Representing Interaction of Agents at Different Time Granularities*

Edjard Mota & David Robertson

Department of Artificial Intelligence
The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN Scotland
email: {edjardm,dr}@aisb.ed.ac.uk

April 16, 1996

Abstract

In this paper we describe **NatureTime** *logic which we use to represent and reason about the behaviour of interacting agents (in an ecological domain), which behave at different time granularities. Although the traditional application fields of temporal representation and reasoning still raise many interesting theoretical issues, we have been investigating some practical problems of ecological systems which suit different representations of time than those embodied in traditional simulation models of ecosystems. These seem well suited to reconstruction using temporal logic programs.*

1 Introduction

An understanding of the world generally needs a way to represent processes at different levels of granularity [1]. This may be done in relation to time, space, and the structural organisation of things we are interested in observing, representing and in many cases simulating. In relation to time and space, it is common that that processes working at finer levels of time can only be observed if we also change to a finer level of space, in order to observe the changes they may produce in the environment. An interesting aspect of these levels of abstraction is to observe how things, which happen at the lowest levels will affect the others at higher levels of granularity. This process is usually called scaling up. Things become complicated when the higher levels also affect, at longer periods of time, the lower levels.

*This work and the first author (on leave for PhD from the Department of Computer Science of the University of Amazonas, Manaus, Brazil) are sponsored by the Brazilian Ministry of Education, grant nọ 01723/93-8/CAPES.

The scaling up process is highly relevant to the development of of simulation models of ecosystems. Many factors make scaling difficult for ecological modellers but a major obstacle is that each individual model is devised to run on its own, single-level, time scale. It is argued in [2] that the use of a temporal logic-based language to specify such simulation models as agents can provide better representations for the behaviour and interaction of eco-agents working at different scales of time. The usefulness of using **NatureTime** logic for the specification of such agents has already been shown [3].

In this paper we present some results on the representation of interacting agents developed in [4], which extends the **NatureTime** logic. We also address some new problems to be tackled when asynchronous agents are introduced into the environment where there are other interacting objects.

In Section 2, we present one example as a motivation for this work, and we also address some related works. In Section 3, we make a brief presentation of the **NatureTime** logic. In Section 4, we show how to enhance the logic with mechanisms for the representation of interacting eco-agents. In Section 5, we show an application of **NatureTime** to the specification and simulation of eco-agents working at different time granularities. Finally in Section 6 we give some concluding remarks.

2 Motivation and Related Works

2.1 Two Interacting Ecological Agents

> *Example*: "We have a model of tree growth, expressed on a weekly time scale. This must interact with a model of a insect pest which moves up and down the tree on a daily time scale. The tree has its growth rate reduced

by 0.02 every day the pest moves above 8 meters. The pest moves continuously up and down, at a rate of 2 meters per day, reversing direction when it reaches the top or bottom".

This example shows an interaction between two entities working at different time scales. The types of question that we might want to answer in problems like this are: "what is the value of the attribute of each agent at an specific time?" or " when will some attribute of either agent have a certain fixed value?".

The behaviour of each of these agents could be represented by means of differential equations, which is an expressive way of representing the *continuity* of their behaviour. But a continuous representation of time is not always best for ecological models. For example, if we want to represent the behaviour of agents which immigrate and emigrate from one population to another, then it is difficult to represent this using continuous functions. A more usual way of modelling the changes in the state of such agents is to perform them at the time step of their corresponding granularity. This yields a discrete approximation to continuous models.

However, a discrete approach does not allow us to compute the value of some attribute at a time in between two consecutive time steps. To overcome this, interpolation could be needed. In this case we have to assume that there is no interaction between the agents that may affect the attributes in question. Otherwise, we cannot estimate the next value in the consecutive time step. Such a future value will depend on the value we are trying to find for the time in between the future and the past. In this work we just take the value of the most recent temporal entity in the past.

2.2 Related Work

Granularity is very important if we intend to look at the world at different levels of abstraction, when switching from one level to another may be necessary for the comprehension of the phenomena being observed [1]. A theoretical analysis of hierarchical time intervals was proposed in [5], where an elaboration over the interval calculus [6] is done to achieve a time framework where units of time can be specified. This work extends the idea of *convex* to *union-of-convex* intervals [7], where there may exist gaps between *convex* intervals. This allows the representation of collections of intervals and limited expressions of cyclicity. However, it was not implemented, as far as we know, so the pragmatics of using it for computation are obscure.

A similar approach was proposed in [8], where the basic idea in this approach is to use a set of primitive collections to specify other collections by using two operators, *slicing* and *dicing*, in order to select intervals from collections of intervals. Each such a primitive is defined by specifying the intervals of which it is composed. In this approach, circular aspects of time can be obtained from the *δ-values* which are treated as if they were a circular list. Although this approach was shown to be useful for reasoning about scheduling, it does not deal with different granularities of time.

In [9, 10] there is proposed a many-sorted first order logic augmented with temporal operators and a metric on time to deal with time granularity. This is achieved by introducing *contextual* and *projection* operations into topological logic [11]. This has been applied in the context of planning systems [12], to achieve plan actions at different scales of time and reduce the computational complexity of such systems.

In a parallel work to ours, [13] proposes an interesting framework of time based on the notion of *calendars* as being cyclic temporal objects. The difference is in the way in which such temporal objects are conceived. This approach, as the others does not include the cyclical aspect of time in their models.

The theory developed in [14] was an attempt to provide a logic based language to represent concepts of time, following closely the forms of expression used informally in descriptions of ecological systems in a target domain. Many of these descriptions include the idea of cyclical processes at many levels of time granularity. In the next section we will summarise the time theory developed to deal with these.

3 A Linear-Cyclic Hierarchy of Time

In this section we will briefly present the time hierarchy of the **NatureTime**. We refer to [14] for more details about the temporal reasoning interpreter, which is basically a standard Prolog meta-interpreter with restricted forms of unification on temporal labels.

3.1 Basic Assumptions

By analysing the temporal knowledge about seasonal cycles of ecological knowledge [15], as in sentences like "Coffee is harvested from August up to April", a hierarchy of time cycles was proposed in [14] in order to represent and reason about cyclical events. A natural mathematical structure for representing cycles is modular arithmetic (also called clock arithmetic [16]). For instance, years are modular sets of months, [lunar] months are sets of days, and so on. Note that this hierarchy of cycles also allows us to define many levels of temporal granularity.

3.2 Elements of the Language

The language is Prolog based enhanced with some special symbols, terms, and a unification algorithm for temporal labels, which also works over terms of second order which not include predicate symbols.

Vocabulary - It is formed by symbols for variables, constants, functions and predicates as follows.

- two countably infinite sets of variables $\mathcal{L}_v$, and $\mathcal{L}_{tv}$, where x_i, y_i, z_i are variables of $\mathcal{L}_v$, s_i, t_i, u_i are variables of $\mathcal{L}_{tv}$.
- a finite set $\mathcal{L}_c$ of constants.
- a finite set $\mathcal{L}_f$ of non-temporal function symbols of the form f^n, where n is the arity of the function.
- a finite set $\mathcal{L}_{tc}$ of temporal constants defined as $\{lowest, flowtime, infinity, smallest\} \cup \mathcal{T}_C$, where $\mathcal{T}_C = \{c_1, \ldots, c_n\}$, and each c_i is a special constant or names of temporal classes.
- a finite set of temporal function symbols f_t^n, where n is the arity of the function, $\mathcal{L}_{tf} = \{p^2$, i^2,t^n,$...^2$,$plus^2$, $after^2$, $before^2$, $of^2\}$, where $...^2$, $plus^2$, $after^2$, $before^2$, of^2 are all of arity 2 but written using infix notation.
- a finite set $\mathcal{L}_p$ of predicate symbols p_i^n, $i \geq 1$, where $n > 0$ is the arity of p_i, where *mod_temp_class* is special predicate symbol of arity 3, and *on* is a special predicate of arity 2.
- the set $\mathbb{Z}$ of integers is also part of the vocabulary.
- the propositional connectives for conjunction, disjunction, and implication are represented here by &, $\vee$, and $\Longleftarrow$, respectively. The truth value for true is represented by $\top$, and false by $\perp$.
- the temporal symbol "@".

Classes of Expressions - By using these symbols we define the following classes of expressions.

By using these symbols we define the following classes of expressions, where the capital letters A, B, C are used for formulae.

. a *logical term* is recursively defined as

  - a variable $x \in \mathcal{L}_v$ is a term
  - a constant $c \in \mathcal{L}_c$
  - if $t_1, \ldots, t_n$ are terms and $f \in \mathcal{L}_f$ and has arity n, then $f^n(t_1, \ldots, t_n)$ is a term.

  Examples of logical terms are *maize*, *grass*, $height(tree(t_1))$, $biomass(forest(f_1))$.

. a *temporal term* (TT) is a temporal variable $s \in \mathcal{L}_{tv}$, or a temporal constant $c_t \in \mathcal{T}_C$, or a temporal function symbol $f_t \in \mathcal{L}_{ft}$ in one of the following forms.

  - a *period*, recursively defined as a 1) *single period* $p(s, m)$ where $s \in \mathbb{Z}$ and $m \in \mathcal{T}_C$; 2) a *composite period* P *plus* P', where P is *single period*, and P' is a *period* term.
  - a *cyclical interval* $i(s...t, m)$, where s, $t \in \mathbb{Z}^+$ and $m \in \mathcal{T}_C$
  - a *smallest temporal entity* $t(t_1, \ldots, t_k)$, where for n as the number of elements in $\mathcal{T}_C$, then $k \leq n$, each $t_i \in \mathbb{Z}^+$, and each i correspond to exactly one element of $\mathcal{T}_C$.
  - a *linear interval* $s_1...s_2$, where s_1 and s_2 are in the form $t(t_1, \ldots, t_k)$.
  - a *collection interval*
    - α where $\alpha \in \mathcal{T}_C$.
    - $f_t(n)$, where $f_t \in \mathcal{L}_{tf}$, and $n \in \mathbb{Z}^+$, and f_t corresponds to one unique symbol $\alpha \in \mathcal{T}_C$. For instance, the function symbol of $week(1)$ corresponds to a *week* of $\mathcal{T}_C$
    - $\alpha(n)$ *of* S, where $f_t \in \mathcal{L}_{tf}$ (same as previous item), and $n \in \mathbb{Z}^+$, and S is a *collection interval*.

Examples of TT are, for example, $i(11...5, month)$ to mean *interval of time between November and May*, $p(1, day)$ *after* $t(17, 7, 1994)$ *one day after 17th July, 1994*, $day(1)$ *of week* means *all Mondays*, and $last(day(2)$ *of week of* $month(2)$ *of* $year(1996))$ means *the last Monday of February of 1996*. The *collection interval* is more general than a *cyclical interval* in the sense that it may define cyclical and noncyclical intervals. For instance, $day(2)$ *of week of month* represents all Mondays of all months. However, $day(2)$ *of week of* $month(2)$ *of* $year(1996)$ is not cyclical because it represents the sequence of days 5th,12th, 19th, and 26th of February 1996.

. a pure temporal expression (PTE)

  - t, if t is a TT, but not a *period term*.
  - p *after* t, where p is a *period* term, and and t is a pure temporal expression.

Examples of PTE are $p(3, month)$ *after* $i(10...11, month)$, $p(24, year)$ *after* $t(17, 7, 1970)$, $hour(4)$ *of* $day(1)$ *of* $week(1)$ *of* $month(3)$.

- an atomic formula (AF) is
  - $\top$ and $\bot$ are atomic formulae
  - if $term_1, \ldots, term_n$ are logical terms and $p^n \in \mathcal{L}_p$, then $p^n(term_1, \ldots, term_n)$ is an atomic formula
- classical atomic formulae can be annotated with a PTE by using the temporal operators "@", forming an atomic temporal formula (ATF), i.e.
  - A, if A is an AF, then it is an ATF.
  - A @ T, where A is an AF and T is a PTE, is an ATF

  For instance, $harvested(maize, highlands)$ @ $i(12...1, month)$ means that *maize is harvested throughout the whole interval of time from December up to January*.
- *body* is in one of the forms A & B, $A \vee B$, or C, where A and B are bodies and C an ATF. A typical example of a body is $time\_between(i(2...4, month), p(3, month))$ & $harvested(maize, high\_lands)$ @ $i(12...1, month)$.
- a well formed temporal formula (WFTF) is
  - if A is an ATF, then A is a WFTF with an *empty body*
  - if A is an ATF and $B \neq \top, \bot$, and B is a body then

    $A \Longleftarrow B$ is a WFTF, and A is called the *head*

  A WFTF where its *head* is an AF, and *body* is formed only by AF is a Prolog clause with no temporal contents. An example of a WFTF is

  $$harvested(Crop) @\ p(D, C)\ after\ T \Longleftarrow planted(Crop)\ @\ T.$$

  Which means *if a Crop is planted throughout an interval T, then it is harvested a D units C of time after T.*

3.3 The Linear-Cyclic Hierarchy Structure

The temporal structure used is a 5-tuple $\mathcal{LC}_H =_{def} \langle \mathcal{T}_h, \mathcal{E}, \overset{t}{\succ}, T, < \rangle$, where $\mathcal{T}_h$ is a finite partially ordered set of *modular temporal classes* (MTCs), $\mathcal{E}$ is a non empty set of temporal entities (or units of time), $\overset{t}{\succ}$ is a binary relation of temporal succession within the hierarchy (the properties of this relation are described in the Appendix A), T is the set of integers ordered by the binary relation of precedence $<$. The set $\mathcal{T}_h$ is composed of two sets $\mathcal{M}_h$ and $\mathcal{F}$ $\{mc_1, \ldots, mc_n\}$. The second is called *fluctuating* MTC, and the former is the main time hierarchy which is induced by the following relations.

. $mc_{n+1} =_{def} mod\_temp\_class(flow\_time, c_n, infinity)$, where $c_n \in \mathcal{T}_C$

. $mc_i =_{def} mod\_temp\_class(c_i, c_{i-1}, m_v)$, where $1 < i \leq n$, $c_i, c_{i-1} \in \mathcal{T}_C$, and $m_v \in \mathbb{Z}^+$

. $mc_1 =_{def} mod\_temp\_class(c_1, lowest, smallest)$, where $c_1 \in \mathcal{T}_C$. Each c_i is called the name of the class mc_i. The flow of time is a sequence of instances of c_n, where each one denotes a stage in a cycle and therefore can recur. For instance, a simple hierarchy which includes temporal classes for a simplified calendar model of time could be defined as follows.

$mod\_temp\_class(flow\_time, year, infinity).$

$mod\_temp\_class(year, month, 12).$

$mod\_temp\_class(month, day, 30).$

$mod\_temp\_class(day, lowest\_level, smallest).$

A view of this hierarchy is depicted in in Figure 1.

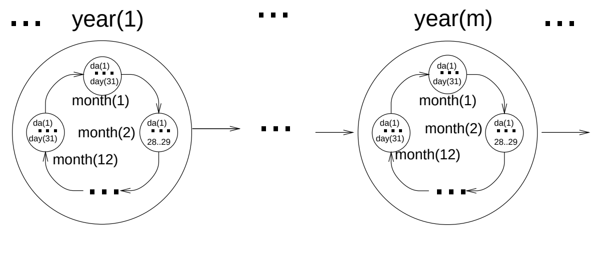

Figure 1: A view of the *Linear-Cyclic Hierarchy* of time. The inner circles represent lower levels of time granularity like *day* and *month*, while the outest circles represent the highest level, like *year*.

These relations define the flow of time, represented by $flow\_time$, as a linear and infinite ($infinity$) sequence of *years*, *year* as a MTC of 12 *month*, *month* as a MTC of 30 *days*, *day* as the smallest ($smallest$) time interval. Note that in this hierarchy we consider months as regular MTC, which is not the case in the real calendar. Although the logic allows us to define real calendars, we will omit it from this paper.

3.4 A Meta-Interpreter for NatureTime

In this section we present a meta-interpreter based on [17] for the **NatureTime** rather than a formal semantics. As we treat temporal reasoning as a problem of unifying PTEs, then we need to control the part of the unification which deals with it. Because of this, the standard $solve/1$ predicate is changed to be a binary relation $solve/2$. Basically, the meta-interpreter accepts queries which are temporal formulae. It succeeds if the query holds throughout some interval within the given PTE. Since it may not hold across the entire interval it is necessary to also show what the restricted PTE is - hence the second argument. In this way we have the following meta-interpreter.

$solve(A \ @\ T1,\ A\ @\ T3) :-$
  $\neg A = (\_ \ \&\ \_),$
  $A\ @\ T2,$
  $temp\_unify(T1, T2, T3).$
$solve(A\ @\ T, A\ @\ RT) :-$
  $A = (A1\ \&\ A2),$
  $solve(A1\ @\ \_, A1\ @\ T1),$
  $solve(A2\ @\ \_, A2\ @\ T2),$
  $temp\_unify(T1, T2, RT1),$
  $temp\_unify(T, RT1, RT).$
$solve(A\ \&\ B, A1\ \&\ B1) :-$
  $solve(A, A1),$
  $solve(B, B1).$
$solve(A\ or\ B, R) :-$
  $(solve(A, R)\ ;$
  $solve(B, R)).$
$solve(A\ @\ T1, A\ @\ T3) :-$
  $A\ @\ T2 \Longleftarrow Precondition,$
  $solve(Precondition, \_),$
  $temp\_unify(T1, T2, T3).$

As clauses 3 and 4 are just another way to re-write the standard clauses for logical conjunction and disjunction, we assume their usual interpretation. We have the following interpretation.

1 - *A is true throughout $T3$, if A is not a composite event, and A is true throughout $T2$, and $T3$ is the temporal unification of $T1$ and $T2$.*

2 - *A is true throughout T if, A is a composite event consisting of $A1$ & $A2$, and $A1$ is true throughout $T1$, and $A2$ is true throughout $T2$, and $RT1$ is the temporal unification of $T1$ and $T2$, and RT is the temporal unification of T and $RT1$.*

5 - *A is true throughout $T3$ if, the $A\ @\ T2$ is the head of a temporal Horn clause with body $Precondition$, and the body can be solved, and $T3$ is the temporal unification of $T1$ and $T2$.*

Briefly, the temporal unification concerns with the reduction of PTE to canonical forms of TT, and then unifying them by performing modular or linear matching using the usual relation between time intervals [6]. The TTs S and T unify if one of the following cases applies.

1. *$S = s_1 ... s_2$ and $T = t_1 ... t_2$, then either they overlap or one is included into the other.*
2. *$S = i(s_1\ ...\ t_1, C)$ and $T = i(t_1 ...\ t_2, C)$, and $mod\_temp\_class(\_, C, M)$ holds, M is an integer, and $i(s...t, C)$ is the modular match between S and T.*
3. *$S = i(s_1 ... s_2, C)$ and $T = t_1 ... t_2$, if there is one time instance $S' = s'_3 ... s'_2$ of S, and T and S' unify.*
4. *$T = t_1 ... t_2$ and $S = i(s_1 ... s_2, C)$, then S and T also unify.*

The *time_instance* creates one linear instance of a cyclical interval. As a cyclical interval represents a collection of linear intervals, there may exist many instances of it in the level of linear intervals.

4 NatureTime for Eco-Agent Specification

This section shows an extension of the meta-interpreter of the **NatureTime** for reasoning about the state of an agent at any time in between two consecutive time steps. To this, we take some advantage of a common way of representing simulation models.

4.1 Dealing with Simulation Clauses

The type of temporal knowledge in a simulation model can usually be represented in a standard clause schemata we call *simulation clause*. If A is the specification of a simulation model it is usually in the form

$$
\begin{array}{l}
A'\ @\ P\ after\ T \\
\quad \Longleftarrow \\
\quad A\ @\ T\ \& \\
\quad \mathcal{R}(A, A').
\end{array}
$$

where P is normally in the form $p(1, C)$ and C is a modular temporal class of the time hierarchy, and the predicate $\mathcal{R}$ is intended to represent the sequence of formulae which involves A and A' to produce their relationship within the flow of time. Finally, A and A' have the same predicate symbol and the same arity. Usually they are in the form $value(Attribute, Agent, V)$, i.e. the value of an $Agent$'s $Attribute$ is V. In order to allow **NatureTime** to reason about those queries of 2, taking into account this particular type of clause, we have the following solution.

1. check if the given temporal entity is a fixed time
2. check if there is a simulation clause such that the formula matches with some part of its body, and that the proposition A has the same structure (i.e. the same predicate symbol and same number of arguments),

3. find one solution for $A1 \ @ \ Ti$, where Ti is a variable
4. if the solution found is the proposition searched within the the specified interval, then the search stops.
5. otherwise, it the PTE of the head of the simulation clause is the future of T then the last state of A is assumed to be the required value, and the search stops.
6. otherwise the constraints represented in the $\mathcal{R}$ relation are attempted to be solved, and the future state of $A1$ throughout interval Ti is the head of the simulation clause, and we back to step 4.

4.2 Representing Interaction Between Agents

As **NatureTime** offers mechanisms for specifying simulation models working at different levels of time granularity, and ecological systems usually involve entities acting on their own clocks, then an ecosystem modelled in such a logic is a straightforward representation of an agent's behaviour in the sense of multi-agent systems (MAS). Such a view is the same as in [18]. In what follows, we consider that agent's attributes which depend on processes specified at a certain scale of time will have their state changed only at that level.

If an agent A_i acting at a coarse level of time interacts with other A_j working at lower level, and the result of this interaction is that an attribute Att_j of A_j affects and attribute Att_i of A_i, then A_i has to find out the sequence of values for Att_i. For instance, the model of a tree for the example we gave should find out the sequence of values for the bug's position during the period of time in which its attribute is assumed to be constant.

The MAS approach we are using takes into account that an agent may "read", or observe the value of the attributes of other agents if such information is relevant to its behaviour. Thus each agent may need a process dedicated to obtaining such information. We will call this *observer process* (OP). In our case, the OP will get all the values of Att_j, in a list L, during an specific period of time of the length of L, and then the agent A_i will compute the influence of these values on its attribute Att_i. This idea is depicted in Figure 2.

Note that the OP needs to know the time scale of the agent A_j in order to know how to relate both scales of time. Thus, every agent must specify the scale of time of the internal processes responsible for

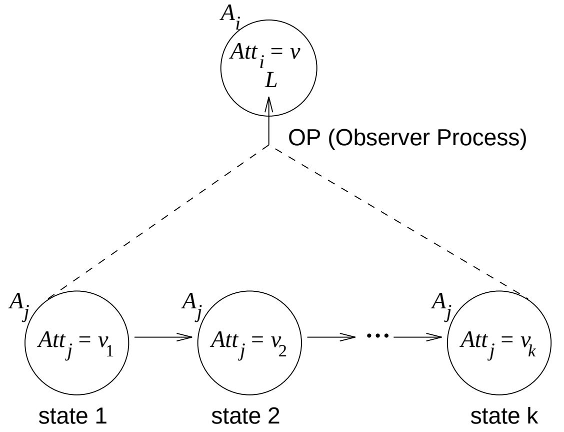

Figure 2: Interaction between agents A_i and A_j, where the OP of A_i get the list L of values of Att_j during a period k units of time.

its behaviour, and also a dependency relation between attribute and process. This will be done by using the predicates $scale(time, Agent, Process, C)$, where C is a MTC, and $depend(Attribute, Process)$.

In this way, we extend the language by introducing a builtin predicate which is specified in the level of the meta-interpreter. This is the predicate $progress/4$ for representing the progress observed of the value of an attribute Att of an agent Obj, from a given temporal entity T during a given period of time P, and the progress will return in a list. A simple specification of the $progress/4$ predicate can be, for example, as follows.

$$
\begin{array}{l}
progress(value(Att, Obj, V)@T, P, [V|R]) :- \\
\quad depend(Att, Proc), \\
\quad scale(time, Obj, Proc, C), \\
\quad solve(value(Att, Obj, V) \ @ \ T, \_), \\
\quad future(T, P, Tf), \\
\quad progression(value(Att, Obj, V) \ @ \ T, Tf, C, R).
\end{array}
$$

$$
\begin{array}{l}
progression(\_ \ @ \ T, Tf, C, [\,]) :- \\
\quad next(T, C, Tf). \\
progression(value(Att, Obj, Vi) \ @ \ Ti, Tf, C, [Vj|R]) :- \\
\quad \neg \ next(Ti, C, Tf), \\
\quad value(Att, Obj, Vj) \ @ \ p(1, C) \ after \ Ti \\
\qquad \Longleftarrow value(Att, Obj, Vi) \ @ \ Ti \ \& \ Constraints, \\
\quad solve(Constraints, \_), \\
\quad next(Ti, C, Tj), \\
\quad progression(value(Att, Obj, Vj) \ @ \ Tj, Tf, C, R).
\end{array}
$$

5 Representing the Tree and Bug Interaction

For the sake of simplicity, we will assume that both eco-agents start their behaviour at the same time. In this way we have the following facts in our KB.

$growth\_rate(tree, 0.5).$
$scale(time, bug, movement, day).$
$scale(time, tree, growing, week).$
$depend(height, growing).$
$value(height, tree, 9)$ @ $t(1,1,1).$
$value(pos, bug, 6)$ @ $t(1,1,1).$

The specification of the *tree*'s growing process can be, for example, as follows.

$value(height(tree), H)$ @ $p(1, week)$ *after* T
$\Longleftarrow$
$value(height, tree, Hi)$ @ T &
$scale(time, bug, movement, Scale)$ &
$progress(value(pos, bug, \_)$ @ $T, p(1, week), Scale, L)$ &
$growth\_rate(tree, GR)$ &
$influence(GR, L, RealGR)$ &
$(H is Hi + RealGR).$

The specification of the *bug*'s movement can as follows.

$value(pos, bug, 6)$ @ $t(1,1,1).$
$value(pos, bug, PB)$ @ $p(1, day)$ *after* T
$\Longleftarrow$
$value(pos, bug, PBi)$ @ T &
$value(height, tree, H)$ @ T &
$new\_pos(PBi, H, PB).$
$new\_pos(Pos1, H, Pos2) :-$
$Pos1 \leq H,$
$Pos2\ is\ Pos1 + 2.$

The $new\_pos/3$ simply changes the bug's position according to its behaviour as specified in the Example. For this specification we have the following results for the simulation of the bug's position.

| : $value(pos, bug, Pos)$ @ T.
>> $value(pos, bug, 6)$ @ $t(1,1,1)$
| : *more.*
>> $value(pos, bug, 8)$ @ $t(2,1,1)$
...
>> $value(pos, bug, 8)$ @ $t(8,1,1)$
| : *more.*

For the tree's growing process we have.

| : $value(height, tree, H)$ @ T.
>> $value(height, tree, 9)$ @ $t(1,1,1)$
| : *more.*
>> $value(height, tree, 9.44)$ @ $t(8,1,1)$
...
>> $value(height, tree, 11.199999999999998)$ @ $t(6,2,1)$
| : *more.*
>> $value(height, tree, 11.639999999999997)$ @ $t(13,2,1)$

This just shows the behaviour of both through the flow of time. For a query about the value of the tree's height at any time we will have the following results, as expected.

| : $value(height, tree, H)$ @ $t(10,1,1)$.
>> $value(height, tree, 9.44)$ @ $t(10,1,1)$
| : $value(height, tree, H)$ @ $t(10,2,1)$.
>> $value(height, tree, 11.199999999999998)$ @ $t(10,2,1)$

Note that queries were for time values in between two synchronous time steps of the tree's growing.

6 Concluding Remarks

In this section we make a brief analysis on the results obtained, the limitations still to be overcome.

Extension of **NatureTime** avoids re-computation of pure temporal expressions already computed by the recursive definition. To our knowledge this is a new technique for controlling the search taking into account a restrict form of clause.

A limitation of the $progress/3$ is that it is assumed that all processes working synchronously rather than asynchronously. This is depicted in Figure 3, where the lines for P_1 and P_2 represent their clock, and each mark is the time point in which the attributes they change are updated.

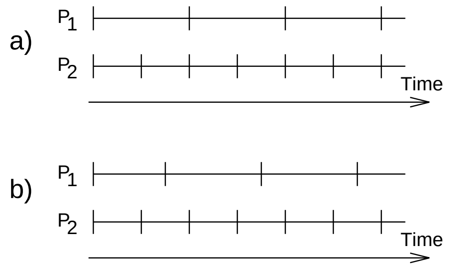

Figure 3: Interacting process P_1 and P_2 where in a) they are synchronous and in b) asynchronous.

For instance, if another agent is introduced at time $t(5,1,1)$ and needs to know the value of the tree every week, the PO for this agent would get the list [9] as the value of the height of the tree. However the correct list should be $[9, 9.44]$ because the new agent was introduced asynchronously in relation to the other agents.

Although the use of the logic for the specification of more complex problems sometimes leads to inefficient local computations, the modularity of computational logic allows us to think in terms of distributing the computation over a set of agents, each one with its own time granularity. We have already done some experiments [2] that strengthen our belief in the value of this type of architecture.

Acknowledgements - We would like to thank very much to Robert Muetzelfeldt for his contribution to our understanding on ecology. We also like to thanks Alan Smaill for his valuable contributions on the time theory. Finally, we thank to the patiency of the referees who carefully read and gave valuable comments for this final version.

References

[1] J. R. Hobbs, "Granularity," in *Proceedings of the of the IJCAI*, pp. 1–4, 1985.

[2] E. Mota, "Time granularity in simulation models within a multi-agent system." DAI Discussion Paper 158, Department of Artificial Intelligence, University of Edinburgh, April 1995.

[3] E. Mota, M. Haggith, A. Smaill, and D. Robertson, "Time granularity in simulation models of ecological systems," in *Workshop on Excutable Temporal Logics- Montreal, Canada*, DAI RP-740, Edinburgh University, 1995.

[4] E. Mota, D. Robertson, and R. Muezelfeldt, "On the granular aspects of time in simulation models," TP-39, Detartment of Artificial Intelligence/University of Edinburgh, 1995.

[5] P. Ladkin, "Primitives units for time specification," in *Proceedings of the AAAI*, pp. 354–359, 1986.

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

[7] P. Ladkin, "Time representation: A taxonomy of interval relations," in *Proceedings of the AAAI*, pp. 360–366, 1986.

[8] B. Leban, D. D. McDonald, and D. R. Foster, "A representation for collections of temporal intervals," in *Proceedings of the AAAI*, pp. 367–371, 1986.

[9] 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.

[10] A. Montanari, "A metric and layered temporal logic for time granularity, syncrony and asyncrony." Unplished work of the First International Conference on Temporal Logic, July 1994.

[11] N. Rescher and A. Urquhart, *Temporal Logic*. Springer Verlag, 1971.

[12] S. Badaloni and M. Berati, "Dealing with time granularity in a temporal planning system," in *First International Conference on Temporal Logic*, (Bonn, Germany), pp. 101–116, Springer-Verlag, July 1994.

[13] D. Cukierman and J. Delgrand, "A language to express time intervals and repetition," in *Proceedings of the of 2nd International Workshop on Temporal Representation and Reasoning*, (Melbourne Beach, Florida - USA), April 1995.

[14] E. Mota, "Temporal representation of ecological domains," DAI TP- 31, Department of Artificial Intelligence, University of Edinburgh, 1994.

[15] M. Haggith, D. Robertson, D. Walker, F. Sinclair, and R. Muetzelfeldt, "TEAK - tools for eliciting agroforestry knowledge," in *British Computer Society Symposium of IT - Enabled Charge in Developing Countries*, 1992.

[16] N. L. Biggs, *Discrete Mathematics*. Oxford Science Publication, 1987.

[17] L. Sterling and E. Shapiro, *The Art of Prolog*. The MIT Press, 1986.

[18] M. Fisher, "Representing and executing agent-based systems," in *ECAI Workshop on Agent Theories,Architectures and Languages (ATAL)*, (Amsterdam, Netherlands), August 1994.

A Properties

Properties of $\stackrel{t}{\succ}$ - In what follows, $\mathcal{C}_i^{mi}$ means the MTC of level i defined with modular value mi. This relation establish a sub-division relationship between MTCs. The $\stackrel{t}{\succ}$ relation has the following properties.

- *transitive* - if $\mathcal{C}_i^{mi}$, $\mathcal{C}_j^{mj}$ and $\mathcal{C}_k^{mk}$ are MTCs and $\mathcal{C}_k^{mk} \stackrel{t}{\succ} \mathcal{C}_j^{mj}$ and $\mathcal{C}_j^{mj} \stackrel{t}{\succ} \mathcal{C}_i^{mi}$, then $\mathcal{C}_k^{mk} \stackrel{t}{\succ} \mathcal{C}_i^{mi}$.
- *reflexive* - if $\mathcal{C}_i^{mi}$ is a MTC then $\mathcal{C}_i^{mi} \stackrel{t}{\succ} \mathcal{C}_i^{mi}$ (every MTC can be subdivided in itself)
- *anti-symetric* - if $\mathcal{C}_i^{mi}$, $\mathcal{C}_j^{mj}$ are MTCs, and $i \neq j$, and $\mathcal{C}_j^{mj} \stackrel{t}{\succ} \mathcal{C}_i^{mi}$, then $\mathcal{C}_i^{mi} \stackrel{t}{\not\succ} \mathcal{C}_j^{mj}$.