Reasoning About the Temporal Aspects of Interacting Agents*

Edjard Mota
email: edjard@dcc.fua.br
Departamento de Ciência da Computação
Universidade Federal do Amazonas
Av. Gen. Rodrigo Otávio, 3000 Mini-Campus
CEP: 69077-000 Manaus - Brasil

April 16, 2001

Abstract

The formalization of temporal reasoning for dealing with actions taking place at different grains of time has been investigated in the recent years. The main concern has been the representation of the real calendar and the mechanisms needed for mapping events "annotaded" with any convex an non-convex intervals to their annotation in the calendar. This work presents some results on the representation and reasoning about the actions of agents interacting at different time scales.

1 Introduction

The investigations on the representation and reasoning about the interaction of agents have not paid too much attention to the *temporal aspects of the interaction* itself. We should understand this as *the way models of agents influence each other as time goes by*, despite the fact that such influences may happen at different grains of time. In domains like simulation models of ecological and/or social systems there can be problems when one needs to integrate models (agents) specified at different time granularities.

In this work we shall see how a temporal logic programming approach can be used to formalize the behaviour of an agent when observing the change of other agents' attribute if suh information is relevant to its behaviour. Three main issues are involved here: a) the representation of information about periods at differente levels, b) the exchange of timing information across different models of autonomous agents, and c) the problems we face when combining both issues above in simulation models of autonoumous agents. We propose a way to tackle the problem (b) and address a brief discussion on (c).

The rest of this paper is organised as follows. In Section 2 we shall see the main motivation for this work . In Section 3 presents a logic programming approach to representing an agent's behaviour. Section 4 presents our solution for the problem this work aims to solve. In Section 5 some related work are addressed. Finally, Section 6 presents a brief discussion and the concluding remars.

2 Motivation and Related Work

2.1 The Problem to be Solved

When modeling how change takes place at different levels of abstraction on such systems, there can be three different situations of temporal interaction:

1. There are two interacting agents, say a_1 and a_2, in which their models are expressed at the same level and they update their states at the same tick of the clock. We shall say they are *aligned* in time.

*This work was partially supported by the Brazilian Research Council (CNPq), Lab-TIAMA grant.

2. There are two interacting agents, say a_1 and a_2, but their models are not expressed at the same level of granularity.

3. There are three interacting agents, say a_1, a_2 and a_3. Agents a_1 and a_2 have their models expressed at the same level, but their states change *non-aligned* in time.

The main questions we have to face are: *i)* what is the value of the attribute of each agent at a specific time in between two consecutive time steps at the agent's scale of time? *ii)* what are the values of an agent's attribute during a certain period of time? (for aligned and non-aligned periods in relation to changes on the agent's attributes).

3 Representing the Behaviour of Agents

Here, models of agents' behaviour are represented by an extension of Prolog, enhanced with a special unification for temporal labels [1]. The logical connectives for conjunction, disjunction, and implication are represented here by &, *or*, and $\Longleftarrow$, respectively. The only temporal operator used here is the throughout @ operator.

A classical formula is "annotated" (in the sense of [2]) with temporal expressions, and in this case it is called an *atomic temporal formula* if the associated classical formula is atomic. In the case of a classical well formed formula, when each of its elements or the whole formula is annotated, then we call it a *well formed temporal formula* (WFTF).

Temporal entities of the hierarchy of time are called *Pure temporal expressions* (PTE). The sorts of PTEs or temporal entities of interest for this work are defined as follows.

- $t(x_1, ..., x_n)$ to represent one moment of time. For instance, $t(16, 6, 2001)$ is a moment in a three level hierarchy of time.

- $P\ after\ T$ represents a time in the future obtained from T after a period of length P. The form of P can be either $p(Value, Scale)$, or $p(Value, Scale)\ plus\ P$, where *plus* is a constructor of composed lengths of time, and P is a period. For instance, $p(10, day)\ plus\ p(3, month)\ after\ t(2, 1, 2001)$.

Any temporal formula A @ T_i is written as A @ $T :: [(T, T_i)]$. A temporal temporal normal logic program has the form $\{C_1 :: \theta_1, \ldots, C_n :: \theta_n\}$, where each C_i is a temporal clause and θ_i is its temporal substitution framework (TSF), used to perform a special unification [1]. We now use this language and set notation to describe the model of an agent's behaviour.

Let Att_a be an attribute of an agent A which changes every P units of time, and Att_b be an attribute of another agent B which affects A. The *simulation clause scheme* (SCS) of A's behaviour which changes Att is given by

$$
\begin{aligned}
&value(Att, A, V_i)\ @\ T_i :: [(T_i, X_i)].\\
&value(Att, A, V_f)\ @\ T \Longleftarrow\\
&\quad value(Att, A, V_p)\ @\ T_p\ \&\\
&\quad V_L = \{V \mid value(Att_b, B, V)\ @\ T_j \text{ and}\\
&\qquad\quad T_j \text{ is included in } T_p...T\}\ \&\\
&\quad \mathcal{R}(V_p, V_L, V_f) :: [(T_p, X_p), (T, P\ after\ T_p)].
\end{aligned}
$$

where $value(Att, At, V_i)$ @ T_i is the initial state of Att and $\mathcal{R}(V_p, V_L, V_f)$ represents a sequence of sub-goals which relates V_p, V_L and V_f.

This clause scheme is used as an abstraction and it could be pluged within any other standard logic of action and change. The actual difference to standard computational logic is the mechanism of inference for dealing with non-aligned processes which we shall see next.

4 Reasoning about Interacting Agents

4.1 Agents Aligned in Time

For this case the deduction process tries to satisfy a given query relative to an attribute's value at an specific time T. If T_f is the final time point of the search interval with length P, then we start from the first value of the attribute and repeatedly get new instances of simulation clause for the changes of the attribute. We stop when either $T = T_f$ or T_f precedes T, where $T_f = P$ $after\ T_p$.

We extend the meta-interpreter of [1] as follows, where $var(X)$ is true if X is a variable and

$body\_of\_scs(A)$ is true if A can be matched with the body of a SCS.

$$\begin{array}{l} solve(value(Att, Ag, V) \ @ \ T, \theta_s, \theta_f) \Longleftarrow \\ \quad (T, T_1) \in \theta_s, \ \& \ var(T_1) \ \& \\ \quad body\_of\_scs(value(Att, Ag, V) \ @ \ T) \ \& \\ \quad clause(value(Att, Ag, V_i) \ @ \ T_i \Longleftarrow \top :: \theta_n) \ \& \\ \quad free\_search(value(Att, Ag, V_i) \ @ \ T_i, \theta_n, \\ \qquad\qquad value(Att, Ag, V) \ @ \ T, \theta_s, \theta_f). \end{array}$$

The declarative meaning of it is that *the value of the attribute Att of the agent Ag is V throughout T if T_1 is the temporal substitution of T in U_s, and T_1 is a variable, and $value(Att, Ag, V)$ @ T can be matched with the body of a simulation clause, and there is a clause $value(Att, Ag, Vi)$ @ $T_i \Longleftarrow \top$ with TSF θ_n, and the search free of fixed time yields a solution for $value(Att, Ag, V)$ at time T, with T in θ_s, and θ_f is its TSF.*

In the $free\_search/5$ the use of new instances of SCS involves temporal combination to generate them. This shall be represented by the predicate $inst\_simul\_cls/5$ which associates X_p @ T_p of the body of a SCS and its TSF θ to the head of this new clause instance, the constraints after X_p @ T_p, as well as in the new TSF resulting from this new instance. This is formalised as follows.

$$\begin{array}{l} free\_search(X \ @ \ T_i, \theta_i, X \ @ \ T, \theta, \theta_f) \Longleftarrow \\ \quad temp\_comb(\theta_i, T_i, \theta, T, \theta_f). \\ free\_search(X_p \ @ \ T_p, \theta_p, X_r \ @ \ T_r, \theta_s, \theta_f) \Longleftarrow \\ \quad inst\_simu\_cls(X_p \ @ \ T_p, \theta_p, X_f \ @ \ T, C, \theta_s) \ \& \\ \quad \theta_{s'} = \theta_s \setminus \{(T_p, V_p)\} \ \& \\ \quad temp\_comb((T_f, V), \theta_f, (T_s, V_s), \theta_{s'}, \theta_n) \ \& \\ \quad solve(C, \theta_n, \theta_e) \ \& \\ \quad free\_search(X_f \ @ \ T_f, \theta_e, X_r \ @ \ T_r, \theta_s, \theta_f). \end{array}$$

4.2 Agents Aligned and Non-Aligned in Time

The idea now is to align the search so that the last time stamp found, or V_f will always be at the given T of the query, i.e. $T = T_f$ will always hold. This policy forces the search to take any new influence into account. Formally we have the following $solve/3$ definition for this case.

$$\begin{array}{l} solve(value(Att, Ag, V) \ @ \ T, \theta_s, \theta_f) \Longleftarrow \\ \quad (T, T_1) \in \theta_s \ \& \\ \quad \neg \ var(T_1) \ \& \\ \quad body\_of\_scs(value(Att, Ag, V) \ @ \ T) \ \& \\ \quad clause(value(Att, Ag, Vi) \ @ \ T_i \Longleftarrow \top :: \theta_i) \ \& \\ \quad (T_i, T_2) \in \theta_i \ \& \ T_1 \text{ does not precede } T_2 \ \& \\ \quad T_1 = P_a \text{ after } T_2 \\ \quad aligned\_search(P_a, value(Att, Ag, V_i) \ @ \ T_i, \theta_i, \\ \qquad\qquad value(Att, Ag, V) \ @ \ T, \theta_s, \theta_f). \end{array}$$

The declarative meaning of this clause is similar to the previous one. In what follows we show the possible cases and their specification, where $change(Att, Proc)$ is true if Att is changed by process $Proc$, $scale(Agent, Proc, S)$ is true if $Agent$ has a process $Proc$ which runs at scale of time S, and $smaller\_period(P_1, P_2)$ is true if period P_1 is a length of time smaller than P_2.

$$\begin{array}{l} aligned\_search(0, X \ @ \ T_n, \theta_n, X \ @ \ T_s, \theta_s, \theta_f) \\ \quad \Longleftarrow \\ \quad \theta_{s'} = \theta_s \setminus \{(T_s, V_s)\} \ \& \ \theta_{n'} = \theta_n \setminus \{(T_n, V_n)\} \ \& \\ \quad temp\_comb((T_s, V_s), \theta_{s'}, (T_n, V_n), \theta_{n'}, \theta_f). \\ aligned\_search(P_a, X_i \ @ \ T_i, \theta_i, X \ @ \ T, \theta_s, \theta_f) \\ \Longleftarrow \\ \quad P_a > 0 \ \& \\ \quad X = value(Att, Agent, \_) \ \& \\ \quad change(Att, Proc) \ \& \\ \quad scale(Agent, Proc, S) \ \& \\ \quad (\ (equivalents(P_a, p(1, S)) \ \& \\ \qquad search(X_i \ @ \ T_i, \theta_i, X \ @ \ T, \theta_s, \theta_f)) \ or \\ \qquad (\ \neg equivalents(P_a, p(1, S)) \ \& \\ \qquad\quad (\ (smaller\_period(P_a, p(1, S)) \ \& \\ \qquad\qquad \theta_f = \theta_s) \ or \\ \qquad\qquad (smaller\_period(p(1, S), P_a) \ \& \\ \qquad\qquad mod\_decomposed(P_a, S, P_1) \ \& \\ \qquad\qquad (\ (P_1 = p(\_, S) \ \& \\ \qquad\qquad\quad search(X_i \ @ \ T_i, \theta_i, X \ @ \ T, \theta_s, \theta_f)) \ or \\ \qquad\qquad\quad (P_1 = p(D_1, S_1) \ plus \ p(\_, S) \ \& \\ \qquad\qquad\quad temp\_comb(T_j, [(T_j, V_j)], T, \\ \qquad\qquad\qquad [(T, p(D_1, S_1) \ after \ T_i) | \theta_i], \theta_j) \ \& \\ \qquad\qquad\quad search(X_i \ @ \ T_j, \theta_j, X \ @ \ T_s, \theta_s, \theta_f)))))). \end{array}$$

The $search/5$ stops when the the final time is reached, otherwise a new instance of simulation clause is matched with the value of the attribute at the previous time, the constraints are solved and the search re-starts with this new value. Formally we have.

$search(X \ @ \ T_n, \theta_n, X \ @ \ T_s, \theta_s, \theta) \Longleftarrow$
  $\theta_{n'} = \theta_n \setminus \{(T_n, V_n)\} \ \& \ \theta_{s'} = \theta_s \setminus \{(T_s, V_s)\} \ \&$
  $temp\_comb((T_s, V_s), \theta_{s'}, (T_n, V_n), \theta_{n'}, \theta_f).$
$search(X_i \ @ \ T_i, \theta_i, X_r \ @ \ T_r, \theta_r, \theta_f) \Longleftarrow$
  $(T_i, V_i) \in \theta_i \ \& \ (T_r, V_r) \in \theta_r \ \&$
  V_i precedes V_r &
  $inst\_simu\_cls(X_i \ @ \ T_i, \theta_i, X_f \ @ \ T, C, \theta) \ \&$
  $\theta' = \theta \setminus \{(T, V)\} \ \&$
  $temp\_comb((T_j, V_j), [\,], (T, V), \theta', \theta_j) \ \&$
  $((precedes(V_r, V_j) \ \& \ X_r = X_i \ \& \ \theta_f = \theta_i) \ or$
   (V_r does not precede V_j & $solve(C, \theta_j, \theta_{j'}) \ \&$
   $search(X_j \ @ \ T_j, \theta_{j'}, X_r \ @ \ T_r, \theta_s, \theta_f)).$

4.3 Representing Interaction Between Agents

Two agents A_i and A_j interact at different grains of time, and an attribute Att_j of A_j affects an attribute Att_i of A_i. Agent A_i has to "observe" the changes on the values of Att_j. The *observer process* (OP) will get the progression of the values of Att_j, in a list L, during a specific period of time of the length of k. This idea is depicted in Figure 1.

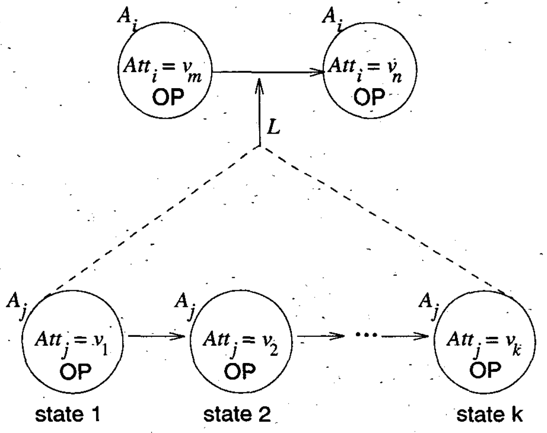

Figure 1: Interaction between agents A_i and A_j, where the OP of A_i get the list L of values of Att_j during a period k units of time.

The extension of the meta-interpreter for dealing with it is simply to call for the predicate $progress\_observed(value(Att, Agent) \ @ \ T, P, L, \theta_s, \theta_f)$. As the value of the observed attribute at the right ending point is unimportant to compute the state of the agent for that time, then the period observation is open on the right. We use $length\_of\_time(I_L, P)$ to associate any linear interval to its length (i.e. its ending points inclusive). From this point there are two possible computations as we show as follows. First the case where agents are aligned.

$progress\_observed(value(Att, Agent) \ @ \ T,$
    $P, [V|R], \theta_s, \theta_f) \Longleftarrow$
  $change(Att, Proc) \ \& \ scale(Agent, Proc, S) \ \&$
  $(T, T_p) \in \theta_s \ \& \ length\_of\_time(T_p ... Tf, P) \ \&$
  $solve(value(Att, Agent, V) \ @ \ T, \theta_s, \theta_n) \ \&$
  $compute\_rest(value(Att, Agent, V) \ @ \ T,$
    $T_f, P, S, \theta_n, \theta_f, R).$

Second, for the case where the initial state of the attribute is either included within the given period or the initial state occurs at some time after the last time of observation, we have.

$progress\_observed(value(Att, Agent) \ @ \ T, P,$
    $Prog, \theta_s, \theta_f) \Longleftarrow$
  $change(Att, Proc) \ \& \ scale(Agent, Proc, C) \ \&$
  $(T, B_p) \in \theta_s \ \& \ length\_of\_time(B_p ... B_f, P) \ \&$
  $\neg solve(value(Att, Agent, \_) \ @ \ T, \theta_s, \_) \ \&$
  $solve(value(Att, Agent, V) \ @ \ T_i, \theta_s, \theta_n) \ \&$
  $(T_i, B_i) \in \theta_n \ \&$
  $(\neg precedes(B_f, B_i) \ \&$
   $Prog = [V|R] \ \&$
   $compute\_rest(value(Att, Agent, V) \ @ \ T_i,$
   $B_f, P, C, \theta_n, \theta_f, R) \ or$
   $precedes(B_f, B_i) \ \&$
   $Prog = [\,])$

In the rest of the progress, represented by $compute\_rest/5$, the important thing to describe is how the period of observation relates to the agent's scale of change. For the simplest case where P is smaller than the length of C, we have four different cases as depicted in Figure 2. First, P starts at a time aligned to an updating time step of the agent's attribute (a). Second, P finishes at an updating time (b). Third, it is in between two consecutive updating time steps, (c), and finally it includes a single updating time step (d). This gives us the following specification.

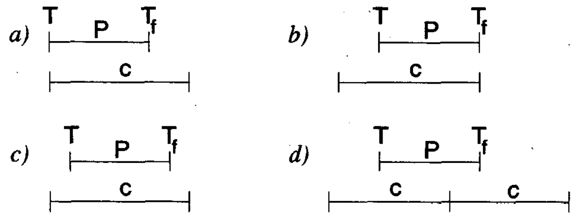

Figure 2: Case 1 - *a)* P starts aligned; *b)* ends aligned; *c)* P is included in between two consecutive time steps; *d)* P includes a single time step.

$compute\_rest(value(Att, Agent, \_)\ @\ \_, T_f,$
 $p(D, C_1), C_2, \theta_s, \theta_f, [V_f]) \Longleftarrow$
 $\neg\ (C_1 = C_2)\ \&$
 $smaller\_period(p(D, C_1), p(1, C_2))\ \&$
 $solve(value(Att, Agent, V_f)\ @\ T, [(T, T_f)|\theta_s], \theta_f).$

The other case is the one in which the length of c is smaller than or equal to the length of P. This can lead us to other possibilities which we classify in two ways. In the first case, depicted in Figure 3, P can be decomposed into C and either it matches the actually updating time steps of the agent's attribute (a), or it may be non-aligned (b). It may also not be decomposable and then it matches at the beginning (c) but not at the end, or vice-versa (d).

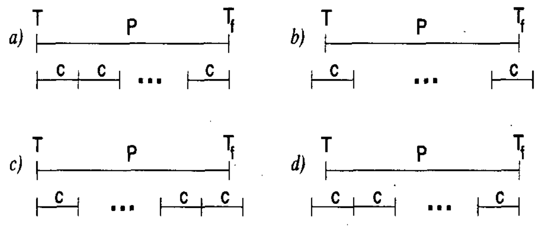

Figure 3: Case 2 - The Period of observation is modularly decomposable into C, (a) and (b), and P is not modularly decomposable, (c) and (d).

The final case is where P and unitary length at c are equivalent and either (a) they are totally aligned (starts and ends at the same time step), or (b) they are non-aligned. This is depicted in Figure 4.

What has to be done, in the context of $compute\_rest/5$, is simply to check if P has a length greater than or equal to the length of c, then it should compute the progression of the value of the attribute until time T_f, taking theses cases above into account. This is formally specified as follows.

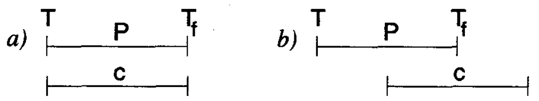

Figure 4: Case 3 - *a)* P is aligned to c; *b)* P and c are non-aligned.

$compute\_rest(value(Att, Agent, V)\ @\ T_i, T_f,$
 $p(D, C_1), C, \theta_s, \theta_f, R) \Longleftarrow$
 $smaller\_period(p(1, C), p(D, C1))\ \&$
 $evolve(value(Att, Agent, V)\ @\ T_i, T_f, C, \theta_s, \theta_f, R).$

Now, there are three possibilities when $evolve/4$ is called:

1. T_i is equal to T_f, then R should be associated with an empty list.
2. T_i is not equal to T_f, then apply resolution between $value(Att, Agent, V)\ @\ T_i$ and the body of a simulation clause

   $value(Att, Agent, V_j)\ @\ p(1, C)\ after\ T_i \Longleftarrow$
   $value(Att, Agent, V_j)\ @\ p(1, C)\ after\ T_i\ \&$
   $Constraints.$

   Either:

   (a) T_f precedes T_j, i.e. $T_i\ after\ p(1, C)$ or the next value to be computed, then find V_f at T_f associate R to $[V_f]$

   (b) T_f does not precedes T_j, then solve the $Constraints$, and associate R to $[V_j|R_1]$ and compute the rest of progression in R_1 from T_j until T_f.

5 Related Work

This work identified four important cases when the period of an agent's observation is shorter than the time scale of the other agent. We can related them to the Allen's interval calculus [3]. Here we described ending interval, beginning interval, wholly within the interval, and overlapping.

Other approaches on temporal granularity did not concerned with this identification, [4, 5]. A logic programming approach similar to this work is Chronolog(MC) [6] which is flexible for multiple granularities, but no exploration was made on representing interacting agents working at different clocks.

Kraus et. al. [7] proposed a strategic model of interaction that takes the passage of time during the negotiation process itself into account. They investigated agents that lose over time and need to share resources. They base their investigation in cooperation of agents while ours does not need to assume cooperation. This means that our approach is not suitable for strategic models.

6 Discussions and Final Remarks

Consider a scenario of interaction as depicted in Figure 5 where all agents work at grain 7. The goal is for the time stamp $t(20,1,1)$, and agents a_3 and a_4 both affect a_1 and a_2 but are introduce some time later. According to policy we used the initial state for the search is $t(6,1,1)$. However, this is not fully accurate because a_3 would has affected a_1 since $t(4,1,1)$.

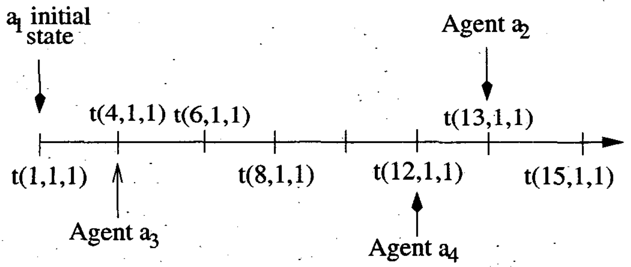

Figure 5: Another situation in which two other non-aligned agents are present.

Alternatively we could take the earliest initial state of all agents affecting (in this case) a_1 and start the search from it. Note that in this situation the last change in a_1 before $t(13,1,1)$ is $t(11,1,1)$. Thus, the value assumed for a_1 state at $t(13,1,1)$ would be the one found at $t(11,1,1)$. However, this would be no better than the one proposed here because it would not take the influence of a_4 upon a_1 into account. This other agent has been introduced at $t(12,1,1)$ which is in between $t(11,1,1)$ and $t(13,1,1)$. As a matter of fact this approach is symmetric to what is proposed here.

This problem only shows how complicated can be the use of models of interacting autonomous agents to predict the future within a temporal structure. Our main contribution is to raise the problems of communication between interacting agents at different grains of time, and to address the possible initial direction to cope with it.

References

[1] E. Mota, "Cyclical and granular time theories as subsets of the herbrand universe," in *Principles of Knowldege Representation and Reasoning*, 2000.

[2] T. Fruhwirth, "Annotated constraint logic programming applied to temporal reasoning," Tech. Rep. 22, ECRC, July 1994.

[3] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, 1983.

[4] E. Ciapessoni, E. Corsetti, A. Montanari, and P. S. Pietro, "Embedding time granularity in a logical specification language for synchronous real-time systems," *Science of Computer Programming*, vol. 20, no. 1, pp. 141–171, 1993.

[5] A. Montanari, "A metric and layered temporal logic for time granularity, synchrony and asynchrony." First International Conference on Temporal Logic, July 1994.

[6] C. Liu and M. A. Orgun, "Dealing with multiple granularity of time in temporal logic programming," *Journal of Symbolic Computation*, vol. 22, no. 5–6, 1996.

[7] S. Kraus, J. Wilkenfeld, and G. Zlotkin, "Multiagent negotiation under time constraints," *Artificial Intelligence*, vol. 75, no. 2, 1995.