Knowledge with Real-Time Semantics

Annie. Z. Shamsudin Tharam. S. Dillon

Department of Computer Science and Computer Engineering
La Trobe University
Bundoora, Victoria-3083, Australia

Abstract

A real-time expert system has to reason with time. Although reasoning about the present is the most mandatory requirement, often it has also to reason with the future and/or the past. This involves the application of knowledge which deals with the time element of data which demonstrate time dependencies. In this paper, we present a method of representing the time dimension in the knowledge of a real-time expert system.

1 Introduction.

There are several important issues associated with the use of expert system techniques in a real-time environment. The most basic need is arguably a need to guarantee an acceptable response time. Also, a real-time expert system must be able to reason not only about the present, but also the past and the future. It must reason with data that is valid only during a given interval of interest. In addition, it must have a facility to reason with phenomena that has a certain ordering in time. Taking these issues into consideration, it appears there is a basic need to develop a method of representing the time dimension in the knowledge of a real-time expert system.

2 Knowledge representations with Real-Time Semantics.

We describe a method of incorporating real-time semantics in production rules, thus making them suitable for representing knowledge of a real-time expert system. It is intended that the knowledge representation should allow the incorporation of time dependent heuristics and dynamic models into the knowledge base.

2.1 Production rules with Real-Time Semantics.

Production rules consist of condition-action pairs and are written as:

if P1 then P2

where P1 is the condition(s) and P2 is an action(s). This means that if the condition(s) P1 has been determined to be true, then the action(s) defined by P2 must be carried out.

- Time Stamps for an action/condition. Under normal circumstances, there is usually a limit to the time during which a condition holds a particular value. This time limit can be specified by associating a time stamp Δt1 with the condition P1. We can also associate a time stamp, Δt2 say, with the action, to indicate the time limit during which the action holds true. Δt1 may well equal Δt2. These are represented as

if P1 | Δt1 then P2 | Δt2.

- Execution time. We define the time needed to execute an action as the Execution time. To execute an action involves firstly activating the action, and then completing it. We can represent the firing delay time as

if P1 | Δt1 then P2 | Δt2 || F2,

where F2 is the rule firing delay time.

We represent the action completion time as
if \( P_1 \) \( \Delta t_1 \) then \\
\( P_2 \) \( \Delta t_2 \) || [\( P_2 \) [\( C_2 \)].

where \( C_2 \) is the action completion time. In many real-time systems however, there is no need to differentiate between the firing delay time and the completion time. Hence we define it as

if \( P_1 \) \( \Delta t_1 \) then \( P_2 \) \( \Delta t_2 \) || \( E_2 \).

where it is expected that Execution time \( E_2 = F_2 + C_2 \). This simpler, and more general, notation will be used in the following sections.

- **Range of values for Execution time.** In this case, \( E_2 \) may be represented as a range of possible values instead, identified by minimum and maximum possible values for \( E_2 \). We now write the production rule as follows:

if \( P_1 \) \( \Delta t_1 \) then \( P_2 \) \( \Delta t_2 \) || \( [E_{2\text{min}}, E_{2\text{max}}] \).

and we require that the Execution time for the action must lie within this range

\( E_{2\text{min}} \) to \( E_{2\text{max}} \).

- **TimeOuts.**

- **Rules that have a timeout requirement.**

Essentially, a timeout monitors the time of execution for each action (or set of actions) for the successful rule. If the time for an action to be implemented exceeds a certain limit, then a timeout is indicated. Assuming that: \( \text{timeout} \) is the value indicating a timeout, \( t_{\text{fire}} \) is the real time when the rule fires, or is evaluated as true, and \( t_{\text{now}} \) is the current (real) time, then the notation to include timeouts is as follows:

if \( P_1 \) \( \Delta t_1 \) and \( t_{\text{now}} < t_{\text{fire}} + \text{timeout} \) then \( P_2 \) \( \Delta t_2 \) || \( [E_{2\text{min}}, E_{2\text{max}}] \).

In this representation, the rule is unconditionally true (given that all the conditions \( P_1 \) are also satisfied) only if the timeout has not occurred. Obviously as soon as the condition \( P_1 \) is satisfied, the action is initiated as the second test must return true initially unless the timeout time is zero in which case this rule will not fire.

Depending on the application, once the timeout occurs, the rule may be reevaluated as false and the associated actions abandoned, or a flag may be set to trigger off some other actions.

- **Rules that implement a timeout.** Usually, a timeout in this category takes on the following form. When a condition, \( A \) say, is satisfied, a timer is set off. This timer will be incremented (or decremented) as time passes. An example of a rule which implements this is as follows:

\[ \text{R1: if } A \text{ then set.timeout}_A = t_{\text{now}} + \text{timeout} \]

A second rule, which actually implements the timeout will monitor this timer. Typically, another condition \( B \) should be set to true before the timer reaches a predetermined timeout threshold. This can be written into a rule of the following form:

\[ \text{R2: if reached.timeout}_A \text{ and not } B \text{ then } P_3. \]

- **Persistence Span of conditions.** Another group of problems which must be considered in a real-time system arises from the need to represent situations which are only true if certain conditions have persisted for a certain time span. If we assume the following: \( t_{\text{persistent}} \) is the time span during which the condition must remain true, \( t_{\text{true}} \) is the real time when \( P \) is first evaluated as true, and \( t_{\text{now}} \) is the current (real) time, the value of \( t_{\text{true}} \) will be reset as soon as \( P \) is reevaluated as false before \( t_{\text{persistent}} \) is up. Then we can define the following representation:

if \( P \) and \( t_{\text{now}} = t_{\text{fire}} + t_{\text{persistent}} \) then \( P_2 \) \( \Delta t_2 \) || \( [E_{2\text{min}}, E_{2\text{max}}] \).

- **Range of time stamp values for an action.** So far, we have stated that an action \( P_2 \) may have a time stamp \( \Delta t_2 \) associated with it, which indicates that its value is true only until \( \Delta t_2 \) has elapsed. In more general situations however, we may find for example, that although an action is normally true for \( \Delta t_2 \), the occurrence of a particular event may result in the termination of the action even before \( \Delta t_2 \) is up. To cater for this, we can specify that the action part of a rule is only true until one of 2 or more possible time stamps, \( \Delta t_{2i} \), \( i = 1 \) to \( n \) (\( n \geq 2 \)), has passed. We write this as:

\[ \text{if } P \text{ and } t_{\text{now}} < t_{\text{fire}} + \text{timeout} \text{ then } P_2 \text{ || } [E_{2\text{min}}, E_{2\text{max}}]. \]
if $P_1 \mid \Delta t_1$
then $P_2 \mid \min(\Delta t_2_1, \Delta t_2_2, \ldots, \Delta t_2_n) \mid I E_2$.

Each of the $\Delta t_2_i$ may be understood as a time stamp which is satisfied (i.e., the time limit is up) as soon as the associated event $i$ occurs. Hence a given $\Delta t_2_k$, $1 \leq k \leq n$, may either:

1) always hold a fixed and known value, if event $k$ always occurs only after a known fixed period of time has elapsed (if at all) or
2) if event $k$ is an event which is dependent on parameters of the real-time system which changes in real time in a non-deterministic way, then $\Delta t_2_k$ may have varying values. Action $P_2$ is terminated as soon as one of the time stamps $\Delta t_2_i$ is satisfied.

An Interrupt is a special case of an event which may terminate an action. If event $n$ is the event associated with an Interrupt, then $\Delta t_2_n$ is a variable time stamp which is satisfied when an Interrupt occurs.

3 Illustration.

The NetManager is a real-time expert system which we have developed to solve the problem of Network Traffic Management of a national Telecommunication network. The semantics above have been applied to the Net Manager. The description of it is given in [1].

4 Summary.

This paper presents a method for incorporating real-time semantics into Production rules. The incorporation of real time semantics in frames is given in [1] and could not be described for reasons of space. It is expected that these representations may be used to capture the knowledge of any real-time expert systems in general.

References


