = ((w1,d1,h1,t2,m2), (add l1 l2))
>   hour = ((0,0,1,0,0),(0,0,0,0,0))

> nextpoint ((w1,d1,h1,t1,m1),l1) ((0,0,h2,t2,m2),l2)
> = np,            before ((w1,d1,h1,t1,m1),l1) np
> = (plus np day), otherwise
>   where
>   np = ((w1,d1,h2,t2,m2), (add l1 l2))
>   day = ((0,1,0,0,0),(0,0,0,0,0))

> nextpoint ((w1,d1,h1,t1,m1),l1) ((0,d2,h2,t2,m2),l2)
> = np,             before ((w1,d1,h1,t1,m1),l1) np
> = (plus np week), otherwise
>   where
>   np = ((w1,d2,h2,t2,m2), (add l1 l2))
>   week = ((1,0,0,0,0),(0,0,0,0,0))

> nextpoint x y = error "nextpoint: can't infer weeks"
```

B.2.2 Goal representation

The type `goal` is implemented as an abstract type. The signature of this abstract type specifies a number of functions for extracting and modifying the fields of the goal structure. (The implementation of these functions are trivial.)

```
> abstype goal
> with get_goal_id  :: goal -> goalid
>      get_goal_imp :: goal -> importance
>      get_goal_dt  :: goal -> delaytime
>      get_goal_dur :: goal -> duration
>      get_goal_dl  :: goal -> deadline
>      put_goal_dt  :: delaytime -> goal -> goal
>      put_goal_dur :: duration -> goal -> goal
>      put_goal_dl  :: deadline -> goal -> goal

> goal == (goalid, importance, delaytime, duration, deadline)
```

There is no essential difference between a `delaytime` and a `deadline` or any other instance of the type `time`, except for the information that is contained within that data structure. Therefore, for the sake of clarity a number of type synonyms are defined below.

```
> importance == num
> delaytime == time
> duration == time
> deadline == time
```

B.2.3 Action representation

The type `action` is implemented as an abstract type. The signature of this abstract type specifies a number of functions for extracting information from actions, and a single function, `put_act_end` for modifying the end point of the action. It is assumed that when an action, `a`, is instantiated in a plan, it will contain up-to-date information about when it should be performed by, `(get_act_end a)`, and how long it will take to perform, `(get_act_dur a)`. (The implementation of these functions are trivial.)

```
> abstype action
> with get_act_id      :: action -> actionid
>      get_act_end     :: action -> time
>      get_act_dur     :: action -> duration
>      preconditions   :: action -> [proposition]
>      postconditions  :: action -> [proposition]
>      put_act_end     :: time -> action -> action

> action == (actionid, duration, time,
>            [proposition], [proposition])
```

B.2.4 Plan representation

In the implementation of a prototype alarm processing machinery, the agent is assumed to employ a causal-link style constraint-posting planning algorithm (Weld, 1994). The agent's plan is represented as a three tuple consisting of a list of actions instantiated in the plan, a list of ordering constraints between these actions, and a list of causal links (or persistence constraints). An ordering constraint can be either a `PrecedenceConstraint` where the first action is to be performed before the second, or a `ExternalConstraint` where the action is to be performed before some point in time. It is common to represent a causal link as a triple `(action, proposition, action)` to explicitly tie together two actions. However, no planner is implemented in this prototype, so the representation `(action, proposition, time)` is used to simplify the function that determines the time shift due to a dangerous intended consequence of a planned action.

A plan is implemented as an abstract type. The signature of this abstract type specifies those functions that may manipulate structures of the type `plan`. The functions `planned_actions`, `ordering_constraints` and `causal_links` are trivial. The functions `dangers`, `propagate_constraints` and `time_commitments` are specified in sections B.2.4, B.2.4 and B.2.4 respectively along with their relevant type definitions.

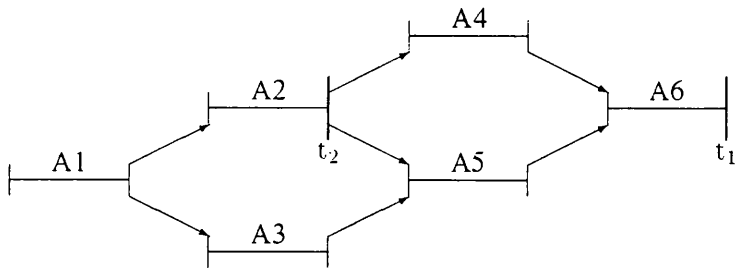


Figure B.1: A partially ordered plan with two external time constraints, t_1 and t_2.

```
> abstype plan
> with planned_actions       :: plan -> [action]
>      ordering_constraints  :: plan -> [order]
>      causal_links          :: plan -> [link]
>      propagate_constraints :: plan -> plan
>      dangers               :: plan -> [appropriate] ->
>                               ([(appropriate, action)],
>                                [(appropriate, link)])
>      time_commitments      :: plan -> [commitment]

> plan == ([action], [order], [link])
> order ::= PrecedenceConstraint action action |
>           ExternalConstraint action time
> link == (action, proposition, time)
```

Constraint propagation

It is assumed that every goal that a motivated agent will pursue through deliberative action has some finite temporal relevance. Therefore, every goal has a deadline. An important consequence of this assumption is that every plan that a motivated agent generates will have at least one external time constraint, and this necessary constraint will be on the action that achieves the goal for which the plan has been constructed. All other actions must be performed before this action, and so every action in the plan must be performed by some point in time for subsequent actions to be performed, and hence for the final action to be performed so that the goal is achieved before its deadline. Consider the partial plan illustrated in figure 5.6 and reproduced in figure B.1. There are two external time constraints on the actions in this plan, but by virtue of the precedence constraints within the plan and the durations of the action that are to be performed, a time constraint can be found for any action in that plan. The function `propagate_constraints` serves to update the constraints on all actions in a plan.

```
> propagate_constraints (as, os, ls)
> = (map (propagate (external_constraints os) os) as)

> external_constraints :: [order] -> [action]
> external_constraints [] = []
```

```
> external_constraints ((PrecedenceConstraint a1 a2):os)
> = external_constraints os
> external_constraints ((ExternalConstraint a t):os)
> = a : external_constraints os
```

The function `propagate` updates the end point of a single target action, `ta`, in a plan given the actions with external time constraints in that plan, `cs`, and the ordering constraints on actions in that plan, `os`. The end point of an action is the minimum end point inferred from the external time constraints on actions and the precedence constraints between actions in the plan. Suppose that an action A1 is the target, and action A6 is constrained to be performed before t_1 (see figure B.1). If the plan contains the precedence constraints (A5,A6), (A3,A5), and (A1,A3), then the action A1 must be performed before A3, which must be performed before A5, which must be performed before A6, which must be performed before t_1. Therefore, A1 must be performed before:

$$t_1 \ominus duration(A6) \ominus duration(A5) \ominus duration(A3)$$

The function `collect_paths` evaluates to a list of paths through the plan from an action with an external time constraint to the target action. So, in that case of action A1 in this example, the function `collect_paths` evaluates to:

```
[[A6,A5,A3],[A6,A5,A2],[A6,A4,A2],[A2]]
```

From this list of paths, the minimum inferred end point of the action A1 can be determined through the use of the functions `infer_end_point`, the in-built function `map`, and `mintime`.

```
> propagate :: [action] -> [order] -> action -> action
> propagate ecs os t
> =  put_act_end (mintime (map (infer_end_point t)
>                              (collect_paths os t ecs)))

> infer_end_point :: action -> [action] -> time
> infer_end_point t [] = get_act_end t
> infer_end_point t path
> = foldl minus (get_act_end (hd path)) (map get_act_dur path)
```

The function `collect_paths` when called by the function `propagate` generates the list of all paths (a single path being a list of actions) through the plan from an action with an external time constraint (i.e. an action in the list `ecs`) to the action under consideration (i.e. the target action, `t`) (see the function `propagate`). This list is constructed by the functions `collect_paths` and `allpaths` operating together. The function `collect_paths` concatenates the lists of paths to the target action, `t`, from a list of actions, `as`, using the function

allpaths (see the function collect_paths). Initially, as contains the actions with external time constraints. The function allpaths is designed to return a list of paths from a single current action, c, to the target action, t. A single path is a list of actions consisting of c at the head of the list, and the rest of the list consisting of the actions between the target action and the constrained action, but not including the target action. The rest of this list is generated by calling the function collect_paths with all the actions that are constrained to be performed before the current action c. (These previous actions are found using the function prev.) There may be more than one action constrained to be performed before c, and so the function collect_paths is used to follow paths from all previous actions.

```
> collect_paths :: [order] -> action -> [action] -> [[action]]
> collect_paths os t as = concat (map (allpaths os t) as)

> allpaths :: [order] -> action -> action -> [[action]]
> allpaths os t c
> = [[]]                                          c=t
> = map (c:) (collect_paths os t (prev os c)), otherwise

> prev :: [order] -> [action]
> prev [] a = []
> prev ((PrecedenceConstraint a1 a2):os) a2 = a1 : prev os a2
> prev (o:os) a = prev os a
```

Distinguishing dangerous intended and unintended actions

The functions dangers and alldangers serve to distinguish between dangers due to intended and unintended consequences of its future actions. If an effect (or postcondition) of an action is intended, the planner is assumed to have posted a causal link (or persistence constraint) on that effect. For a particular conflict, (app,p,a), there can be either one or no causal links constraining the proposition p which is a consequence of the action a to some point in time. If there are no such causal links, the consequence of that action is said to be unintended. If there is a causal link, the consequence is intended. The function dangers evaluates to a tuple of lists of tuples corresponding to the unintended and intended dangerous consequences of the actions in the agent's plan.

```
> appropriate == (proposition, time)

> dangers (as, os, ls) apps
> = alldangers (allconflicts apps as) ls

> alldangers :: [(appropriate, proposition, action)] -> [link] ->
>               ([(appropriate, action)], [(appropriate, link)])
> alldangers [] ls = []
> alldangers ((app, p, a):rest) ls
> = (unintended, ((app, (hd cls)):intended)), cls ~= []
> = (((app, a):unintended), intended),        otherwise
>   where
```

```
>    (unintended, intended) = alldangers rest ls
>    cls = [l | l<-ls ; ((fst3 l) = a) & ((snd3 l) = p)]
```

The function `allconflicts` returns a list of appropriateness condition, proposition, action, triples that correspond to those actions within the plan that either as an intended or unintended consequence, causes a conflict with an appropriateness condition due to its postcondition `p`. An action conflicts with an appropriateness condition if any of the postconditions of that action conflict with the proposition that is the first field of the appropriateness condition tuple (`find_conflicts`). The function `conflict` depends on the domain, and so only the type of the function is of interest here.

```
> allconflicts :: [appropriate] -> [action] ->
>                 [(appropriate, proposition, action)]
> allconflicts apps as
> = concat (map (find_conflicts apps) as)

> find_conflicts :: [appropriate] -> action ->
>                   [(appropriate, proposition, action)]
> find_conflicts apps a
> = [(app, p, a) | app<-apps ; p<-(filter (conflict (fst app))
>                                         (postconditions a))]

> conflict :: proposition -> proposition -> bool
```

Time commitments

A plan represents the commitments that an agent has made to action in the pursuit of its goals. So, a plan can be viewed as a number of intervals of time in which the agent is committed to action. However, this information is implicitly represented in the durations of actions, ordering constraints between actions and external time constraints imposed on the plan. Therefore, the function `time_commitments` is used to transform a structure of type `plan` into a list of commitment intervals representing the intervals of time that the agent has committed to action in the construction of this plan. A single `commitment` is a tuple, the first field of which represents the duration, and the second field represents the end of that commitment. For example, the `egcommit` represents a commitment to about an hour of activity before 3pm on the Wednesday of the third week from `zerotime`.

For all the actions in the plan, `as`, a commitment interval may be found for that action in isolation using the function `propagate_constraints`. At this point it is assumed that the actions within the plan contain up-to-date information about their minimum end point. The commitments that the actions in the plan represent are extracted using the function `get_commitments`, and sorted in order of the latest commitment first via the `sort_commitments` function. (This is implemented using the quick sort algorithm.) The, now sorted, commitment intervals are then simplified using the function `simplify`.

```
> commitment == (time, time)

> egcommit :: commitment
> egcommit = (maketime ((0,0,1,0,0),(0,0,0,0,2)),
>             maketime ((3,3,15,0,0),(0,0,0,1,0)))

> time_commitments (as,os,ls)
> = (simplify.sort_commitments.get_commitments) as

> get_commitments :: [action] -> [commitment]
> get_commitments as = [(get_act_dur a, get_act_end a) | a<-as]

> sort_commitments :: [commitment] -> [commitment]
> sort_commitments [] = []
> sort_commitments (x:xs)
> = sort_commitments small ++ [x] ++ sort_commitments large
>   where
>   (small, large) = split_commitments xs x

> split_commitments :: [commitment] -> commitment ->
>                      ([commitment], [commitment])
> split_commitments [] x = ([],[])
> split_commitments (u:xs) x
> = ((x:small), large), if before (snd x) (snd u)
> = (small, (x:large)), otherwise
>   where
>   (small, large) = split_commitments xs x
```

The simplification of commitment intervals is simple once the intervals are sorted so that the interval with the latest end point is first. If there is a single commitment interval, `c`, contained in the list, the function terminates. If there are two or more commitment intervals (`c1`, `c2` and the, possibly empty, list of actions `cs`) in the list, and the second, `c2`, ends before the first, `c1`, starts, then they cannot be combined and a new list is generated with the first commitment interval as its head using the ":" operator. Otherwise, these two intervals can be combined by summing their durations.

```
> simplify :: [commitment] -> [commitment]
> simplify [c] = [c]
> simplify (c1:c2:cs)
> = c1 : simplify (c2:cs),          before dl2 (minus dl1 dur1)
> = simplify ((plus dur1 dur2, dl1):cs), otherwise
>   where
>   (dur1, dl1) = c1
>   (dur2, dl2) = c2
```

B.2.5 Demons

Demons are used for a variety of purposes within the agent architecture described in this thesis. They are used for the detection of situations that may warrant the generation of a goal (section 4.4), the invoking of a replenishment process by detecting the deletion of a previously replenished goal (section 4.5.1), detecting a situation in which a replenishment process

will be deleted (section 4.5.2), detecting potential opportunities to satisfy inactive goals (section 5.1), and the detection of dangers to the timely satisfaction of inactive goals due to unexpected changes in the environment (section 5.2.3). For this reason, a demon is implemented as a function that generates an event that is reported to the relevant process under certain conditions; it does not evaluate to a simple boolean as specified in definition 4.8. A demon will evaluate to `NoEvent` if the conditions that characterise that demon do not hold.

A `goalid` uniquely identifies a particular goal that is either a candidate for generation in the case of `ConsiderGoal`, or an existing goal in all other cases. A `belief` is a valid proposition in the world model. At the start of every cycle (section 3.1.5) the changes in the agent's beliefs are checked, and if a demon fires the relevant event is reported. (The use of the predefined function `filter` in the function `check_demons` is to simply remove any `NoEvent` returned by a demon.)

```
> event ::= ConsiderGoal [belief] goalid |
>           TriggerRGoal goalid |
>           DeleteRGoal goalid |
>           AddOpportunity action goalid |
>           DelOpportunity action goalid |
>           AddDanger appropriate goalid |
>           DelDanger appropriate goalid |
>           NoEvent

> demon == [belief] -> event

> check_demons :: [demon] -> [belief] -> [event]
> check_demons ds bs = filter (~=NoEvent) [d bs | d<-ds]
```

If a `ConsiderGoal` event occurs, the agent will decide whether or not to generate a goal in response. If a `TriggerRGoal` event occurs, a replenishment process is fired which generates a goal automatically according to some specification (see section B.2.7). If a `DeleteRGoal` event occurs, the replenishment process indicated by the identifier `goalid` is deleted along with this demon and the demon that generates relevant `TriggerRGoal` events. If an `AddOpportunity` event occurs, the action associated with this opportunity is added to the alarm structure that encapsulates the goal with identifier `goalid`. Only when an `AddOpportunity` event has occurred, may the reciprocal `DelOpportunity` demon fire. If an opportunity to achieve the goal with the unique identifier `goalid` no longer exists (i.e. a `DelOpportunity` event has occurred), the action associated with this opportunity is deleted from the relevant alarm structure. (Note, if the alarm has subsequently triggered and been mitigated, the opportunity will have already been deleted, see page 115.) If an `AddDanger` or `DelDanger` event occurs, the appropriateness condition `appropriate` (see section B.2.4) is added or deleted as a possible influence on the alarm indicated by the identifier `goalid`.

B.2.6 The alarm and mitigate functions

For an alarm to be generated, the agent must determine the values of the variables that specify the delay time, an estimate of the time required to achieve the goal, the deadline, the alarm's minimum intensity and the importance of the goal. With this information, an alarm function is defined following definition 4.7.

```
> intensity == num

> alarm_function :: delaytime -> duration -> deadline ->
>                   importance -> time -> intensity
> alarm_function dt dur dl imax t
> = 0,                                              before t dt
> = imax,                                           ~before t tmax
> = (divide (minus t dt) (minus tmax dt)) * imax, otherwise
>   where
>   tmax = minus dl dur
```

For an alarm to be mitigated, the agent must determine the time at which the mitigation should cease, `toff`, and the present intensity of the alarm: i.e. the intensity of the alarm function, `alfn`, at the time at which the alarm was mitigated, `ton`, but taking into account the time shift due to dangers and time commitments, `shift`. With this information, a mitigation function is defined following definition 6.4.

```
> mitigation_function :: (time -> intensity) -> time -> time ->
>                        time -> time -> intensity
> mitigation_function alfn ton toff shift t
> = 0,                                before t' ton' \/ ~before t' toff'
> = (-i) * (divide (minus toff' t') (minus toff' ton')), otherwise
>   where
>   i = alfn ton'
>   t' = plus t shift
>   ton' = plus ton shift
>   toff' = plus toff shift
```

B.2.7 Replenishment

Replenishment is an automatic process that is triggered when an appropriate event of type `TriggerRGoal` occurs. A replenishment process will either generate a goal with a deadline and delay time set a fixed period of time after the time that the replenishment process is invoked, or according to some timetable.

```
> replenishtype ::= Periodic time time | Timetabled [time] time
```

Consider a periodic replenishment process that is invoked at 9am on some Monday. The `replenishtype` of this R-Goal template specifies the period of time from the time now at which the deadline and delay time of the replenished goal should be set. Suppose that `egrepl1` specifies the temporal characteristics of this periodic replenishment process. When this replenishment process is invoked at 9am on some Monday the deadline of the replenished goal will be

set to 9am the following day give or take an hour (i.e. about a day later), and the delay time will be set to 3pm on Monday give or take an hour (i.e. about six hours later).

```
> egrepl1 :: replenishtype
> egrepl1 = Periodic (maketime ((0,1,0,0,0),(0,0,1,0,0)))
>                    (maketime ((0,0,6,0,0),(0,0,1,0,0)))
```

Consider a timetabled replenishment process that is invoked at 9am on some Monday. The `replenishtype` of this R-Goal template specifies the times at which deadlines should be set, and the period of time before that deadline that the delay time should be set. Suppose that `egrepl2` specifies the particular times of the week at which the deadline of a replenished goal should be set, and how the delay time can be found. In this example, the goal should be replenished with either a deadline set at 5pm on the third day of the week (i.e. Wednesday), or 11am on the Saturday depending on the circumstances in which it is invoked. If it is invoked at 9am on some Monday, the deadline of the replenished goal will be set to 5pm on the Wednesday of the same week give or take ten minutes, and so the delay time will be set to 5pm on Monday give or take about six hours.

```
> egrepl2 :: replenishtype
> egrepl2 = Timetabled [maketime ((0,3,17,0,0),(0,0,0,1,0)),
>                       maketime ((0,6,11,0,0),(0,0,0,1,0))]
>                        (maketime ((0,2,0,0,0),(0,0,6,0,0)))
```

The only difference between two different goals that are replenished by the same replenishment process at different times are their deadlines and delay times. An `rgoaltemplate` contains a partially instantiated goal (`delaytime` and `deadline` are set to `zerotime` by default), a `replenishtype`, a list of conditions under which the replenishment of goals according to this template is appropriate, a list of anticipated opportunities, and a list of appropriateness conditions. The function `find_dt_dl` is used to generate the deadline and delay time for a replenished goal when that replenishment process is invoked at the time `now`.

```
> rgoaltemplate == (goal, replenishtype, [condition],
>                   [anticipated], [appropriate])

> find_dt_dl :: time -> replenishtype -> (delaytime, deadline)
> find_dt_dl now (Periodic x y) = (plus now y, plus now x)
> find_dt_dl now (Timetabled [xs] y)
> = (minus dl y, dl)
>   where
>   dl = infertime now xs
```

The implemented meta-replenishment functions are limited. The prototype agent is capable of generating and deleting replenishment processes, but not able to modify periods of replenishment or other characteristics through experience. The generation of replenishment processes are

implemented as part of goal generation, and triggered by a `ConsiderGoal` event. The deletion of a replenishment process is triggered by a `DeleteRGoal` event.

B.2.8 Alarm processing

An `alarm` is a seven tuple consisting of a goal, a list of appropriateness conditions, and alarm function, a list of detected opportunities to satisfy the encapsulated goal, a time shift due to detected dangers to the timely satisfaction of that goal, a time shift due to the effects of the time commitments that the agent has made, and a mitigation function. The `alarm` structure is implemented as an abstract type, with its signature given below.

```
> abstype alarm
> with get_alarm_goal    :: alarm -> goal
>      get_alarm_id      :: alarm -> goalid
>      generate_alarm    :: (goal, [appropriate]) -> alarm
>      replenish_alarm   :: time -> rgoaltemplate -> alarm
>      alarm_intensity   :: alarm -> time -> intensity
>      mitigate_alarm    :: time -> alarm -> alarm
>      add_opportunity   :: alarm -> opportunity -> alarm
>      del_opportunity   :: alarm -> opportunity -> alarm
>      danger_shift      :: alarm -> plan -> alarm
>      commitment_shift  :: alarm -> [commitment] -> alarm

> alarm == (goal, [appropriate], (time->intensity),
>           [opportunity], shift, shift, (time->time->intensity))

> shift == time
```

The generation of an alarm requires a fully instantiated goal, and a set of appropriateness conditions for the temporal relevance of that goal. The alarm function, `alfn`, is set according to the delay time, duration, deadline and importance of the goal. Initially, there is an empty list of opportunities to satisfy the goal (although opportunity demons are set at the same time as the alarm is generated), no time shifts due to dangers (although again danger demons are set at this time) or time commitments, and the mitigation function is zero for all time (`zerofunction`).

```
> generate_alarm (g, apps)
> = (g,apps,alfn,[],zerotime,zerotime,zerofunction)
>   where
>   alfn = alarm_function (get_goal_dt g) (get_goal_dur g)
>                         (get_goal_dl g) (get_goal_imp g)

> zerofunction :: time -> time -> intensity
> zerofunction t u = 0
```

A goal is replenished on the basis of a partially instantiated goal, `g`, a replenishment type containing information about how the deadline and delay time of the replenished goal should be set, `rt`, and the appropriateness conditions of a goal replenished according to this specification, `apps`. The goal is instantiated and the alarm function defined with the delay time and

deadline determined from the function `find_dt_dl`. (Danger and opportunity demons are set seperately.)

```
> replenish_alarm now (g, rt, cs, ant, apps)
> = (instantiatedgoal,apps,(alarm_function dt dur dl imp),
>    [],zerotime,zerotime,zerofunction)
>   where
>   instantiatedgoal = put_goal_dt dt (put_goal_dl dl g)
>   (dt, dl) = find_dt_dl rt now
>   dur = get_goal_dur g
>   imp = get_goal_imp g
```

The intensity of the alarm depends on whether there are opportunities, `opps`, to satisfy the goal, `g`, and the levels of the alarm function, `alfn`, and mitigation function, `mfn`, at the present time relative to any time shift due to dangers, `dan`, or time commitments, `tc`. If the current intensity of the sum of the alarm and mitigation functions is above zero and the agent has detected one or more opportunity to satisfy that goal, the intensity of the alarm is set to maximum. Otherwise, the alarm is the sum of the current values of the two functions of intensity, `alfn` and `mfn`.

```
> alarm_intensity :: alarm -> time -> intensity
> alarm_intensity (g, apps, alfn, opps, dan, tc, mfn) now
> = get_goal_imp g,      (currentintensity > 0) & (opps ~= [])
> = currentintensity,    otherwise
>   where
>   currentintensity = (alfn sampletime) + (mfn sampletime)
>   sampletime = minus now (plus dan tc)
```

The `mitigate_alarm` function generates a mitigation function that decreases to minus the present alarm intensity at the present time and increases to zero at some time determined by the function `end_of_mitigation`. On mitigation, all opportunities to satisfy the goal are deleted; it is assumed that they have been considered and rejected.

The period of mitigation (determined by the end of the mitigation) will decrease to some time `min_mitigation` as t_{max} approaches, and increase to `max_mitigation` after t_{max} (see definition 6.5). The variables `min_mitigation` and `max_mitigation` may be altered as required. As presented, they represent a minimum mitigation duration of around ten minutes and a maximum duration after t_{max} of around an hour.

```
> mitigate_alarm now (g, apps, alfn, opps, dan, tc, mfn)
> = (g, apps, alfn, [], dan, tc,
>    (mitigation_function alfn now (end_of_mitigation now tmax)))
>   where
>   tmax = minus (get_goal_dl g) (get_goal_dur g)

> min_mitigation :: time
> min_mitigation = maketime ((0,0,0,1,0),(0,0,0,0,1))
```

```
> max_mitigation :: time
> max_mitigation = maketime ((0,0,1,0,0),(0,0,0,1,0))

> end_of_mitigation :: time -> time -> time
> end_of_mitigation now tmax
> = plus now min_mitigation, before now tmax &
>                            difference <= min_mitigation
> = plus now max_mitigation, before tmax now &
>                            difference >= max_mitigation
> = tmax,                    otherwise
>   where
>   difference = positive tmax now

> positive :: time -> time -> time
> positive tmax now
> = minus tmax now, before now tmax
> = minus now tmax, otherwise
```

The functions reproduced below for computing the time shift on an alarm due to a detected danger to the timely satisfaction of an inactive goal follow definition 5.2. The appropriateness conditions of a particular alarm, `apps`, are used to distinguish dangers due to intended and unintended effects of planned actions using the function `dangers`, section B.2.4. Then, depending on when the dangerous actions are scheduled and the extent of the causal link in the case of an intended effect, and on when the goal encapsulated in the alarm is scheduled, the extent of the time shift due to this danger can be computed. The idea behind these functions is to determine the period of time before t_{dl} that the alarm should reach maximum so that it does so a period $\Delta_a t$ before any dangerous action is taken. If there are more than one dangerous effects on the inactive goal due to the agent's plan, the maximum time shift computed is used, `(maxtime shfts)`.

```
> danger_shift (g, apps, alfn, opps, old, tc, mfn) p
> = (g, apps, alfn, opps, (maxtime shfts), tc, mfn)
>   where
>   shfts = (map (shift_intended_danger tmax dl) intended) ++
>           (map (shift_unintended_danger tmax dl) unintended))
>   tmax = minus dl (get_goal_dur g)
>   dl = get_goal_dl g
>   (intended, unintended) = dangers p apps

> shift_intended_danger :: time -> deadline ->
>                          (appropriate, link) -> time
> shift_intended_danger tmax dl ((p1,t1),(a,p2,t2))
> = zerotime,               before t2 (minus tmax t1) \/
>                           before dl (minus aend adur)
> = dan_shift dl aend adur, otherwise
>   where
>   aend = get_act_end a
>   adur = get_act_dur a

> shift_unintended_danger :: time -> deadline ->
>                            (appropriate, link) -> time
```

```
> shift_unintended_danger tmax dl ((p1,t1),(a,p2,t2))
> = zerotime,                 before aend (minus tmax t1) \/
>                             before dl (minus aend adur)
> = dan_shift dl aend adur, otherwise
>   where
>   aend = get_act_end a
>   adur = get_act_dur a

> dan_shift :: deadline -> time -> duration -> time
> dan_shift dl aend adur
> = plus (minus dl aend) adur,     before aend dl
> = minus (adur (minus aend dl)), before dl aend
> = adur,                          otherwise
```

The function `commitment_shift` is used to determine the time shift for an alarm due to the effects of the agent's prior commitments made during planning. If the goal is scheduled so that t_{max} is after the last action in the present plan is to be performed, there is no need to shift the alarm function. (Note that the time commitments returned by the function `time_commitments` are ordered with the latest commitment first.) However, if this is not the case a gap in the agent's commitments must be found in which the goal can be achieved. The goal that is encapsulated in the alarm is represented as a time commitment from t_{max} to t_{dl}. This interval is shifted back in time by the function `find_gap` until there is a period of time between the agents time commitments that can accommodate this period. (Note, the function `plus` is used as an infix operator in the function `find_gap` using the symbol `$` as a prefix to the function name.)

```
> commitment_shift (g, apps, alfn, opps, dan, old, mfn) cs
> = (g, apps, alfn, opps, dan, zerotime, mfn), before (end c) dl
> = (g, apps, alfn, opps, dan, shft, mfn),     otherwise
>   where
>   shft = plus (minus dl startc) (find_gap cs (dur, startc))
>   dur = get_goal_dur g
>   dl = get_goal_dl g
>   startc = start c
>   c = hd cs

> find_gap :: [commitment] -> commitment -> time
> find_gap [c] x = zerotime
> find_gap (c:cs) old
> = zerotime,                    before (end c) startold
> = shift $plus find_gap cs new, otherwise
>   where
>   new = (fst old, minus (snd old) shift)
>   shift = minus startold (start c)
>   startold = start old

> end = snd
> start x = minus (snd x) (fst x)
```

B.2.9 A prototype motivated agent

The prototype motivated agent operates by calling the function `run_agent`. This function requires a list of states, a list of replenishment processes and the initial alarms. A state is a tuple with fields of type `threshold`, `plan`, `[belief]` and `time`. This represents the threshold, plan and updates to the agent's beliefs at a particular time.

```
> state == (threshold, plan, [belief], time)
> threshold == intensity
> belief == proposition
```

The function `run_agent` evaluates to a list of characters; this list of characters corresponding to the output presented in appendix C. The function operates as follows:

- The functions `getnewgoals` and `generate_alarm` are used to generate new goals and encapsulate them in appropriate alarm structures in response to the list of beliefs that have altered in the present state. These beliefs are also checked to see if a goal that was generated through replenishment has been deleted; if so, a new alarm is replenished through the functions `getdeleted` and `replenish_alarm`. These newly generated and replenished alarms, and the existing alarms are then further processed in the following way:
  - Each alarm is checked to see if there are any time commitments or dangers that must be taken into account due to the agent's plan, `p`. The alarm structure is updated with any time shift required.
  - Each alarm is checked to see if there are any opportunities to satisfy the goal it encapsulates or if there are any dangers due to changes in the external environment. The alarm structure is updated with any opportunity or danger detected.
  - Each alarm is then evaluated at the present time and tested to see if the intensity of the alarm exceeds the present threshold level. The function `trigger` splits the list of alarms into a list of those that have triggered and a list of those that have not, and hence will not be considered at this time.
- The alarms that have triggered are then considered to determine whether or not they will be activated. Those that will be mitigated are returned by the function `getmitigated`, and the mitgation function within the alarm structure is updated by the function `mitigate_alarm` for each of the alarms that are to be mitigated.
- The present state of the alarm processing machiery is displayed. The list of triggered

alarms, those that have been mitigated, and those that were not considered at this time are shown.

- Finally, `runagent` is called with the rest of the states that it is to be processes with. If there are no states left, the function terminates.

```
> runagent :: [(threshold, plan, [belief], time)] -> [rgoal]
>                                      -> [alarm] -> [char]
> runagent [] rgs as = ""
> runagent ((thd,p,s,now):rest) rgs as
> = "\nNow = " ++ (showtime now) -- "\tThreshold = " -- (show thd) --
>   "\nAlarms triggered\n" ++ (showalarms trig now) ++
>   "\nAlarms mitigated\n" ++ (showalarms mit now) ++
>   "\nAlarms not considered\n" -- (showalarms nc now) --
>   runagent rest rgs (nc ++ mit)
>   where
>   mit = map (mitigate_alarm now) (getmitigated trigg s)
>   (nc, trig) = next_step as thd p s rgs now

> next_step :: [alarm] -> threshold -> plan -> [belief] ->
>              [rgoaltemplate] -> time -> ([alarm],[alarm])
> next_step as thd p s rgs now
> = ((trigger thd now).(impulses bs).(timeoffset p))
>     ((map (replenish_alarm now) (getdeleted rgs bs)) --
>      (generate_alarm (getnewgoals bs)) ++ as)

> trigger :: threshold -> time -> [alarm] -> ([alarm],[alarm])

> impluses :: [belief] -> [alarm] -> [alarm]

> timeoffset :: plan -> [alarm] -> [alarm]

> getdeleted :: [rgoaltemplate] -> [belief] -> [rgoaltemplate]

> getnewgoals :: [belief] -> [(goal,[appropriate])]
```

Appendix C

Simulation of the alarm processing machinery

This appendix presents and discusses a simulation of the alarm processing machinery for an agent operating in the warehouse domain. This simulation uses the Miranda code described in appendix B, and the output presented in this appendix is the output produced by that code. The consideration of goals, and the planner, effector and sensor functionality that would be included in a complete agent are hand simulated.

Points and intervals of time are modeled as a number of minutes, tens of minutes, hours, days and weeks with an appropriate accuracy measured in the same terms. The agent is initialised with a number of replenishment processes. The characteristics of the goals replenished are captured in tuples: an rgoal is a 4-tuple that contains an uninstantiated goal of the type that will be replenished, a time from which the delay time can be calculated, an enumerated type `rtype` that captures the type of replenishment process and a list of conditions under which the rgoal will be deleted.

```
> rgoal == (goal, time, rtype, [condition])
> goal == (goalid, importance, delaytime, duration, deadline)
> rtype ::= Periodic time   Timetabled [time]
```

The warehouse agent is initialised with a number of structures of the type `rgoal`, some of which are given below. The function `template` returns an uninstantiated goal template of the required type, and the function `maketime` generates an object of the abstract type `time`. The replenishment goal template below represents a regular order that the agent has agreed with customer 2. The agent has agreed to prepare orders consisting of 5 units of the commodity Salsa, 5 units of Coffee and 10 units of Yoghurt, each Monday at 10am and each Thursday at 10am. Suppose that the state of the agent is such that a goal is replenished according to this specification at 1:35pm on the Thursday of week 3 (i.e. at ((3,4,13,3,5),(0,0,0,0,1))). The deadline of the replenished goal will be set to the next Monday or Thursday at 10am, and the delay time is set to 12 hours before this deadline (this is specified by the second field in the rgoal 4-tuple illustrated below). In this case the deadline will be ((4,1,10,0,0),(0,0,1,0,1)) and the delay time will

be ((4,0,22,0,0),(0,0,2,0,1)). The set of conditions in which this R-Goal is deleted are tested with the functions `notrequired` and `notreliable`: these simply test to see if the customer has cancelled the order or is considered unreliable.

```
> (template (Prepare order2),
>   maketime ((0,0,12,0,0),(0,0,1,0,0)),
>   Timetabled [maketime ((0,1,10,0,0),(0,0,1,0,0)),
>               maketime ((0,4,10,0,0),(0,0,1,0,0))],
>   [notrequired order2, notreliable customer2])

> order2 :: order
> order2
> = ("customer 2", zerotime, [(Salsa, 5),
>                             (Coffee, 5),
>                             (Yoghurt, 10)])
```

The following five R-Goal template examples can never be deleted: the function "invariant" evaluates to the boolean symbol False.

```
> (template (Curious (Temp A)),
>   maketime ((0,0,2,0,0),(0,0,0,1,0)),
>   Periodic (maketime ((0,0,4,0,0),(0,0,0,1,0))),
>   [invariant])

> (template (Curious OrderBook),
>   maketime ((0,0,1,0,0),(0,0,0,1,0)),
>   Periodic (maketime ((0,0,2,0,0),(0,0,0,1,0))),
>   [invariant]),

> (template (Restock Pizza),
>   maketime ((0,2,0,0,0),(0,0,1,0,0)),
>   Timetabled [maketime ((0,1,0,0,0),(0,0,1,0,0)),
>               maketime ((0,4,0,0,0),(0,0,1,0,0))],
>   [invariant])

> (template (Restock CatFood),
>   maketime ((0,2,0,0,0),(0,0,1,0,0)),
>   Timetabled [maketime ((0,4,0,0,0),(0,0,1,0,0))],
>   [invariant])

> (template (Tidy E),
>   maketime ((0,0,12,0,0),(0,0,1,0,0)),
>   Periodic (maketime ((0,2,0,0,0),(0,0,1,0,0))),
>   [invariant])
```

The simulation follows a 24 hour period in the warehouse scenario from 6am on the Monday of week 1 to 5am on the Tuesday of week 1. The interval at which the alarm functions of the inactive goals are evaluated varies from one hour to 5 minutes. These intervals are chosen specifically to highlight certain types of behaviour of the alarm processing machinery. Each state of the alarm functions is generated automatically: the resulting data is presented in the format produced by the simulation.

C.1 Initial state

The following program output displays the initial state of the agent's active and inactive goals. The agent has a single active goal to have tidied room C of the warehouse (see figure 2.2 on page 26), and 28 inactive goals that are encapsulated in appropriate alarm structures. Each entry shows the identifier of the goal (a unique identifier), the importance of the goal (an integer), the delay time, travel time and deadline of the goal, each being objects of type `time`. Of these 29 goals, 6 have been generated through decision, and 23 through replenishment. The six generated through decision are the goals to have prepared orders 4–9.

Active goals

```
Tidy C, 5, Sat 21:00 wk0, 2.0hrs, Mon 9:00
```

Inactive goals

```
Prepare order 4, 10, Mon 9:00, 2.0hrs, Mon 12:00
Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
Curious (Temp A), 8, Mon 4:00, 10mins, Mon 6:00
Curious (Temp B), 8, Mon 5:00, 10mins, Mon 7:00
Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30
Curious (Temp D), 8, Mon 9:00, 10mins, Mon 11:00
Curious OrderBook, 8, Sun 18:00, 20mins, Sun 19:00
Restock FrozenPizza, 7, Sat 0:00 wk0, 30mins, Mon 0:00
Restock IceCream, 7, Sat 0:00 wk0, 30mins, Mon 0:00
Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
Restock Yoghurt, 7, Sun 0:00, 30mins, Tue 0:00
Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
Restock TortillaChips, 7, Sat 0:00 wk0, 30mins, Mon 0:00
Restock Coffee, 7, Mon, 30mins, Wed
Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
Recharge, 10, Mon 8:00, 30mins, Mon 10:00
Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
Prepare order 2, 10, Sun 22:00, 1.0hrs, Mon 10:00
Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

Each of sections C.2–C.28 shows the state of the agent's alarms at a particular time point. At each time point, an alarm may trigger. If the alarm triggers the goal may be activated or the alarm mitigated. If the alarm does not trigger, the goal is not considered. Therefore, the state of the alarm processing machinery at each time point may be represented by the set of alarms that have triggered, been mitigated, and not considered. The simulation of the alarm processing machinery generates output at every time point specified in the following form. Each alarm is represented by its intensity at the time indicated (this is the number in brackets), the identifier

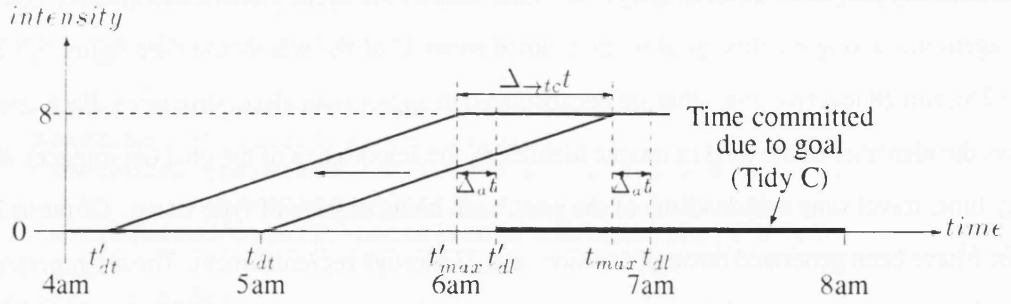


Figure C.1: The effect of a prior time commitment on the alarm encapsulating the goal to have mitigated curiosity of the temperature of room B.

of the goal, its importance, delay time, travel time, deadline, any time shift due to dangers and time commitments and any opportunities that have been detected.

C.2 Time = Mon 6:00, τ=5

The simulation starts at 6am on the Monday of week 1. At this time, the threshold (τ) is set to 5, and the goal to have tidied room C is being actively pursued by the agent.

Seven alarms trigger at this point. The agent then considers activating (i.e. passing to the planner) the goals to have mitigated curiosity about the temperature is rooms A and B, the state of the order book, to have restocked the warehouse with frozen pizza, ice cream and tortilla chips and to have prepared order number 2.

The agent's plan states that it intends to have tidied room C by ten minutes past eight and that achieving this goal is predicted to take two hours (see figure C.1). This time commitment conflicts with the goal to have checked the temperature of room B because the agent cannot achieve this goal before 7am (t_{dl}) while tidying room C. A time shift ($\Delta_{\rightarrow tc}t$) of 50 minutes is therefore required to ensure that the agent is reminded of the goal to have checked the temperature of room B so that both goals may be achieved in time; i.e. before t'_{max} (see figure C.1 and section 5.3). (Note that in this simulation it is not possible for the agent to take into account the possibility of interrupting an active goal in calculating the time shift of an alarm due to prior time commitments. The agent may realise when both goals are active that the goal to have tidied room C can be interrupted so that it may check the temperature of room B. It is possible to flag goals that may be interrupted, and use this information to generate a better estimate of time shifts due to prior time commitments, but at increased cost in alarm processing.)

Of the seven triggered alarms, three are mitigated and four activated. The agent's focus

of planning attention is directed towards checking the order book and restocking the warehouse with frozen pizza, ice cream and tortilla chips. The agent decides to delay activating the other three goals.

Alarms triggered

```
(8) Curious (Temp A), 8, Mon 4:00, 10mins, Mon 6:00
(8) Curious (Temp B), 8, Mon 5:00, 10mins, Mon 7:00, 50mins
(8) Curious OrderBook, 8, Mon 5:00, 20mins, Mon 6:00
(7) Restock FrozenPizza, 7, Sat 0:00 wk0, 30mins, Mon 0:00
(7) Restock IceCream, 7, Sat 0:00 wk0, 30mins, Mon 0:00
(7) Restock TortillaChips, 7, Sat 0:00 wk0, 30mins, Mon 0:00
(7.27) Prepare order 2, 10, Sun 22:00, 1.0hrs, Mon 10:00
```

Alarms mitigated

```
(0) Curious (Temp A), 8, Mon 4:00, 10mins, Mon 6:00
(0.0) Curious (Temp B), 8, Mon 5:00, 10mins, Mon 7:00, 50mins
(0.0) Prepare order 2, 10, Sun 22:00, 1.0hrs, Mon 10:00
```

Alarms not considered

```
(0) Prepare order 4, 10, Mon 9:00, 2.0hrs, Mon 12:00
(0) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
(0) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30
(0) Curious (Temp D), 8, Mon 9:00, 10mins, Mon 11:00
(1.79) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(4.42) Restock Yoghurt, 7, Sun 0:00, 30mins, Tue 0:00
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0.88) Restock Coffee, 7, Mon, 30mins, Wed
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(1.76) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(3.82) Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
(0) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Recharge, 10, Mon 8:00, 30mins, Mon 10:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

C.3 Time = Mon 6:05, τ=5.5

At five minutes past six, the threshold has risen to 5.5 due to the increased number of active goals and the rate of alarm triggering (hand simulated). Despite this, four alarms trigger. The goal to have checked the temperature of room A triggers again because the period of mitigation is short when the deadline of a goal is reached (the deadline is at 6am). Compare this to the intensity of the alarm encapsulating the goal to have checked the temperature in room B. This alarm was mitigated at the same time, but due to their different deadlines their periods of mitigation are different (see section 6.2.3).

The agent also considers activating the goals to have restocked the warehouse with salsa, yoghurt and coffee. At 6am, the intensity of the alarms encapsulating these goals was $\sim$1.8.

$\sim$4.4 and $\sim$0.9 respectively, but the fact that the agent activated the goals to have restocked the warehouse with frozen pizza, ice cream and tortilla chips at 6pm constitutes an opportunity to satisfy these goals: the agent can place a bulk order for all commodities required including salsa, yoghurt and coffee. However, the agent is not reminded of the goal to have restocked the warehouse with cat food: this is because this goal has a delay time of midnight on Tuesday, and hence (according to the agent's predictions) it is not relevant for consideration at the present time.

The agent considers the four goals associated with the alarms that have triggered and decides to activate only the goal to have restocked the warehouse with yoghurt. The decision not to activate the goals to have restocked the warehouse with salsa and coffee may be due to space limitations in the warehouse for example.

Alarms triggered

```
(7) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00, PlaceOrder Salsa
(7) Restock Yoghurt, 7, Sun 0:00, 30mins, Tue 0:00, PlaceOrder Yoghurt
(7) Restock Coffee, 7, Mon, 30mins, Wed, PlaceOrder Coffee
(8) Curious (Temp A), 8, Mon 4:00, 10mins, Mon 6:00
```

Alarms mitigated

```
(0.0) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00, PlaceOrder Salsa
(0.0) Restock Coffee, 7, Mon, 30mins, Wed, PlaceOrder Coffee
(0.0) Curious (Temp A), 8, Mon 4:00, 10mins, Mon 6:00
```

Alarms not considered

```
(0) Prepare order 4, 10, Mon 9:00, 2.0hrs, Mon 12:00
(0) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
(0) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30
(0) Curious (Temp D), 8, Mon 9:00, 10mins, Mon 11:00
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(1.78) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(3.84) Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
(0) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Recharge, 10, Mon 8:00, 30mins, Mon 10:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(0.67) Curious (Temp B), 8, Mon 5:00, 10mins, Mon 7:00, 1.5hrs
(0.23) Prepare order 2, 10, Sun 22:00, 1.0hrs, Mon 10:00
```

C.4 Time = Mon 7:00, τ=5.5

At 7am, the agent is again reminded of the goals to have checked the temperatures in room A and B, but again due to the time taken up pursuing the goals that are presently active, the agent decides to consider them for activation some time later. Note that this time (cf. section C.3), the

period of mitigation for A will be greater than that for B due to their respective deadlines (see section 6.2.3).

Alarms triggered

```
(8) Curious (Temp B), 8, Mon 5:00, 10mins, Mon 7:00
(8) Curious (Temp A), 8, Mon 4:00, 10mins, Mon 6:00
```

Alarms mitigated

```
(0) Curious (Temp B), 8, Mon 5:00, 10mins, Mon 7:00
(0) Curious (Temp A), 8, Mon 4:00, 10mins, Mon 6:00
```

Alarms not considered

```
(0) Prepare order 4, 10, Mon 9:00, 2.0hrs, Mon 12:00
(0) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
(0) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30
(0) Curious (Temp D), 8, Mon 9:00, 10mins, Mon 11:00
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(1.91) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(3.97) Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
(0) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Recharge, 10, Mon 8:00, 30mins, Mon 10:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(2.73) Prepare order 2, 10, Sun 22:00, 1.0hrs, Mon 10:00
(0.37) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(0.15) Restock Coffee, 7, Mon, 30mins, Wed
```

C.5 Time = Mon 8:00, τ=5

Again at 8am the agent is reminded of the previously mitigated goals to have checked the temperature of rooms A and B. At this point the agent has achieved the goals to have restocked the warehouse with various commodities, and has checked the order book. The agent has also nearly completed satisfying the goal to have tidied room C. Therefore, it decides to redirect attention towards checking the temperature in rooms A and B and preparing order 2; the three alarms triggered at this time. The agent does not mitigate any of these triggered alarms.

The fact that the agent has satisfied its goals to have restocked the warehouse with frozen pizza, ice cream, tortilla chips and yoghurt, and to have checked the order book causes goals of the same type, but with different temporal contexts (i.e. according to definition 4.1, three different goals) to be replenished. In addition to these new alarms, the agent in mitigating its curiosity about the state of the order book has noticed that two new orders have arrived. The agent decides to generate two new alarms encapsulating goals to have prepared orders 10 and 11 (the

requested orders). The agent is now influenced by 27 alarms encapsulating inactive goals and four active goals.

Alarms triggered

```
(5.45) Prepare order 2, 10, Sun 22:00, 1.0hrs, Mon 10:00
(8) Curious (Temp B), 8, Mon 5:00, 10mins, Mon 7:00
(8) Curious (Temp A), 8, Mon 4:00, 10mins, Mon 6:00
```

Alarms mitigated

Alarms not considered

```
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Curious OrderBook, 8, Mon 9:00, 20mins, Mon 10:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 4, 10, Mon 9:00, 2.0hrs, Mon 12:00
(0) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
(0) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30
(0) Curious (Temp D), 8, Mon 9:00, 10mins, Mon 11:00
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.06) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(4.12) Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
(0) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0.0) Recharge, 10, Mon 8:00, 30mins, Mon 10:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(0.76) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(0.32) Restock Coffee, 7, Mon, 30mins, Wed
```

C.6 Time = Mon 9:00, τ=6

At 9am the agent has planned to have order 2 prepared by 10am (i.e. when the customer is expected to arrive), and to have checked the temperatures in rooms A and B before preparing this order. The agent also has an inactive goal to have prepared an order for a customer that is expected to arrive at 12 noon. The agent had predicted that it will take a couple of hours (give or take a few tens of minutes) to prepare that order. The alarm processing machinery is designed to use the predictions of future events made available by the goal generation processes in a conservative manner. Therefore, according to the alarm processing machinery, the two hour period between 10am and 12am is not considered sufficient for the agent to be certain that it will have time to prepare order 4: according to its predictions, preparing order 4 may take ten or twenty minutes more than two hours. (This conservative behaviour can be modified by modifying the

way inferences are made from temporal knowledge, see section 4.2.) Hence the alarm is shifted by three hours. This means that the deadline is effectively 9am rather than 12am, and so the alarm is at maximum intensity. The other triggered alarms are also each shifted by one hour, but the agent decides only to activate the goal to have recharged.

At some time between 8 and 9am the agent will have finished tidying room C. The goal is deleted, and hence a new goal replenished.

Alarms triggered

```
(8) Curious OrderBook, 8, Mon 9:00, 20mins, Mon 10:00, 1.0hrs
(10) Prepare order 4, 10, Mon 9:00, 2.0hrs, Mon 12:00, 3.0hrs
(10) Recharge, 10, Mon 8:00, 30mins, Mon 10:00, 1.0hrs
```

Alarms mitigated

```
(0.0) Curious OrderBook, 8, Mon 9:00, 20mins, Mon 10:00, 1.0hrs
```

Alarms not considered

```
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0.0) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
(0.0) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30
(0.0) Curious (Temp D), 8, Mon 9:00, 10mins, Mon 11:00
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.21) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(4.26) Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
(0) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(1.16) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(0.49) Restock Coffee, 7, Mon, 30mins, Wed
```

C.7 Time = Mon 9:30, τ=7

In satisfying the goal to have recharged that was activated at 9am, the agent must visit the recharge point, which is situated in the office. The order book is also in the office, and so the agent's present location constitutes an opportunity to satisfy the goal of checking the order book. (The agent does not need to make a special journey to the office.) The agent decides to activate this goal in the light of the opportunity. Neither of the goals to have checked the temperatures of rooms C and D are activated.

Alarms triggered

```
(8) Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30, 1.83hrs
(8) Curious (Temp D), 8, Mon 9:00, 10mins, Mon 11:00, 1.33hrs
(8) Curious OrderBook, 8, Mon 9:00, 20mins, Mon 10:00, Sample OrderBook
```

Alarms mitigated

```
(0.0) Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30, 1.83hrs
(0.0) Curious (Temp D), 8, Mon 9:00, 10mins, Mon 11:00, 1.33hrs
```

Alarms not considered

```
(0) Curious (Temp A), 8, Mon 11:30, 10mins, Mon 13:30
(0) Curious (Temp B), 8, Mon 11:30, 10mins, Mon 13:30
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0.71) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
(0.59) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.28) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(4.34) Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
(0) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(1.36) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(0.58) Restock Coffee, 7, Mon, 30mins, Wed
```

C.8 Time = Mon 10:00, τ=6.5

At some time between 9.30am and 10am the agent checks the order book. This leads to that agent making a decision to generate a new goal to prepare order 12. The goal to have checked the order book is then deleted and subsequently replenished so that the agent will again be reminded to check the order book before 12 noon. However, the agent has planned to act on the goal to have prepared order 4 so that it may be collected by the customer at around 12 noon. This, and the agent's other commitments influence the intensity of this newly replenished goal. Despite the fact that the delay time of this goal is 11am, the agent is reminded of it due to a time shift of 2.33 hours. At this time the goal is not activated, but this behaviour illustrates the potential effect of prior time commitments on an agent's alarms. Prior time commitments also have an effect on the alarms encapsulating the goals to have checked the temperatures in rooms C and D. The alarms are both shifted by 1.83 hours. However, this is not sufficient to cause either of these alarms to trigger at this time.

The agent has also achieved the goal to have prepared order 2. This goal is deleted, and because this is a regular order, a replenishment process automatically generates a new goal which is encapsulated in an appropriate alarm.

Alarms triggered

```
(8) Curious OrderBook, 8, Mon 11:00, 20mins, Mon 12:00, 2.33hrs
```

Alarms mitigated

```
(0.0) Curious OrderBook, 8, Mon 11:00, 20mins, Mon 12:00, 2.33hrs
```

Alarms not considered

```
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Prepare order 12, 8, Mon 16:30, 20mins, Mon 18:00
(0) Curious (Temp A), 8, Mon 11:30, 10mins, Mon 13:30
(0) Curious (Temp B), 8, Mon 11:30, 10mins, Mon 13:30
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(1.43) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
(1.18) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.35) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(4.41) Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
(0) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(1.56) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(0.66) Restock Coffee, 7, Mon, 30mins, Wed
(0.62) Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30, 1.83hrs
(2.67) Curious (Temp D), 8, Mon 9:00, 10mins, Mon 11:00, 1.33hrs
```

C.9 Time = Mon 11:00, τ=7.5

At this time the threshold has risen to 7.5 due to the rate of alarm triggering and the agent's plan of action. This plan expresses the agent's intention to have prepared order 4 by midday. The agent is reminded of its goals to have checked the order book and the temperature of room D, both are activated.

Alarms triggered

```
(8) Curious (Temp D), 8, Mon 9:00, 10mins, Mon 11:00, 10mins
(8.0) Curious OrderBook, 8, Mon 11:00, 20mins, Mon 12:00, 50mins
```

Alarms mitigated

Alarms not considered

```
(0) Recharge, 10, Mon 13:00, 30mins, Mon 15:00
```

```
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Prepare order 12, 8, Mon 16:30, 20mins, Mon 18:00
(0) Curious (Temp A), 8, Mon 11:30, 10mins, Mon 13:30
(0) Curious (Temp B), 8, Mon 11:30, 10mins, Mon 13:30
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(2.86) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
(2.35) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.5) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(4.56) Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
(0) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(1.96) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(0.83) Restock Coffee, 7, Mon, 30mins, Wed
(6.88) Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30, 20mins
```

C.10 Time = Mon 12:00, τ=6.5

By midday the agent has succeeded in preparing order 4 in time, and this goal is deleted. The agent is reminded at this time to check the temperature of room C. This goal is activated to join the goals to have checked the temperature in room D and the order book.

Alarms triggered

```
(7.55) Curious (Temp C), 8, Mon 9:30, 10mins, Mon 11:30, 10mins
```

Alarms mitigated

Alarms not considered

```
(0) Recharge, 10, Mon 13:00, 30mins, Mon 15:00
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Prepare order 12, 8, Mon 16:30, 20mins, Mon 18:00
(2.18) Curious (Temp A), 8, Mon 11:30, 10mins, Mon 13:30
(2.18) Curious (Temp B), 8, Mon 11:30, 10mins, Mon 13:30
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(4.29) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
(3.53) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
```

```
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.65) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(4.71) Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
(0.0) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(2.36) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(1.00) Restock Coffee, 7, Mon, 30mins, Wed
```

C.11 Time = Mon 13:00, τ=5.5

At 1pm the threshold has dropped to 5.5 due to the agent having achieved its recently activated goals. Four alarms trigger; those encapsulating the goals to have checked the temperatures in rooms A and B, and to have prepared orders 5 and 6. The agent activates the two more short term goals, but mitigates both goals that were generated in the service of the motive to satisfy customer orders.

Goals to have checked the temperatures in rooms C and D and to have checked to the order book have been achieved, and new goals are replenished. Furthermore, the agent generates three goals through decisions to satisfy orders 13, 14 and 15 requested by customers. The agent is now influenced by two active and thirty two inactive goals.

Alarms triggered

```
(6.55) Curious (Temp A), 8, Mon 11:30, 10mins, Mon 13:30
(6.55) Curious (Temp B), 8, Mon 11:30, 10mins, Mon 13:30
(5.71) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
```

Alarms mitigated

```
(0.0) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
```

Alarms not considered

```
(0) Curious (Temp D), 8, Mon 17:00, 10mins, Mon 19:00
(0) Curious (Temp C), 8, Mon 17:00, 10mins, Mon 19:00
(0) Curious OrderBook, 8, Mon 14:00, 20mins, Mon 15:00
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0.0) Recharge, 10, Mon 13:00, 30mins, Mon 15:00
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Prepare order 12, 8, Mon 16:30, 20mins, Mon 18:00
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(4.71) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
```

```
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.79) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(4.85) Tidy E, 5, Sun 15:00, 2.0hrs, Tue 3:00
(0.15) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(2.76) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(1.17) Restock Coffee, 7, Mon, 30mins, Wed
```

C.12 Time = Mon 14:00, τ=5.5

At 2pm, the agent considers recharging itself, and preparing order 6. The decision is made to mitigate the alarm encapsulating the goal to have prepared order 6, and thus consider it again at a time nearer to its deadline. Goals to have checked the temperature in rooms A and B are replenished.

While achieving the goal to have checked the temperature in room A, the agent detected a fault in the freezer system of this room. This puts the commodities stored in room A at risk. In response to this problem, a goal is generated to have saved the stock that is currently stored in room A (e.g. by moving it to room B, which has similar storage conditions). This goal is generated through decision and encapsulated in an alarm in the usual way, but this goal has an immediate temporal context and high importance.

The intensity of the alarm encapsulating the goal to have tidied room E reaches maximum at this time. This maximum (5) is low relative to the other alarms influencing the agent's behaviour at this time. Despite this alarm having reached maximum intensity, the threshold is sufficiently high to prevent it from triggering. The goal to have tidied room E will not be considered until the threshold drops to 5. In this simulation, the threshold drops to 5 at 8pm (see section C.19).

Alarms triggered

```
(6.67) Recharge, 10, Mon 13:00, 30mins, Mon 15:00
(5.88) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
```

Alarms mitigated

```
(0.0) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
```

Alarms not considered

```
(0) Curious (Temp A), 8, Mon 16:00, 10mins, Mon 18:00
(0) Curious (Temp B), 8, Mon 16:00, 10mins, Mon 18:00
(0) SaveStock A, 12, Mon 14:00, 30mins, Mon 14:40
(0) Curious (Temp D), 8, Mon 17:00, 10mins, Mon 19:00
(0) Curious (Temp C), 8, Mon 17:00, 10mins, Mon 19:00
(0.0) Curious OrderBook, 8, Mon 14:00, 20mins, Mon 15:00
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
```

```
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Prepare order 12, 8, Mon 16:30, 20mins, Mon 18:00
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.94) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(5) Tidy E, 5, Sun 4:00, 2.0hrs, Mon 16:00
(0.29) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0.0) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(3.16) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(1.34) Restock Coffee, 7, Mon, 30mins, Wed
(2.86) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
```

C.13 Time = Mon 14:05, τ=5.5

A short time later, the goal to have saved the stock in room A is considered and activated. At this time the agent is in the office recharging its batteries. This constitutes an opportunity to to check the order book (due to the agent being in the office).

Alarms triggered

```
(6.0) SaveStock A, 12, Mon 14:00, 30mins, Mon 14:40
(8) Curious OrderBook, 8, Mon 14:00, 20mins, Mon 15:00, Sample OrderBook
```

Alarms mitigated

Alarms not considered

```
(0) Curious (Temp A), 8, Mon 16:00, 10mins, Mon 18:00
(0) Curious (Temp B), 8, Mon 16:00, 10mins, Mon 18:00
(0) Curious (Temp D), 8, Mon 17:00, 10mins, Mon 19:00
(0) Curious (Temp C), 8, Mon 17:00, 10mins, Mon 19:00
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Prepare order 12, 8, Mon 16:30, 20mins, Mon 18:00
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
```

```
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.95) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(5) Tidy E, 5, Sun 4:00, 2.0hrs, Mon 16:00
(0.31) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0.31) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(3.19) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(1.35) Restock Coffee, 7, Mon, 30mins, Wed
(3.10) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
(0.22) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
```

C.14 Time = Mon 15:00, τ=5.5

The primary influence on the agent's action between 2.05pm and 3pm was the satisfaction of the goal to have saved the stock in room A. This goal has now been achieved, and the agent can plan to act on its other active goals. The agent is again reminded of the goal to have prepared order 5 by 5pm. This goal is activated.

Alarms triggered

```
(5.71) Prepare order 5, 10, Mon 9:00, 1.0hrs, Mon 17:00
```

Alarms mitigated

Alarms not considered

```
(0) Curious (Temp A), 8, Mon 16:00, 10mins, Mon 18:00
(0) Curious (Temp B), 8, Mon 16:00, 10mins, Mon 18:00
(0) Curious (Temp D), 8, Mon 17:00, 10mins, Mon 19:00
(0) Curious (Temp C), 8, Mon 17:00, 10mins, Mon 19:00
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Prepare order 12, 8, Mon 16:30, 20mins, Mon 18:00
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(3.09) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(5) Tidy E, 5, Sun 4:00, 2.0hrs, Mon 16:00
(0.44) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0.15) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(3.56) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(1.50) Restock Coffee, 7, Mon, 30mins, Wed
(2.65) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
```

C.15 Time = Mon 16:00, τ=6

None of the agent's alarms have a sufficiently high intensity for them to trigger at this time. The threshold is around 0.7 above the alarm of greatest intensity at this time: i.e. the alarm encapsulating the goal to have prepared order 6.

Alarms triggered

Alarms mitigated

Alarms not considered

```
(0) Curious OrderBook, 8, Mon 17:00, 20mins, Mon 18:00
(0) Recharge, 10, Mon 18:00, 30mins, Mon 20:00
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0.0) Curious (Temp A), 8, Mon 16:00, 10mins, Mon 18:00
(0.0) Curious (Temp B), 8, Mon 16:00, 10mins, Mon 18:00
(0) Curious (Temp D), 8, Mon 17:00, 10mins, Mon 19:00
(0) Curious (Temp C), 8, Mon 17:00, 10mins, Mon 19:00
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Prepare order 12, 8, Mon 16:30, 20mins, Mon 18:00
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(3.24) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(5) Tidy E, 5, Sun 4:00, 2.0hrs, Mon 16:00
(0.59) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0.29) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(3.96) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(1.67) Restock Coffee, 7, Mon, 30mins, Wed
(5.29) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
```

C.16 Time = Mon 17:00, τ=6

Between 4pm and 5pm the agent's action has been primarily directed towards preparing order 5. At this time, the agent is reminded of its next order of the day (order 6, due to be collected at 6pm). The agent estimates that it will take about half an hour to prepare this order, and so it is reminded of the goal in plenty of time to achieve it before 6pm.

Alarms triggered

```
(7.94) Prepare order 6, 10, Mon 9:00, 30mins, Mon 18:00
```

Alarms mitigated

Alarms not considered

```
(0.0) Curious OrderBook, 8, Mon 17:00, 20mins, Mon 18:00
(0) Recharge, 10, Mon 18:00, 30mins, Mon 20:00
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(4.36) Curious (Temp A), 8, Mon 16:00, 10mins, Mon 18:00
(4.36) Curious (Temp B), 8, Mon 16:00, 10mins, Mon 18:00
(0.0) Curious (Temp D), 8, Mon 17:00, 10mins, Mon 19:00
(0.0) Curious (Temp C), 8, Mon 17:00, 10mins, Mon 19:00
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(3.43) Prepare order 12, 8, Mon 16:30, 20mins, Mon 18:00
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0.0) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(3.38) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(5) Tidy E, 5, Sun 4:00, 2.0hrs, Mon 16:00
(0.74) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0.44) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(4.36) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(1.84) Restock Coffee, 7, Mon, 30mins, Wed
```

C.17 Time = Mon 18:00, τ=6.5

At 6pm the goals to have checked the temperature of rooms A and B are considered, and this activated along with the goals to have mitigated curiosity about the order book and to have prepared order 12. Note that the deadline for the goals to have prepared order 12 is 6pm, but if the alarm was sampled at 5.30pm, the intensity would be 6.86, sufficient for the alarm to trigger at this time if the threshold is 6.5. The agent will therefore be reminded of the goal in time for it to be achieved before the deadline.

Alarms triggered

```
(8) Curious OrderBook, 8, Mon 17:00, 20mins, Mon 18:00
(8) Curious (Temp A), 8, Mon 16:00, 10mins, Mon 18:00
(8) Curious (Temp B), 8, Mon 16:00, 10mins, Mon 18:00
(8) Prepare order 12, 8, Mon 16:30, 20mins, Mon 18:00
```

Alarms mitigated

Alarms not considered

```
(0.0) Recharge, 10, Mon 18:00, 30mins, Mon 20:00
```

```
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(4.36) Curious (Temp D), 8, Mon 17:00, 10mins, Mon 19:00
(4.36) Curious (Temp C), 8, Mon 17:00, 10mins, Mon 19:00
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(2.67) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0.0) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0.15) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(3.53) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(5) Tidy E, 5, Sun 4:00, 2.0hrs, Mon 16:00
(0.88) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0.59) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(4.75) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(2.01) Restock Coffee, 7, Mon, 30mins, Wed
```

C.18 Time = Mon 19:00, τ=5.5

At 7pm, the agent is reminded of the final order of the day (this alarm will actually trigger sometime between 6pm and 7pm), order 10. This goal is activated and redirects the agent's attention towards it. The agent also activates the goals to have checked the temperature in rooms C and D, but mitigates the alarm encapsulating the goal to have recharged. The goals to have checked the order book, and the temperature of rooms A and B are replenished. At this time there are no new orders on the order book.

Alarms triggered

```
(6.67) Recharge, 10, Mon 18:00, 30mins, Mon 20:00
(8) Curious (Temp D), 8, Mon 17:00, 10mins, Mon 19:00
(8) Curious (Temp C), 8, Mon 17:00, 10mins, Mon 19:00
(8) Prepare order 10, 8, Mon 17:30, 20mins, Mon 19:20
```

Alarms mitigated

```
(0.0) Recharge, 10, Mon 18:00, 30mins, Mon 20:00
```

Alarms not considered

```
(0) Curious OrderBook, 8, Mon 20:00, 20mins, Mon 21:00
(0) Curious (Temp A), 8, Mon 21:00, 10mins, Mon 23:00
(0) Curious (Temp B), 8, Mon 21:00, 10mins, Mon 23:00
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
```

```
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0.15) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0.29) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(3.68) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(5) Tidy E, 5, Sun 4:00, 2.0hrs, Mon 16:00
(1.03) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0.74) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(5.15) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(2.18) Restock Coffee, 7, Mon, 30mins, Wed
```

C.19 Time = Mon 20:00, τ=5

From this time onwards the agent has no orders to prepare, so there is little for it to do except for monitoring important variables in the domain (e.g. the temperature of room A), tidying the warehouse etc. At this time, the agent redirects attention to recharging itself, tidying room E and restocking the warehouse with salsa. Note that the intensity of the alarm encapsulating the goal to have tidied room E has been maximal since 2pm, but this is the first time that it has equalled or exceeded the threshold. The behaviour of the alarm encapsulating this goal is from 6am to the present time is illustrated in figure 7.2.

Alarms triggered

```
(5) Tidy E, 5, Sun 4:00, 2.0hrs, Mon 16:00
(5.55) Restock Salsa, 7, Mon 0:00, 30mins, Tue 0:00
(10) Recharge, 10, Mon 18:00, 30mins, Mon 20:00, 30mins
```

Alarms mitigated

Alarms not considered

```
(0.0) Curious OrderBook, 8, Mon 20:00, 20mins, Mon 21:00
(0) Curious (Temp A), 8, Mon 21:00, 10mins, Mon 23:00
(0) Curious (Temp B), 8, Mon 21:00, 10mins, Mon 23:00
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
```

```
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0.29) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0.44) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(3.82) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(1.18) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(0.88) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
(2.35) Restock Coffee, 7, Mon, 30mins, Wed
```

C.20 Time = Mon 21:00, τ=4

At this time, the agent has a plan to tidy room E in the next couple of hours. This plan (prior time commitment) may conflict with the timely achievement of the goals to have mitigated curiosity about the temperatures in rooms A and B. These alarms are both shifted by 1.83 hours, and hence are at maximal intensity when evaluated at this time. The agent is at present placing an order for salsa with some supplier; this constitutes an opportunity to satisfy the goal to have restocked the warehouse with coffee.

Alarms triggered

```
(8) Curious OrderBook, 8, Mon 20:00, 20mins, Mon 21:00
(8) Curious (Temp A), 8, Mon 21:00, 10mins, Mon 23:00, 1.83hrs
(8) Curious (Temp B), 8, Mon 21:00, 10mins, Mon 23:00, 1.83hrs
(7) Restock Coffee, 7, Mon, 30mins, Wed, 0mins, PlaceOrder Coffee
```

Alarms mitigated

```
(0.0) Curious (Temp A), 8, Mon 21:00, 10mins, Mon 23:00, 1.83hrs
(0.0) Curious (Temp B), 8, Mon 21:00, 10mins, Mon 23:00, 1.83hrs
```

Alarms not considered

```
(0) Curious (Temp C), 8, Tue 1:00, 10mins, Tue 3:00
(0) Curious (Temp D), 8, Tue 1:00, 10mins, Tue 3:00
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0.0) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0.44) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0.59) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(3.97) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(1.32) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(1.03) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

C.21 Time = Mon 22:00, τ=4

At 10pm the threshold has dropped to 4 reflecting the agent's lower activity at this time. Alarms tend to trigger earlier. For example the goals to have checked the temperature of rooms A and B are both activated at an intensity of 5.82. The agent also activates the goal to have tidied room D at an intensity of 4.12, and to have checked the order book at maximal intensity. The intensity of the alarm encapsulating the goal to have tidied room D has been gradually increasing for a while, but prior to the present time, the threshold has been too high for it to trigger (if the threshold was the same at 6pm as it is now this alarm would have triggered at that time).

Alarms triggered

```
(8) Curious OrderBook, 8, Mon 23:00, 20mins, Tue 0:00, 2.0hrs
(4.12) Tidy D, 5, Sun 18:00, 2.0hrs, Tue 6:00
(5.82) Curious (Temp A), 8, Mon 21:00, 10mins, Mon 23:00, 1.0hrs
(5.82) Curious (Temp B), 8, Mon 21:00, 10mins, Mon 23:00, 1.0hrs
```

Alarms mitigated

Alarms not considered

```
(0) Recharge, 10, Tue 0:00, 30mins, Tue 2:00
(0) Restock Salsa, 7, Wed 0:00, 30mins, Thu 0:00
(0) Restock Coffee, 7, Mon wk2, 30mins, Wed wk2
(0) Curious (Temp C), 8, Tue 1:00, 10mins, Tue 3:00
(0) Curious (Temp D), 8, Tue 1:00, 10mins, Tue 3:00
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0.15) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0.59) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0.74) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(1.47) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(1.18) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

C.22 Time = Mon 23:00, τ=3.5

Alarms triggered

Alarms mitigated

Alarms not considered

```
(0) Curious (Temp A), 8, Tue 1:00, 10mins, Tue 3:00
(0) Curious (Temp B), 8, Tue 1:00, 10mins, Tue 3:00
```

```
(0) Recharge, 10, Tue 0:00, 30mins, Tue 2:00
(0) Restock Salsa, 7, Wed 0:00, 30mins, Thu 0:00
(0) Restock Coffee, 7, Mon wk2, 30mins, Wed wk2
(0) Curious (Temp C), 8, Tue 1:00, 10mins, Tue 3:00
(0) Curious (Temp D), 8, Tue 1:00, 10mins, Tue 3:00
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0.29) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0.74) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(0.88) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(1.62) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(1.32) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

C.23 Time = Tue 0:00, τ=3

At this point, the agent has checked the order book, found that there are no new orders, and deleted the goal. This joins the alarms encapsulating the goals to have checked the temperature in rooms A and B, which were replenished at 11pm.

Alarms triggered

Alarms mitigated

Alarms not considered

```
(0) Curious OrderBook, 8, Tue 1:00, 20mins, Tue 2:00
(0) Curious (Temp A), 8, Tue 1:00, 10mins, Tue 3:00
(0) Curious (Temp B), 8, Tue 1:00, 10mins, Tue 3:00
(0.0) Recharge, 10, Tue 0:00, 30mins, Tue 2:00
(0) Restock Salsa, 7, Wed 0:00, 30mins, Thu 0:00
(0) Restock Coffee, 7, Mon wk2, 30mins, Wed wk2
(0) Curious (Temp C), 8, Tue 1:00, 10mins, Tue 3:00
(0) Curious (Temp D), 8, Tue 1:00, 10mins, Tue 3:00
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0.44) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0.0) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0.0) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0.0) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
```

```
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0.0) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(0.88) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(1.03) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(1.76) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(1.47) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

C.24 Time = Tue 1:00, τ=3

At 1am on Tuesday the agent considers activating the goal to have checked the order book, and the goal to have recharged. Both are activated. At this time, the agent has completed its tidying of room E, and so a new goal is generated by a periodic replenishment process in response to the deletion of this completed goal, the characteristics of which are illustrated on page 172. The period of replenishment is two days, and so the deadline is set to Thursday at 1am. The travel time does not change; this is set to two hours by the replenishment rule. The delay time is set to 1pm on Tuesday (12 hours from now).

Alarms triggered

```
(8.0) Curious OrderBook, 8, Tue 1:00, 20mins, Tue 2:00, 2.0hrs
(10) Recharge, 10, Tue 0:00, 30mins, Tue 2:00, 2.0hrs
```

Alarms mitigated

Alarms not considered

```
(0) Tidy E, 5, Tue 13:00, 2.0hrs, Thu 1:00
(0.0) Curious (Temp A), 8, Tue 1:00, 10mins, Tue 3:00
(0.0) Curious (Temp B), 8, Tue 1:00, 10mins, Tue 3:00
(0) Restock Salsa, 7, Wed 0:00, 30mins, Thu 0:00
(0) Restock Coffee, 7, Mon wk2, 30mins, Wed wk2
(0.0) Curious (Temp C), 8, Tue 1:00, 10mins, Tue 3:00
(0.0) Curious (Temp D), 8, Tue 1:00, 10mins, Tue 3:00
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0.59) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0.15) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0.15) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0.15) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0.15) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(1.03) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(1.18) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(1.91) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(1.62) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

C.25 Time = Tue 2:00, τ=3

At 2am four alarms trigger. The agent considers checking the temperature in rooms A, B, C and D. The agent has also finished tidying room D and mitigated its curiosity about the state of the order book, and so two new goals are replenished and encapsulated in appropriate alarms. (Note, no new orders have arrived.)

Alarms triggered

```
(4.36) Curious (Temp A), 8, Tue 1:00, 10mins, Tue 3:00
(4.36) Curious (Temp B), 8, Tue 1:00, 10mins, Tue 3:00
(4.36) Curious (Temp C), 8, Tue 1:00, 10mins, Tue 3:00
(4.36) Curious (Temp D), 8, Tue 1:00, 10mins, Tue 3:00
```

Alarms mitigated

Alarms not considered

```
(0) Tidy D, 5, Tue 14:00, 2.0hrs, Thu 2:00
(0) Curious OrderBook, 8, Tue 3:00, 20mins, Tue 4:00
(0) Tidy E, 5, Tue 13:00, 2.0hrs, Thu 1:00, 0mins
(0) Restock Salsa, 7, Wed 0:00, 30mins, Thu 0:00
(0) Restock Coffee, 7, Mon wk2, 30mins, Wed wk2
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0.74) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0.29) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0.29) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0.29) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0.29) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(1.18) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(1.32) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.06) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(1.76) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

C.26 Time = Tue 3:00, τ=3

Alarms triggered

Alarms mitigated

Alarms not considered

```
(0) Recharge, 10, Tue 5:00, 30mins, Tue 7:00
(0) Tidy D, 5, Tue 14:00, 2.0hrs, Thu 2:00
(0.0) Curious OrderBook, 8, Tue 3:00, 20mins, Tue 4:00
(0) Tidy E, 5, Tue 13:00, 2.0hrs, Thu 1:00
(0) Restock Salsa, 7, Wed 0:00, 30mins, Thu 0:00
(0) Restock Coffee, 7, Mon wk2, 30mins, Wed wk2
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
```

```
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(0.88) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0.44) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0.44) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0.44) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0.44) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(1.32) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(1.47) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.21) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(1.91) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0.0) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

C.27 Time = Tue 4:00, τ=2.5

Alarms triggered

```
(8) Curious OrderBook, 8, Tue 3:00, 20mins, Tue 4:00, 0mins
```

Alarms mitigated

Alarms not considered

```
(0) Curious (Temp A), 8, Tue 6:00, 10mins, Tue 8:00
(0) Curious (Temp B), 8, Tue 6:00, 10mins, Tue 8:00
(0) Curious (Temp C), 8, Tue 8:00, 10mins, Tue 10:00
(0) Curious (Temp D), 8, Tue 8:00, 10mins, Tue 10:00
(0) Recharge, 10, Tue 5:00, 30mins, Tue 7:00, 0mins
(0) Tidy D, 5, Tue 14:00, 2.0hrs, Thu 2:00, 0mins
(0) Tidy E, 5, Tue 13:00, 2.0hrs, Thu 1:00, 0mins
(0) Restock Salsa, 7, Wed 0:00, 30mins, Thu 0:00
(0) Restock Coffee, 7, Mon wk2, 30mins, Wed wk2
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(1.03) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0.59) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0.59) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0.59) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0.59) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(1.47) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(1.62) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.35) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
(2.06) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(0.91) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

C.28 Time = Tue 5:00, τ=2.5

During the last few hours the agent's action has been directed towards very little; mostly monitoring a number of important aspects of the warehouse domain such as the temperatures of frozen and refrigerated rooms and the order book and tidying rooms D and E. These goals have been recently achieved and new goals of the same type replenished. Those generated in the service of the motive of curiosity were replenished at 4pm, and those generated in the service of the motive of tidyness, at 1am and 2am. At 4pm three new orders were also recorded and alarms set for these new goals. At 5pm the agent is reminded of its goal to have tidied room F. The agent is now influenced by a single active goals (the goal to have tidied room F) and 33 inactive goals.

Alarms triggered

```
(2.5) Tidy F, 5, Mon 12:00, 2.0hrs, Wed 0:00
```

Alarms mitigated

Alarms not considered

```
(0) Curious OrderBook, 8, Tue 6:00, 20mins, Tue 7:00, 0mins
(0) Prepare order 17, 10, Thu 16:30, 20mins, Thu 19:20, 0mins
(0) Prepare order 18, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2, 0mins
(0) Prepare order 19, 10, Tue 14:40, 1.0hrs, Tue 17:40, 0mins
(0) Curious (Temp A), 8, Tue 6:00, 10mins, Tue 8:00
(0) Curious (Temp B), 8, Tue 6:00, 10mins, Tue 8:00
(0) Curious (Temp C), 8, Tue 8:00, 10mins, Tue 10:00
(0) Curious (Temp D), 8, Tue 8:00, 10mins, Tue 10:00
(0.0) Recharge, 10, Tue 5:00, 30mins, Tue 7:00, 0mins
(0) Tidy D, 5, Tue 14:00, 2.0hrs, Thu 2:00, 0mins
(0) Tidy E, 5, Tue 13:00, 2.0hrs, Thu 1:00, 0mins
(0) Restock Salsa, 7, Wed 0:00, 30mins, Thu 0:00
(0) Restock Coffee, 7, Mon wk2, 30mins, Wed wk2
(0) Prepare order 16, 10, Wed 6:00 wk2, 30mins, Wed 15:00 wk2
(0) Prepare order 13, 10, Thu 16:30, 20mins, Thu 19:20
(0) Prepare order 14, 10, Mon 9:00 wk2, 1.0hrs, Mon 12:00 wk2
(0) Prepare order 15, 10, Tue 14:40, 1.0hrs, Tue 17:40
(0) Prepare order 2, 10, Wed 22:00, 1.0hrs, Thu 10:00
(1.18) Tidy C, 5, Mon 21:00, 2.0hrs, Wed 9:00
(0.74) Restock FrozenPizza, 7, Tue 0:00, 30mins, Thu 0:00
(0.74) Restock IceCream, 7, Tue 0:00, 30mins, Thu 0:00
(0.74) Restock TortillaChips, 7, Tue 0:00, 30mins, Thu 0:00
(0) Restock Yoghurt, 7, Wed 0:00, 30mins, Fri 0:00
(0) Prepare order 11, 10, Wed 9:00, 1.0hrs, Wed 12:00
(0) Prepare order 7, 10, Tue 9:00, 2.0hrs, Tue 17:00
(0) Prepare order 8, 10, Fri 9:00, 1.0hrs, Fri 14:00
(0) Prepare order 9, 10, Mon 9:00 wk2, 1.0hrs, Mon 16:00 wk2
(0.74) Restock CatFood, 7, Tue 0:00, 30mins, Thu 0:00
(1.62) Tidy A, 5, Mon 18:00, 2.0hrs, Wed 6:00
(1.76) Tidy B, 5, Mon 17:00, 2.0hrs, Wed 5:00
(2.21) Tidy LoadingBay, 5, Mon 14:00, 2.0hrs, Wed 2:00
(1.82) Prepare order 1, 10, Tue 3:00, 1.0hrs, Tue 15:00
(0.0) Prepare order 3, 10, Tue 5:00, 1.0hrs, Tue 17:00
```

Bibliography

Agre, P. & Chapman, D. (1987), Pengi: An implementation of a theory of activity, in *Proceedings of the Sixth National Conference on Artificial Intelligence*, pp. 268–272.

Agre, P. & Chapman, D. (1990), What are plans for?, *Robotics and Autonomous Systems* **6**, 17–34.

Allen, J. & Koomen, J. (1983), Planning using a temporal world model, in *Proceedings of the Eigth International Joint Conference on Artificial Intelligence*, pp. 741–747.

Anderson, J. (1993), *Rules of the mind*, Lawrence Erlbaum Associates.

Aylett, R., Fish, A. & Bartrum, S. (1991), Task planning in an uncertain world, in *International Conference on Control*, Vol. 2, pp. 801–806.

Bates, J., Loyall, A. & Reilly, W. (1992), Integrating reactivity, goals, and emotion in a broad agent, in *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*.

Beaudoin, L. (1994), Goal processing in autonomous agents, PhD thesis, School of Computer Science, University of Birmingham.

Beaudoin, L. & Sloman, A. (1993), A study of motive processing and attention, in *Prospects for Artificial Intelligence: Proceedings of the Ninth Biennial Conference on Artificial Intelligence and Cognitive Science*, pp. 229–238.

Bird, R. & Walder, P. (1988), *Introduction to functional programming*, Prentice-Hall International.

Birnbaum, L. (1986), Integrated processing in planning and understanding, PhD thesis, Department of Computer Science, Yale University, YALEU/CSD/RR#489.

Birnbaum, L. & Collins, G. (1984), Opportunistic planning and Freudian slips, in *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, pp. 124–127.

Boddy, M. & Dean, T. (1989), Solving time-dependent planning problems, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 979–984.

Boddy, M. & Dean, T. (1994), Deliberation scheduling for problem solving in time-constrained environments, *Artificial Intelligence* **67**, 245–285.

Boden, M. (1994), Artificiality and autonomy, *AISB Quarterly* **Spring**, 22–27.

Brandimonte, M., Einstein, G. O. & McDaniel, M. A. (eds.) (1996), *Prospective memory: Theory and applications*, Lawrence Erlbaum Associates, New Jersey.

Bratman, M. (1992), Planning and the stability of intention, *Minds and Machines* **2**(1), 1–16.

Bratman, M., Israel, D. & Pollack, M. (1988), Plans and resource-bounded practical reasoning, *Computational Intelligence* **4**(4), 349–355.

Brooks, R. (1986), A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation* **2**, 14–23.

Brooks, R. (1991), Elephants don't play chess, in P. Maes (ed.), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, Bradford-MIT Press.

Bylander, T. (1994), The computational complexity of propositional STRIPS planning, *Artificial Intelligence* **69**, 165–204.

Carbonell, J. (1982), Where do goals come from?, in *Proceedings of the Fourth Annual Conference of the Cognitive Science Society*, pp. 191–194.

Castelfranchi, C. (1995), Guarantees for autonomy in cognitive agent architecture, in M. Wooldridge & N. Jennings (eds.), *Intelligent Agents: Proceedings of the ECAI-94 workshop on Agent Theories, Architectures, and Languages*, Vol. 890 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 56–70.

Chapman, D. (1987), Planning for conjunctive goals, *Artificial Intelligence* **32**, 333–377.

Chapman, D. (1991), *Vision, instruction, and action*, MIT Press.

Cherniak, C. (1986), *Minimal rationality*, MIT Press/Bradford.

Cohen, P. & Levesque, H. (1990), Intention is choice with commitment, *Artificial Intelligence* **42**, 213–61.

Cohen, P., Greenberg, M., Hart, D. & Howe, A. (1989), Trial by fire: Understanding the design requirements for agents in complex environments, *AI Magazine*.

Collins, G. & Pryor, L. (1995), Planning under uncertainty: Some key issues, in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1567–1573.

Cooper, R., Shallice, T. & Farringdon, J. (1995), Symbolic and continuous processes in the automatic selection of actions, *AISB Quarterly* **Spring**, 13–24. Presented at the Tenth Biennial Conference on Artificial Intelligence and Cognitive Science.

Dean, T. & Wellman, M. (1991), *Planning and control*, Morgan Kaufmann.

Dean, T., Firby, R. & Miller, D. (1988), Hierarchical planning involving deadlines, travel time, and resources, *Computing Intelligence* **4**, 381–98.

Dennett, D. (1987), *The intentional stance*, MIT Press.

Doukidis, G. & Angelides, M. (1994), A framework for integrating artificial intelligence and simulation, *Artificial Intelligence Review* **8**(1), 55–85.

Ellis, J. (1994), Prospective memory or the realisation of delayed intentions: A conceptual framework, in M. Brandimonte, G. Einstein & M. McDaniel (eds.), *Prospective memory: Theory and applications*, Lawrence Erlbaum Associates.

Ellis, J. & Nimmo-Smith, I. (1993), Recollecting naturally-occuring intentions: A study of cognitive and affective factors, *Memory* **1**(2), 107–126.

Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N. & Williamson, M. (1992), An approach to planning with incomplete information, in *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pp. 115–125.

Fikes, R. & Nilsson, N. (1971), STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence* **2**, 189–208.

Firby, R. (1987), An investigation into reactive planning in complex domains, in *Proceedings of the Sixth National Conference on Artificial Intelligence*, pp. 202–206.

Fischer, K., Müller, J. & Pischel, M. (1996), A pragmatic BDI architecture, in M. Wooldridge, J. Müller & M. Tambe (eds.), *Intelligent Agents II: Proceedings of the IJCAI-95 workshop on Agent Theories, Architectures, and Languages*, Vol. 1037 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 203–218.

Fox, M. (1993). A formal basis for hierarchical planning, in *Proceedings of the Sixth Australian Joint Conference on Artificial Intelligence*.

Fox, M. & Long, D. (1995). Hierarchical planning using abstraction, *IEE Proceedings: Control Theory and Applications* **142**(3), 197–210.

Georgeff, M. & Ingrand, F. (1989). Decision-making in an embedded reasoning system, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 972–978.

Georgeff, M. & Lansky, A. (1987). Reactive reasoning and planning, in *Proceedings of the Sixth National Conference on Artificial Intelligence*, pp. 677–681.

Ginsberg, M. (1989). Universal planning: An (almost) universally bad idea, *AI Magazine* **10**(4), 40–44.

Grune, D. (1987). How to compare the incomparable, *Information processing letters* **24**, 177–181.

Haddadi, A. & Sundermeyer, K. (1996). Belief-desire-intention agent architectures, in G. O'Hare & N. R. Jennings (eds.), *Foundations of Distributed Artificial Intelligence*, Wiley-Interscience, pp. 169–186.

Haddawy, P. & Hanks, S. (1992). Representations for decision-theoretic planning: Utility functions for deadline goals, in *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pp. 71–82.

Hammond, K. (1989a). *Case-based planning: Viewing planning as a memory task*, Academic Press.

Hammond, K. (1989b). Opportunistic memory, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 504–510.

Hammond, K., Converse, T. & Grass, J. (1995), The stabilisation of environments, *Artificial Intelligence* **72**, 305–327.

Harris, J. & Wilkins, A. (1982). Remembering to do things: A theoretical framework and an illustrative experiment, *Human Learning* **1**, 123–136.

Hayes-Roth, B. (1985). A blackboard architecture for control, *Artificial Intelligence* **26**(3), 251–321.

Hayes-Roth, B. (1995). An architecture for adaptive intelligent systems. *Artificial Intelligence* **72**, 329–365.

Hayes-Roth, B. & Hayes-Roth, F. (1979). A cognitive model of planning. *Cognitive Science* **3**, 275–310.

Hayes-Roth, B., Washington, R., Hewett, R., Hewett, M. & Seiver, A. (1989). Intelligent monitoring and control. in *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pp. 243–249.

Jennings, N. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* **75**, 195–240.

Kaelbling, L. & Rosenschein, S. (1990). Action and planning in embedded agents. *Robotics and Autonomous Systems* **6**(1 and 2), 35–48.

Kolodner, J., Simpson, R. & Sycara-Cyranski, L. (1985). A process model of case-based reasoning in problem solving. in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*.

Kvavilashvili, L. (1987). Remembering intention as a distinct form of memory. *British Journal of Psychology* **78**, 123–136.

Laffey, T., Cox, P., Schmidt, J., Kao, S. & Read, J. (1988). Real-time knowledge-based systems. *AI Magazine* **9**(1), 27–45.

Lee, H., Tannock, J. & Williams, J. (1993). Logic-based reasoning about actions and plans in artificial intelligence. *The Knowledge Engineering Review* **8**(2), 91–120.

Lesser, V., Pavlin, J. & Durfee, E. (1988). Approximate processing in real-time problem solving. *AI Magazine* **9**(1), 49–74.

Long, D. (1989). A review of temporal logics. *The Knowledge Engineering Review* **4**(2), 141–162.

Luck, M. & d'Inverno, M. (1995). A formal framework for agency and autonomy. in *Proceedings of the First International Conference on Multi-Agent Systems*, pp. 254–260.

Maes, P. (1989). The dynamics of action selection. in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 991–997.

Maes, P. (1991). Situated agents can have goals. in P. Maes (ed.), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. Bradford-MIT Press.

McAllester, D. & Rosenblitt, D. (1991). Systematic non-linear planning. in *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 634–639.

McDermott, D. (1982). A temporal logic for reasoning about processes and plans. *Cognitive Science* **6**, 101–155.

Müller, H. J. (1996). Negotiation principles. in G. M. P. O'Hare & N. R. Jennings (eds.), *Foundations of Distributed Artificial Intelligence*, Wiley.

Musliner, D. J. (1994), Using abstraction and nondeterminism to plan reaction loops. in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 1036–1041.

Musliner, D. J., Hendler, J. A., Agrawala, A. K., Durfee, E. H., Strosnider, J. K. & Paul, C. J. (1995). The challenges of real-time AI. *IEEE Computer*.

Neisser, U. (1963). The imitation of man by machine. *Science* **139**, 193–197.

Newell, A. & Simon, H. A. (1972). *Human problem solving*, Prentice-Hall.

Nilsson, N. J. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research* **1**, 139–158.

Norman, D. & Shallice, T. (1986). Attention to action. in R. J. Davidson, G. E. Schwartz & D. Shapiro (eds.), *Consciousness and self-regulation: Advances in theory and research*, Plenum Press.

Norman, T. J. (1994). Motivated goal and action selection. in *Working Notes of the AISB Workshop on Models or Behaviours — Which Way Forward for Robotics?* University College London, Department of Computer Science Research Note 94/18.

Norman, T. J. & Long, D. P. (1995a), Alarms: Heuristics for the control of reasoning attention. in *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pp. 494–499.

Norman, T. J. & Long, D. P. (1995b). Goal Creation in Motivated Agents. in M. J. Wooldridge & N. R. Jennings (eds.), *Intelligent Agents: Proceedings of the 1994 workshop on Agent Theories, Architectures, and Languages (ATAL-94)*, Vol. 890 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 277–290.

Norman, T. J. & Long, D. P. (1996), Alarms: An implementation of motivated agency, in M. J. Wooldridge, J. P. Müller & M. Tambe (eds.), *Intelligent Agents II: Proceedings of the IJCAI-95 workshop on Agent Theories, Architectures, and Languages*, Vol. 1037 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 219–234.

Norman, T. J., Jennings, N. R., Faratin, P. & Mamdani, E. H. (1997), Designing and implementing a multi-agent architecture for business process management, in J. P. Müller, M. J. Wooldridge & N. R. Jennings (eds.), *Intelligent Agents III: Proceedings of the ECAI-96 Workshop on Agent Theories, Architectures, and Languages*, Vol. 1193 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 261–276.

Ortony, A., Clore, G. & Collins, A. (1988), *The cognitive structure of emotions*, Cambridge University Press.

Patalano, A., Seifert, C. & Hammond, K. (1993), Predictive encoding: Planning for opportunities, in *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pp. 800–805.

Penberthy, J. & Weld, D. (1992), UCPOP: A sound, complete, partial order planner for ADL, in *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pp. 103–114.

Polk, T. & Rosenbloom, P. (1994), Task-independent constraints on a unified theory of cognition, in F. Boller & J. Grafman (eds.), *Handbook of Neuropsychology*, Vol. 9, Elsevier.

Pollack, M. (1992), The uses of plans, *Artificial Intelligence* **57**, 43–68.

Pollack, M. & Ringuette, M. (1990), Introducing the Tileworld: Experimentally evaluating agent architectures, in *Proceedings of the Eighth National Conference on Artificial Intelligence*.

Pollack, M., Joslin, D., Nunes, A., Ur, S. & Ephrati, E. (1994), Experimental investigation of an agent commitment strategy, Technical Report 94–31, University of Pittsburgh, Department of Computer Science.

Pryor, L. & Collins, G. (1992a), Planning to perceive: A Utilitarian approach, in *Working Notes of the AAAI Spring Symposium on the Control of Selective Perception*.

Pryor, L. & Collins, G. (1992b), Reference features as guides to reasoning about opportunities, in *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pp. 230–235.

Rao, A. (1996), Decision procedures for propositional linear-time belief-desire-intention logics, in M. Wooldridge, J. Müller & M. Tambe (eds.), *Intelligent Agents II: Proceedings of the IJCAI-95 workshop on Agent Theories, Architectures, and Languages*, Vol. 1037 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 33–48.

Rao, A. & Georgeff, M. (1991), Modelling rational agents within a BDI-architecture, in *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pp. 473–484.

Rao, A. & Georgeff, M. (1992), An abstract architecture for rational agents, in *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*.

Rao, A. & Georgeff, M. (1995), BDI agents: From theory to practice, in *Proceedings of the First International Conference on Multi-Agent Systems*, pp. 312–319.

Reason, J. (1984), Lapses of attention, in W. Parasuraman, R. Davis & J. Beattie (eds.), *Varieties of attention*, Academic Press.

Rich, E. & Knight, K. (1991), *Artificial Intelligence*, McGraw Hill.

Rosenschein, J. & Zlotkin, G. (1994), *Rules of encounter: Designing conventions for negotiation among computers*, MIT Press.

Russell, S. & Wefald, E. (1991), *Do the right thing: Studies in limited rationality*, MIT Press.

Sacerdoti, E. (1975), The nonlinear nature of plans, in *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pp. 206–214.

Schank, R. & Abelson, R. (1977), *Scripts, Plans, Goals, and Understanding*, Laurence Erlbaum Associates.

Schoppers, M. (1987), Universal plans for reactive robots in unpredictable domains, in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 1039–1046.

Shoham, Y. (1993), Agent-oriented programming, *Artificial Intelligence* **60**, 51–92.

Simina, M. & Kolodner, J. (1995). Opportunistic reasoning: A design perspective, in *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pp. 78–83.

Simon, H. (1957). *Models of man: social and rational*, John Wiley.

Simon, H. (1967). Motivational and emotional controls of cognition. *Psychological Review* **74**, 29–39.

Simon, H. (1981). *The sciences of the artificial*, 2nd edition, MIT Press.

Slade, S. (1994). *Goal-based decision making: An interpersonal model*, Lawrence Erlbaum Associates.

Sloman, A. (1987). Motives, mechanisms, and emotions. *Cognition and Emotion* **1**(3), 217–233.

Sloman, A. (1992). Prolegomena to a theory of communication and effect, in A. Ortony, J. Slack & O. Stock (eds.), *Communication from an Artificial Intelligence Perspective: Theoretical and Applied Issues*, Springer-Verlag, pp. 229–260.

Sloman, A. (1994). Exploration in design space, in *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pp. 578–582.

Sloman, A. & Croucher, M. (1981). Why robots will have emotions, in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pp. 197–202.

Sloman, A. & Poli, R. (1996). SIM_AGENT: A toolkit for exploring agent designs, in M. Wooldridge, J. Müller & M. Tambe (eds.), *Intelligent Agents II: Proceedings of the IJCAI-95 workshop on Agent Theories, Architectures, and Languages*, Vol. 1037 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 392–407.

Smith, G. (1992). Strategies, scheduling effects, and the stability of intentions. *Minds and Machines* **2**(1), 17–26.

Strosnider, J. & Paul, C. (1994). A structured view of real-time problem solving. *AI Magazine* **15**(2), 45–66.

Tate, A. (1977). Generating project networks, in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp. 888–893.

Tenenberg, J. (1991). Abstraction in planning, in J. Allen, H. Kautz, R. Pelavin & J. Tenenberg (eds.), *Reasoning about plans*, Morgan Kaufman, chapter 4.

Vere, S. (1983). Planning in time: Windows and durations for activities and goals. *IEEE Transactions in Pattern Recognition and Machine Intelligence* **5**(3), 246–267.

Warrick, S. & Fox, M. (1994). Symbol acquisition in an adaptive agent architecture. in *Working Notes of the AISB Workshop on Models or Behaviours — Which Way Forward for Robotics?* University College London, Department of Computer Science Research Note: 94/13.

Weld, D. (1994). An introduction to least commitment planning. *AI Magazine* **15**(4), 27–61.

Wellman, M. & Doyle, J. (1991). Preferential semantics for goals. in *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 698–703.

Wilensky, R. (1983). *Planning and understanding: A computational approach to human reasoning*. Addison-Wesley.

Wooldridge, M. & Jennings, N. (1995a). Intellgent agents: Theory and practice. *Knowledge Engineering Review* **10**(2), 115–152.

Wooldridge, M. & Jennings, N. (eds.) (1995b). *Intelligent Agents: Proceedings of the ECAI-94 workshop on Agent Theories, Architectures, and Languages*. Vol. 890 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Wooldridge, M., Müller, J. & Tambe, M. (eds.) (1996). *Intelligent Agents II: Proceedings of the IJCAI-95 workshop on Agent Theories, Architectures, and Languages*. Vol. 1037 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Wright, I., Sloman, A. & Beaudoin, L. (to appear). Towards a design-based analysis of emotional episodes. *Philosophy, Psychiatry and Psychology*.