Time Granularity in Simulation Models of Ecological Systems

Edjard Mota* ,Mandy Haggith, Alan Smaill, Dave Robertson

The University of Edinburgh,
Department of Artificial Intelligence,
80 South Bridge, Edinburgh EH1 - 1HN, Scotland,
Phone: 44 + 131 + 650- 27{27,07,10,06},
Fax : 44 + 131 + 650-6516,
email: {edjardm,hag,smaill,dr}@aisb.ed.ac.uk

Abstract. Granularity of time is an important issue for the understanding of how actions performed at coarse levels of time interact with others, working at finer levels. However, it has not received much attention from most AI work on temporal logic. In traditional domains of application (e.g. databases, planning, natural language, etc.), we may not need to consider it as a problem, but it becomes important in more complex domains, such as ecological modelling. In this domain, aggregation of processes working at different time granularities is very difficult to do reliably. We have proposed a new time granularity theory based on *modular temporal classes*, and have developed a temporal reasoning system to reason about seasonal cycles. This time theory may be a suitable framework for an executable temporal logic for the specification of ecological models, where each ecological entity is an active agent.

1 Introduction

Temporal reasoning systems usually treat time as a linear sequence of discrete points or linear intervals, and the representation of different levels of time granularity is done by grouping intervals in what are called *convex intervals* [Allen 83], [Allen & Hayes 85], [Ladkin 86], or *non-convex intervals* [Leban *et al.* 86]. In such ontologies of time, cyclical processes are normally represented by using recurrent intervals as in [Koomen 89].

Although many of these theories are formally well developed, very few of them have been used for the specification of simulation models of problems we normally find in nature. The main reason may be the fact that the temporal concepts which people normally use in such domains do not seem to have a direct relation with those of temporal logics. Many phenomena in nature cannot be easily understood in only one scale of time, and using different scales is essential to their comprehension.

* On leave from University of Amazonas, Manaus, Brazil and sponsored by the Coordenation for the Improvement of Academic Staff, CAPES, of the Brazilian Ministry of Education, grant nọ 01723/93-8.

Nowadays, integration of ecological models is an important issue. There are many individual models of parts of systems, and people want to solve problems which require the behaviour of a number of different models to be combined. However, there is little standardisation in the combination process. This is critical when the models were conceived for different levels of time granularity to simulate processes working at different levels of abstraction, but which are somehow related. Meanwhile, some temporal logics for the specification of reactive systems do not deal with time granularity, and so there is a gap between these areas which we aim to bridge. This cross-fertilisation might result in a new way of modelling ecological systems, and raise other types of temporal problems.

This paper shows a temporal logic which deals with time granularity based on an ontology of time called *Linear-Cyclic Hierarchy*. The logic was successfully used in the representation of seasonal cycles in agroforestry problems, and can be used as a programming language to develop simulation models working at different levels of time granularities, particularly for eco-systems. We also address some points about our current research. This concerns the development of an executable temporal logic for the specification of different simulation models of eco-systems to be integrated. We argue that our hierarchy of time is a suitable base for this logic, where each ecological species is treated as an agent (in the sense of multi-agent systems), each with its own scale of time. In this way, the behaviour of these systems may emerge from the interaction between their component agents.

The rest of this paper is organised as follows. In Section 2, we present the motivation for the research by showing an example of a eco-system model, highlighting the problems raised when simulating such domains. We also present a brief review of temporal logics applied to the ecological domain. In Section 3, we present our theory of time granularity. In Section 4, we show how it can be used to specify simulation models in ecology, particularly in the example shown in Sect. 2. We also address its drawbacks and why its direct use does not lead us to feasible solutions for simulating the interaction of many ecological species. In Section 5, we explain how our time theory could be used in a multi-agent systems architecture to provide an executable temporal logic to attack the problem of integrating different simulation models working at different time granularities. Finally, in Section 6, we present some concluding remarks about the current state of our research and its future.

2 Motivation

The objective of this section is to show the complexity of modelling a real eco-system, the agents involved, and the effect of their actions which can be observed and assumed at various time granularities. At the end of the section, we briefly describe related work and why we did not use traditional temporal logics.

2.1 The Biomass Increase of a Forest

Example 1. A piece of forest, from now on f_1, composed of 10 trees, each one allocated in a square with sides of 3 meters, where the shape of the tree is assumed to be unimportant. Each tree has a growth rate r_i, per some unit of time, which varies according to the season and the level of nutrients in the soil. The growing process of one tree may affect the growing of its neighbours because of the competition for nutrients and light, assuming the absorption of light per area of the canopy of the tree. To give an intuitive idea, the situation is depicted in Figure 1. Each t_i, $i = 1, \ldots, 10$ represents a tree.

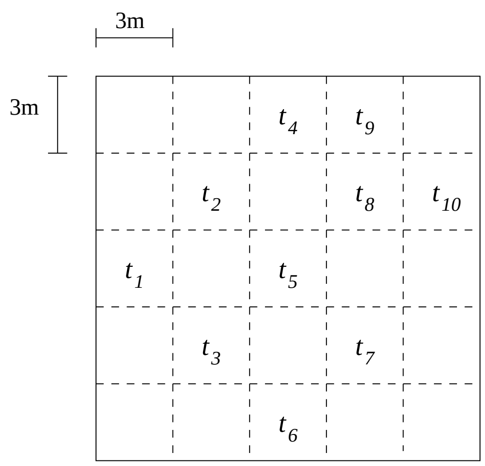

Fig. 1. A simple forest.

The problem of interest in this scenario is to predict the change of height of the trees in a weekly scale of time, and as a consequence their biomass and the biomass of the whole forest. Due the great complexity of the processes involved, what is usually done is to develop a single simulation model, for each process, at an appropriate level of abstraction and time granularity, and after simulating all of them separately the results are joined. But as mentioned in Sect. 1 this practice is very unlikely to give accurate results. For example, the height increase of a tree t_i might be represented by an equation of the form

$$\frac{dH_i}{dt} = r_i H_i \left(1 - \frac{H_i}{MAX_{H_i}}\right) . \tag{1}$$

where MAX_{H_i} is the maximum height that a tree t_i can reach during its whole life, and r_i is the growth rate of t_i, and H_i is the height of t_i. Usually, r_i is assumed to be an average value per some unit of time. It is intended to represent all necessary "information" about the growth of t_i. However, this does not model explicitly the interaction among trees, which may affect this rate. So, a more sophisticated simulation model for this problem should consider as a relevant part of the scenario the following processes:

- the rate of water uptake, based on the features of the soil where the tree is placed. This rate would be at some scale of time. Note that the water in-flow of the soil would also change according to the season of the year. So we may need to represent it as a cyclical process.
- the rate of the absorption of nutrients (or uptake), e.g. carbon (photosynthesis) and nitrogen (roots). Another time scale may be needed for this.
- the influence of the concentration of water, nutrients, etc., in the soil, according to the age of the forest. It will be responsible for stopping the increase of tree biomass. Another time scale may be needed.
- the competition for light, assuming the absorption of light per area of the canopy of the trees, and also its height. Another scale may be needed.
- the action of external events which change the environment
  - natural events such as fire, storms, epidemics of insects, new trees appearing due to natural production of seeds, etc.
  - cutting some trees down, reforesting some areas, etc.

Let's consider the process involving the leaves and the roots of a tree, and make an intuitive analysis of their effect in the growth rate. The effect of photosynthesis changes in a scale of minutes, since the incidence of light changes minute by minute within a day. On the other hand, the effects of water uptake and nitrogen absorption by the roots are more effectively described on a hourly time scale. The processes clearly work at different time granularities and we need to integrate them to properly represent their influence in the growth rate of a tree.

For the purpose of this paper, we will concentrate only on the influence of one tree on the growing process of another tree. The influence on a tree t_i by other trees is approximated as a function of their height and their distance from t_i. This is intended to represent how the acquisition of biomass of other trees affects t_i. Basically, the taller a tree t_j, the more effect it will have on the growth rate of t_i, and the further t_j is from t_i the less effect it will cause. This will be represented by the quantity $k\frac{H_j}{d_{ij}}$, where k is a constant which would normally be determined empirically, and d_{ij} is the distance between t_i and t_j. Equation (1) now becomes

$$\frac{dH_i}{dt} = (\sum_{j=1}^{nb} k\frac{H_j}{d_{ij}})H_i(1 - \frac{H_i}{MAX_{H_i}}) \ . \tag{2}$$

where nb is the number of tree neighbours of t_i.

2.2 Related Work

A direct attempt to model temporal aspects of ecological phenomena in terms of temporal logic, was proposed in [Gray 93], using a linear point structure representation of time. Such an approach does not have any mechanism of inference to reason about simple temporal knowledge, and also nothing about how to solve temporal conflicts on seasonal cycles.

The same problems of reasoning about cyclic events were found when we tried to use other well known logics, such as Gabbay's temporal logic [Gabbay 87]: the temporal logic with *Since* and *Until* and fixed point operators [Gabbay 89], [Barringer *et al.* 91]. Furthermore, these approaches have no commitment to both the "durational" aspects of temporal logic and with different levels of time granularity, but with the specification of possible states of the world in a single time scale.

Another possibility would be Allen's system [Allen 84], [Allen & Hayes 85], where the problem of reasoning about cycles could be implemented using the work on reasoning about recurrences [Koomen 89]. However, these approaches do not clearly explain how to use, in practice, the inference mechanisms they provide to model cyclical events in a "natural" way to obtain appropriate inferences. These logics treat time as a mathematical structure based on natural numbers, or integers, or reals, etc.Almost no attempt has been made to develop a temporal logic theory based on notions of time which we can actually conceive in the natural world, such as day, lunar month, and seasonal cycles, all together. Any temporal logic application, based on whatever mathematical structure, must choose which scale is being regarded. Since we have established that the integration of temporal scales must be an interactive process, it is essential that the notation used to do this can be communicated to the users.

Granularity is very important if we intend to look at the world at different levels of abstraction, and when switching from one level to another it is necessary for the comprehension of the phenomena being observed [Hobbs 85]. This has been applied in the context of temporal databases, in order to obtain efficient maintenance of them [Dean 89], and planning systems, to achieve plan actions at different scales of time and reduce the computational complexity of such systems [Badaloni & Berati 94]. In reactive systems, few temporal logics deal with granularity of time. Recently, [Montanari 94] proposes a many-sorted first order logic augmented with temporal operators and a metric on time to deal with time granularity.

The theory we developed in [Mota 94] 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. In the next section we will present the main rationale of this hierarchical theory of time.

3 NatureTime Logic

In this section we will present our hierarchical theory of time in a very informal way, showing some formalisms only when it becomes necessary. A more formal description can be found in [Mota 94]. To give a better intuition of how such theory of time can be implemented, we will show a simple temporal reasoner interpreter we called **NatureTime**. This is a standard Prolog meta-interpreter with restricted forms of unification on temporal labels.

3.1 Basic Assumptions

Although our logic for reasoning about time and change through time is declarative, it has a procedural (or executable) interpretation because the rules are in the form *future* $\Leftarrow$ *past*. Furthermore, the language used to implement the meta-interpreter was Prolog, which has been the basis for some executable temporal logic definitions due to its procedural interpretation.

In terms of time ontology, instead of simply introspecting on what might be technically possible, and then trying to find examples of such temporal constructs, we did the opposite starting with some domain examples in the form of English text. For instance, about the example showed in Sect. 2.1, one might say "the tree grows faster during the rainy season (say February) than in any other season". By analysing simple statements like this, we first detected the *sorts* of expressions we normally use to refer about inherently cyclical temporal classes, and that are hierarchically related. Afterwards, we attempted to fit them within a many-sorted logic formalism.

This was done by using a different mathematical framework to model temporal classes like "seasons", or cyclical processes. The framework of time we suggested is to model such a *hierarchy of cycles* using modular arithmetic, which is also called clock arithmetic [Biggs 87]. In this view, each cyclical temporal class is defined by one modular set, and so we permit the succession of time in a cyclical way, where the last element is followed by the first. The theory is based on the following assumptions.

1. *temporal entity* (TE) - is a reference to a measure of time, e.g. June, 24th, 1980.
2. TEs are grouped and circularly ordered, forming *modular temporal classes* (MTC), e.g. December and January are TEs of the class "month", and the last element, December, is followed by the first, January.
3. Temporal classes are modularly sub-divided in other temporal classes by what we call *modular value*. For instance, 60 is the modular value which sub-divides hour into minutes.
4. The specification of one modular temporal class defines one level of a time hierarchy. There may exist as many levels of hierarchy as we want. For instance, one could be interested just in days and hours, and so there would be two levels where the second (day) would be defined by the first (hour).
5. The number of levels of a time hierarchy must be finite. The highest can be considered as the larger interval, and the lowest as the smallest.
6. The highest level of the hierarchy is not circularly grouped to form a modular temporal class, but it is linearly ordered in an infinite sequence. Thus, for each instance of the highest temporal class there is one positive integer. In the example of the previous item, day would be a temporal class with its instances linearly ordered, but not those from minutes which would be circularly ordered.

The first allows us to introduce the idea of "previous" and "next" time without inconsistencies, since they will refer to temporal entities at the same level of time granularity. For instance, "next month" would refer to another month, as "previous year" refers to another year.

The second provides a short and elegant mechanism to refer to intermittent intervals. In a *pure* linear structure of time they would be represented by collections of intervals.

The third provides us with a facility to divide a temporal class in the way we need. For instance, if one has defined week as a sequence of seven days, then we can create *labour-week* as another modular temporal class. How these two classes would related to each other and other elements of the hierarchy is beyond the scope of this paper.

The fourth allows us to have multiply nested levels of time granularity. For instance, from the example 1 we could define levels of time from years down to minutes, or from months down to hours.

The fifth makes the time ontology a tractable one, since without this restriction there would be no way to compute the operations over our time expressions. This analysis is beyond the scope of the present paper.

Finally the sixth allows us to have the flow of time. If no level were a linear sequence, then this time framework would loop for ever.

3.2 Logic Definition

In what follows, we will use `this font` to refer to the elements of our language. Our logic is an extension of Prolog, and so its syntax is assumed to be same, but enhanced with some special symbols, terms, and a unification algorithm for temporal labels. We are going to show the extensions introduced at the same time as we present our hierarchy of time theory. Afterwards we will show how the logic is implemented.

The vocabulary is formed by the same symbols for variables, constants, functions and predicates as in Prolog. The set of propositional connectives for conjunction, disjunction, and implication are represented here by `&`, `or`, and $\Longleftarrow$, respectively. A classical formula can be "annotated" 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). When a classical formula is not annotated, then it is assumed to mean that it is true in all states of the world.

Temporal expressions, which we also call *pure temporal expressions*, represent the temporal entities of the hierarchy of time which is composed by *modular temporal classes*, or simply MTC. MTCs are defined by using the special predicate

`mod_temp_class(C1,C2,ModularValue)`,

where this relation says that `C1` is modularly defined by `ModularValue` elements of the class `C2`. The hierarchy of time we can define is always a finite one in relation to the number of levels. More formally, a *Linear-Cyclic Hierarchy* of time is a finite ordered set $\mathcal{T}_h$ of modular temporal classes $\{mc_1, \ldots, mc_n\}$, called the set of *time levels*, where

- $mc_n =$ `mod_temp_class(flow_time,`c_n`, infinity)`, where `flow_time` and `infinity` are special constants.
- $mc_i =$ `mod_temp_class(`c_i, c_{i-1}, m_v`)`, $1 < i < n$, where c_i is a constant symbol called the name of the class mc_i, and c_{i-1} is another constant symbol, $c_{i-1} \neq$ `lowest`, which is a special constant, and m_v is a natural number.
- $mc_1 =$ `mod_temp_class(`c_1`, lowest, smallest)`, and c_1 is one constant symbol, and `smallest` is another special constant.

Example 2. A simple version of the calendar model of time can 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,smallest).
```

This sequence of relation defines the flow of time, represented by `flow_of_time`, as a linear and infinity (`infinity`) sequence of `year`, `year` as a MTC of 12 `month`, `month` as a MTC of 30 `day`[2], `day` as the smallest (`smallest`) interval. Within this hierarchy we have the following recursive definition of sorts of PTEs:

- `t(x1,...,xn)` to represent one single smallest interval of time through the whole hierarchy. For instance, in the hierarchy defined above, the number of levels is n=3, and so `t` is a function with 3 arguments.
- `i(Range,Class)` to represent *cyclical* intervals. For example, `i(11...2,month)` which represents every instance of the interval from November up to February, or the class of all intervals between the November of one year to the February of the next year.
- `P after T` to relate a temporal entity `T` and another temporal entity in the future, in which they are separated by a length of time `P`. `P` is a period of time if it is in the form `p(Value,Class)`, or `p(Value,Class) plus P`, where `plus` is a constructor of composed lengths of time, and `P` is a period. For instance, `p(15,day) after i(2...2,month)` represent "fifteen days after February", and `p(10,day) plus p(3,month) after t(23,3,1994)` represent "3 days and 3 months after March 23, 1994".

The temporal operator we use is the throughout "`@`" operator, which means that a classical logical formula is true throughout the whole interval. It can be defined in terms of the `at` operator[Fruhwirth 94], for instants of time, or intervals with length zero. In this work we prefer to treat such "instants" of time as small intervals that can always be sub-divided, and so the "`@`" is quite enough to refer to them.

3.3 The Enhanced Meta-interpreter for Temporal Reasoning

In this section we will show the specification of the simple meta-interpreter of the **NatureTime** system. We will show the clauses of the extended temporal meta-interpreter, with an explanation for each extension.

A Solver to NatureTime The meta-interpreter, denominated by *temp_solver*, is an extension of the one presented by [Sterling & Shapiro 86]. The extensions are the introduction of the logical connective to represent conjunction and disjunction; as we have temporal Horn clauses, we will change the predicate `clause` for a convenient representation for temporal Horn clauses; and finally, the introduction of a unification predicate to unify the PTEs. Because of the need to control the unification over PTEs, the standard `solve/1` predicate is changed to be a binary relation `solve/2`. The first argument is the temporal formula to be solved, and the second the result of the computation for the first argument be true. The enhanced meta-interpreter is shown below.

```
solve(A, A):-                           % A is true if
   \+ temp_formula(A),                  % A is not a temporal formula
   A.                                   % and is true as a Prolog goal.
solve(A @ T1, A @ T3):-                 % A@T1 is solved to A@T3 if
```

[2] This was done for the sake of simplicity. An extension of this logic is being developed to deal with irregularities such as the real calendar.

```
   \+ A = (_ & _),                       % A is not a composite event
   A @ T2,                               % there is some A@T2 and
   temp_unify(T1, T2, T3).               % T3 is the MGU of T1 and T2.
solve(A @ T,A @ T):-                     % A@T is true if
  X = (A1 & A2),                         % A consists of A1 & A2 events
  solve(A1 @ _,A1 @ T1),                 % A1 is true throughout T1
  solve(A2 @ _,A2 @ T2),                 % A2 is true throughout T2
  temp_unify(T1,T2,T).                   % T is the MGU of T1 and T2.
solve(A & B, A1 & B1):-                  % A & B is solved to A1 & B1
   solve(A, A1),                         % if A is solved to A1, and
   solve(B, B1).                         %    B is solved to B1
solve(A or B, R):-                       % A or B is solved to R
   (solve(A, R) ;                        % if A is solved to R, or
    solve(B, R)).                        % if B is solved to R
solve(A @ T1, A @ T3):-                  % A@T1 is solved to A@T3 if
   A @ T2 <== Precondition,              % A@T2 is the head of a THC
   solve(Precondition, _),               % The body of THC is true
   temp_unify(T1, T2, T3).               % T3 is the MGU of T1 and T2.
```

The first clause simply verifies if the formula is not a WFTF, and solves the goal by calling a standard Prolog goal. The second clause consults the Knowledge Base (KB) to check whether we can directly deduce `A @ T1` or not. The third clause is to deal with composite events, where we want to know if there is some time interval throughout which they can happen all together. Clauses 4 and 5 are just another way to re-write the standard clauses for logical conjunction and disjunction, and so do not need any detailed explanation. The last clause is the clause for logical implication, which gives an alternative for the first two clauses if they fail. To understand their semantics we need to understand the declarative interpretation of each one; they should be interpreted as follows.

2 - `A` *is true throughout* `T3`*, if* `A` *is not a composite event, and* `A` *is true throughout* `T2`*, and it is true that* `T3` *is the temporal unification of* `T1` *and* `T2`.

3 - `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* `T` *is the temporal unification of* `T1` *and* `T2`.

6 - `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`.

Unification Algorithm The specialised unification algorithm we developed to treat PTEs consists of two steps. First, every complex PTE is reduced to a canonical form of temporal unit. Second, canonical forms of temporal unit are unified according to the relations between intervals. Unification is represented by the relation `temp_unify(Term1,Term2,RUTerm)`, where its meaning is `RUTerm` is the temporal unification of `Term1` and `Term2` if

1. `Term1` *and* `Term2` *are equal terms, and* `Term1` *can be reduced to* `RUTerm`.
2. `Term1` *is not equal to* `Term2`*, and* `Term1` *can be reduced to* `RTerm1`*, and* `Term2` *can be reduced to* `RTerm2`*, and* `UTerm` *can be reduced to* `RUTerm`*, and the unification of the units* `RTerm1` *and* `RTerm2` *is* `RUTerm`.

The reduction of one term `reduce(Term,RTerm)` is true if `Term`

1. *is a temporal variable, given by* `time_var`*, and* `RTerm` *is equal to* `Term`. The `time_var` interpretation is a term is a temporal variable if it is a Prolog variable, or a canonical form of time unit where all its elements are Prolog variables.
2. *is a linear interval* `T1...T2`*, and* `T1` *can be reduced to* `RT1`*, and* `T2` *can be reduced to* `RT2`*, and* `RTerm` *is equal to* `RT1...RT2`;
3. *is a cyclical interval* `i(Range,Class)`*, and RTerm equal to* `Term`.
4. *is of the form* `Delta after T`*, and* `T` *can be reduced to* `TExp1`*, and* `RTerm` *can be reduced to* `TExp2`*, and the relation* `future(TExp1,Delta,TExp2) is true`*. The* `future/3` *relation means that the future of* `TExp1` *after the period of time* `Delta` *is* `TExp2`.

For instance,
`period(1,week) after (period(1,week) plus period(3,day) after t(14,2,1994)` is reduced to `t(1,3,1994)`.

Example 3. Suppose we have the following sentences in a knowledge base

```
foo(corn) @ i(5...6,month).
foo(tea) @ i(2...10, month).
foo(coffee) @ i(8...4,month).
foo(rice) @ I <== (foo(tea) & foo(coffee)) @ I.
```

In what follows, the symbols "`:|`" and ">>" represent the query and answer prompts, respectively, of the **NatureTime** system. When a query is done the dialog interface call the `solve/2`, and show only the second argument of it as the answer.

```
:| foo(corn) @ i(6...9,month).
>> foo(corn) @ i(6...6,month)
:| foo(corn) @ i(3...5,month).
>> foo(corn) @ i(5...5,month)
:| more.
>> Sorry, no further solution is possible.
|: foo(rice) @ T.
>> foo(rice)@i(8...10,month)
|: more.
>> foo(rice)@i(2...4,month)
|: more.
A:  Sorry, no further solution is possible!
```

The `more` command is just a special command of **NatureTime** to allow all possible solutions by backtracking (analogous to the ";" command in standard Prolog).

4 Application to Simulation Models

This section shows a direct application of our logic language to develop a simulation model for the growing process of the trees of the example of Sect. 2.1. After this, we point out the drawbacks of this implementation and of the language.

4.1 Simulation Models in NatureTime

In our previous scenario, as the scale of time use was week, we could define another MTC within the hierarchy defined, that is `mod_temp_class(week,day,7)`[3]. The rainy season, assumed to be only in February, can be written as `season(rain) @ interval(2...2,month)`. So, in every instance of February, within a MTC of year it will rain.

The height of a tree can be calculated by using the Equation (1), but we have to make the growth rate r_i a function of the interaction between the tree and its neighbour trees as in Equation (2). To do this, we first define a predicate to represent the neighbours of a tree along with the distance between them. The height must be calculated for each time step. This will be represented by the following predicate definition, where the base case is the initial height for each tree.

```
height(Tree,H) @ p(1,week) after T
  <==
   height(Tree,H1) at T                       &
   max_height(Tree,MAX)                       &
   real_gr(Tree,T...(p(1,week) after T,RGr)   &
   neighbours(Tree,NTrees)                    &
   influences(NTrees,T,R)                     &
   sum(R,TR)                                  &
   (Gr is RGr - TR)                           &
   (C is Gr*H1*(1 - H1/MAX))                  &
   (H is H1 + C).
```

The `influences/3` predicate represents the influence of the neighbours of a tree *since* the time `T`. The `sum/2` predicate represent the relation between a list of values and a number which is the sum of these values. The `real_gr/3` gets the supposed "real" growth rate of the tree throughout the interval of one week. The `real_gr/3` can be defined in a standard Prolog style, for example, as follows

```
real_gr(Tree,T,Gr):-                        % Gr is the growth rate if
   solve(season(rain) @ T,_),               % T is a rainy season
   growth_rate(Tree,LowGr),                 % Lowgr is a standard Gr
   Gr is 1.2*LowGr.                         % Gr = LowGr*1.2
real_gr(Tree,T,Gr):-                        % Gr is the growth rate if
   \+ solve(season(rain) @ T,_),            % it is not true that T is
   growth_rate(Tree,Gr).                    % a rainy season, and get Gr
```

Note that we call the `solve` meta-interpreter to use its facilities of unification as we saw in Sect. 3.3. The complete knowledge base of our example is shown in the Appendix A.

Below, we show the simulation of the growing process. Note that when the simulation "leaves" the season of raining, the growth rate decreases, as expected.

```
|: height(t1,H) @ T.
>> height(t1,1)@ t(14,2,1994)
```

[3] In a further extension of the logic we are going to treat the nested notations over MTC which are not part of the main hierarchy of time.

```
|: more.
>> height(t1,1.0097385444743936)@ t(21,2,1994)
|: more.
>> height(t1,1.019549379017451)@ t(28,2,1994)
|: more.
>> height(t1,1.0294326000485676)@ t(5,3,1994)
|: more.
>> height(t1,1.0376322103100148)@ t(12,3,1994)
|: more.
>> height(t1,1.0458800768343346)@ t(19,3,1994)
|: more.
>> height(t1,1.054176211878584)@ t(26,3,1994)
|: more.
>> height(t1,1.0625206230928805)@ t(3,4,1994)
```

4.2 Drawbacks of the NatureTime System

These include

- The limitation of the meta-interpreter which does not allow an explicit specification of temporal aspects of certain processes, e.g. the growth rate.
- The unification over the `after` operator causes a serious problem of time consumption. The reason is because recursive specifications on time steps produce big pure temporal expressions to be reduced. To be more precise, in the kernel of our unification algorithm, every time an expression of the form `P after T` is found, `T` is *reduced* to a canonical form, but it can also be another `P' after T'`, which is not canonical, and the reduction is applied, and so on.
  The problem with this is not the reduction algorithm itself, but the redundant recomputations of such pure temporal expressions. This could be solved as discussed in [Robertson *et al.* 91], if we had used some technique of recording the results of satisfying particular goals.
- The representation of different agents, e.g. forest, working at different time granularity would cause an exponential explosion in the search space of the deduction process. For instance, to produce information for forest in order to compute its influence in low levels, it would be necessary to regenerate this information.

All of these problems are much more related to the sequential computation of the logic-language used than to our theory of time granularity itself. Another way to attack the problem may help us not only to obtain efficient specifications, but also to achieve better integration of simulation models. We believe that our *Linear-Cyclic Hierarchy* of time is suitable for this new approach which we will briefly point out in the next section.

5 Ecological Simulation in Multi-Agent Systems: Some Insights

From the representational point of view, simulation models in ecology, developed in programming languages where the computation is sequential, do not feel particularly natural, mainly because, in such kinds of systems, many processes happen concurrently in time, and ecological entities have their own behaviour (in a particular time scale), constantly reacting to the influence of their environment[Levin 92]. In this way, we propose to consider an eco-system as a very complex reactive system, in the sense of [Pnueli 86], since the behaviour of ecological entities are represented by simulation models. In this Section we address some insights about the *ecological agent* view of eco-systems, and we also suggest our time granularity theory as a basis for an executable temporal logic for simulation models in ecology.

5.1 From Logic-based to Agent-based Simulation

In the logic-based simulation of the Sect. 4, the clauses for the growth rate and maximum height of a tree can be seen as "individual knowledge" of each tree, and the clause for height as *common hierarchical knowledge* (CHK) for all species of tree, since the growing process of all trees are assumed to be same. In this way we may consider the previous set of clauses as a collection of agents. This view, similar to that of agent as part of expert systems as described in [Dieng 91], we say is a *weak agent definition*.

The clause for forest biomass can been seen as the knowledge from all agents, or the forest if we consider it as an agent as well. Our knowledge base would consist of a set of sets of clauses, each set for one agent, in a hierarchical way. For instance, in the previous example the biomass of a forest depends on the biomass of the trees, and the biomass of one tree at a given time depends on the its specific weight and its height, and its height depends on the influence of its neighbour trees at previous time. However, nothing in the language of specification is related with the view of the world as consisting of active agents.

The paradigm of Multi-Agent Systems (MAS) seems to fit quite well as a framework of modelling ecological systems. By using MAS, the behaviour of complex eco-systems is expected to emerge from the interactions between agents. As a consequence, integrations with higher accuracies can be done, and improved comprehension of how processes working at different levels of time granularity can be achieved. In this way, we propose to treat each ecological entity as an *ecological agent*, with its *individual knowledge* (internal and external) to represent its own state descriptions. We will need to specify the external description of an agent. This means which information about one agent is worth making available to other agents of its environment. We will also need to represent CHK about the relation of its processes with other species and the environment (some kind of "social reasoning").

The idea now is that instead of using a linear point structure of time, we can use the Linear-Cyclic structure of time, as defined in Sect. 3, because:

- it allows us to easily define as many levels of time granularity as needed. Furthermore, each agent could have its on clock, and we need to synchronise the clocks of different agents when integrating, or creating macro-agents by composing small ones.
- it allows us to define temporal cycles, also important in such simulation models.

By using this theory, we can assume that the time stamp `T` of the specification above can be in one of the time units [4], e.g. `year(3)` for the third year. In this way, assuming a time hierarchy with day, month, and year, the value of the tree's height for

[4] In the current extension of our time ontology we are introducing mechanisms to represent entities of all levels without using the representation $t(t_1, \ldots, t_k)$.

a given time could be "asked", for instance, `value(height,tree(t1),H) at month(3)`, which means "the value of the height of the tree t1 at the 3rd month of life is H".

To allow the definition of the time scale of the processes of an agent, the CHK of a group of agents related to a process, should contain the following predicate

`scale(time,tree,Process,MClass)`,

where `Process` is to say which process of the tree works in the level of time defined by the modular temporal class `MClass`. We also need to say which attribute of an agent is changed by a process. This can be defined by another predicate which is `change(Attribute,Process)`. As an example, the definition of a tree as an agent could be

```
Agent     : tree
Attributes: height, growth_rate, area_of_leaves,
            length_of_roots, spatial_position, specific_weight
Process   : growing, water_up_take, nitrogen_up_take, photosynthesis
Time Scale: scale(time,tree,growing,week).
            scale(time,tree,water_up_take,hour).
            scale(time,tree,nitrogen_up_take,hour).
            scale(time,tree,photosynthesis,minute).
Relation between processes and attributes
            change(height,growing).
            change(growth_rate,water_up_take).
            change(growth_rate,nitrogen_up_take).
            change(growth_rate,photosynthesis).
            ...
```

Based on a such high level of description, the time theory we provide may be considered with no commitment to the computational features of the model. However, due to the inefficiency of computation of systems like **NatureTime**, the theory may not be useful or provide mechanisms to allow complex models to be implemented. Furthermore, the concurrent aspect of eco-systems could be better understood if we had a language to specify the behaviour of ecological agents, i.e. the processes related with such agents. These processes may work at different scales of time and also in parallel. For instance, water and nitrogen up take, and photosynthesis may happen at the same time, with their effect noted in different time scales.

The efficiency of distributed systems as used in MAS is what we intend to use, and explore how to fit our time granularity theory, where each agent "acts" according to its own time scale. Now the problem is how to provide the logic with mechanisms which deal with different time granularities.

5.2 What Would be Needed?

To our knowledge, temporal logics for reactive systems developed so far do not worry much about processes working at different levels of time granularity. Our suggestion is that the use of an executable temporal logic, to specify simulation models in ecology, could help to make the points of contact more obvious, and the problem of integration more tractable. However, to our knowledge, the state of art in programming language for MAS is much more concerned with the tense aspects of agent's properties and behaviour through time than with the "durational" aspects of these features, e.g.

[Fisher 94]. One approach which seems to worry about it is AGENT0 [Shoham 90], but this does not deal explicitly with time granularity.

An executable temporal logic for the specification of simulation models for eco-systems, in the sense of MetateM [Fisher 94] but with time-stamped temporal operators, should have the following features.

- the temporal operators should be consistent in all levels of time granularity.
- the processes of transition, in any level, from one state to another should be represented by interaction with other processes occurring in other levels as well. For instance, if we want to introduce soil as an "agent" to control the resource of water, then it should be done by message passing from a tree to the soil.

In this way, each process related with the behaviour of an agent could be separately specified at different time scale. This means that for each agent there exists a set of "reactive systems", working and interacting concurrently in time, and at different levels of time granularity. We have implemented a very specialised MAS for the *Example 1*, by using Prolog as specification language, and the *tuple space* of Linda [Carriero & Gelernter 89] to represent the external description of agents and also as medium for passing of message. However, this does not completely correspond to what we really want to do, and to explore our time granularity theory within this architecture is our next step.

6 Concluding Remarks

In this paper we presented a new theory of time granularity which can be easily understood and used to define as many levels of time hierarchy as needed. We also showed that such a theory is useful in the representation of cyclical processes in simulation models for eco-systems.

Although this theory has been implemented in a very simple temporal reasoning system, we suggested that it can be used as a basis to develop a time-stamped executable temporal logic for the specification of simulation models, by involving agents working at different scales of time. This language should be based on a non-sequential computation in order to better represent agents interacting and concurrently working.

By using an executable temporal logic with such features, a modeller could specify his/her simulation model as an agent, with its own knowledge and scale of time. In this way, complex simulations would be expected to result from the interaction among different ecological agents.

Acknowledgements - The first author would like to thank Wamberto Wasconcelos, for helpful discussions on the time theory during our minutes for "academic coffee". We would also like to thank Robert Muetzelfeldt and Paulo Salles for many discussions on the stuff of ecological simulation.

References

[Allen & Hayes 85] James F. Allen and P. J. Hayes. A common sense theory of time. In *Proceedings of the of the IJCAI*, pages 528–531, 1985.

[Allen 83] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.

[Allen 84] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(1), 1984.

[Badaloni & Berati 94] Silvana Badaloni and Marina Berati. Dealing with time granularity in a temporal planning system. In *First International Conference on Temporal Logic*, pages 101–116, Bonn, Germany, July 1994. Springer-Verlag.

[Barringer *et al.* 91] Howard Barringer, Michael Fisher, Dov Gabbay, and Anthony Hunter. Meta-reasoning in executable temporal logic. In *Knowldege Representation*, 1991.

[Biggs 87] Norman L. Biggs. *Discrete Mathematics.* Oxford Science Publication, 1987.

[Carriero & Gelernter 89] Nicholas Carriero and David Gelernter. Linda in context. *Communications of ACM*, 32(4), 1989.

[Dean 89] Thomas Dean. Using temoral hierarchies to efficietly maintain large temporal databases. *Journal of the ACM*, 36(4):687–718, 1989.

[Dieng 91] Rose Dieng. Relations linking cooperating agents. In Yves Demazeu and Jean-Pierre Muller, editors, *DECENTRALIZED A. I. - 2*, pages 51–69. Elsevier Science Publishers, 1991.

[Fisher 94] M. Fisher. A survey of concurrent metatem - the language and its applications. In *First International Conference on Temporal Logic(ICTL)*, Bonn, Germany, July 1994.

[Fruhwirth 94] Thom Fruhwirth. Annotated constraint logic programming applied to temporal reasoning. Technical Report 22, ECRC, July 1994.

[Gabbay 87] Dov Gabbay. *Modal and Temporal Logic Programming. In Temporal Logics and their Applications - Antony Galton Editor.* ACADEMIC PRESS, 1987.

[Gabbay 89] Dov Gabbay. The declarative past and imperative future. In H. Barringer, editor, *Coloquim on Temporal Logics and Specification*, pages 409–448. Springer Verlag, 1989. V(398).

[Gray 93] Babak Gray. T-prolog - a temporal logic for an ecological domain. Unpublished M.Sc. thesis, Department of Artificial Intelligence/University of Edinburgh, 1993.

[Hobbs 85] Jerry R. Hobbs. Granularity. In *Proceedings of the of the IJCAI*, pages 1–4, 1985.

[Koomen 89] Johaness A.G.M. Koomen. Reasoning about recurrence. Technical Report Technical Report 307, Department of Computer Science - University of Rochester, Rochester, USA, 1989.

[Ladkin 86] Peter Ladkin. Primitives units for time specification. In *Proceedings of the AAAI*, pages 354–359, 1986.

[Leban *et al.* 86] Bruce Leban, David D. McDonald, and David R. Foster. A representation for collections of temporal intervals. In *Proceedings of the AAAI*, pages 367–371, 1986.

[Levin 92] Simon A. Levin. The problem of pattern and scale in ecology. *Ecology*, 73(6):1943–1967, 1992.

[Montanari 94] Angelo 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.

[Mota 94] Edjard Mota. Temporal representation of ecological domain. DAI Technical paper 31, Detartment of Artificial Intelligence/University of Edinburgh, 1994.

[Pnueli 86] Amir Pnueli. Specification and development of reactive systems. *Information Processing*, 1(1), 1986.

[Robertson *et al.* 91] David Robertson, Alan Bundy, Robert Muetzefeldt, Mandy Haggith, and Michael Uschold. *Eco-Logic Logic-Based Approaches to Ecological Modelling.* The MIT Press, 1991.

[Shoham 90] Yoav Shoham. Agent-oriented programming. Technical Report STAN-CS-90-1335, Stanford University - Department of Computer Science, Standford, California, 1990.

[Sterling & Shapiro 86] Leon Sterling and Ehud Shapiro. *The Art of Prolog.* The MIT Press, 1986.

A KB Definition for the Tree's Height

```
% -------------------------------------------------------------------
%                     NEIGHBOURS DEFINITION
neighbours(t1,[(t2,4.24),(t3,4.24),(t5,6)]).
neighbours(t2,[(t1,4.24),(t3,6),(t4,4.24),(t5,4.24)]).
neighbours(t3,[(t1,4.24),(t2,6),(t5,4.24),(t6,4.24),(t7,6)]).
neighbours(t4,[(t2,4.24),(t5,6),(t8,4.24),(t9,3)]).
neighbours(t5,[(t1,6),(t2,4.24),(t3,4.24),(t4,6),(t6,6),(t7,4.24),(t8,4.24)]).
neighbours(t6,[(t3,4.24),(t5,6),(t7,4.24)]).
neighbours(t7,[(t3,6),(t5,4.24),(t6,4.24),(t8,6)]).
neighbours(t8,[(t2,6),(t4,4.24),(t5,4.24),(t7,6),(t9,3),(t10,3)]).
neighbours(t9,[(t4,3),(t8,3),(t10,2.24)]).
neighbours(t10,[(t8,3),(t9,4.24)]).
% -------------------------------------------------------------------
%                   INITIAL HEIGHT OF EACH TREE
height(t1,1) @ t(14,2,1994).                 % initial height of t1
height(t2,1) @ t(14,2,1994).                 % initial height of t2
height(t3,1) @ t(14,2,1994).                 % initial height of t3
height(t4,1) @ t(14,2,1994).                 % initial height of t4
height(t5,1) @ t(14,2,1994).                 % initial height of t5
height(t6,1) @ t(14,2,1994).                 % initial height of t6
height(t7,1) @ t(14,2,1994).                 % initial height of t7
height(t8,1) @ t(14,2,1994).                 % initial height of t8
height(t9,1) @ t(14,2,1994).                 % initial height of t9
height(t10,1) @ t(14,2,1994).                % initial height of t10
% -------------------------------------------------------------------
%                   MAXIMUM HEIGHT OF EACH TREE
max_height(t1,7).                             % max height of t1
max_height(t2,10).                            % max height of t2
max_height(t3,12).                            % max height of t3
```

```
max_height(t4,5).                              % max height of t4
max_height(t5,5).                              % max height of t5
max_height(t6,7).                              % max height of t6
max_height(t7,12).                             % max height of t7
max_height(t8,6).                              % max height of t8
max_height(t9,4).                              % max height of t9
max_height(t10,13).                            % max height of t10
% ------------------------------------------------------------------
%           STANDARD GROWTH RATE OF OF EACH TREE
growth_rate(t1,0.01).                          % rate growth of t1
growth_rate(t2,0.012).                         % rate growth of t2
growth_rate(t3,0.015).                         % rate growth of t3
growth_rate(t4,0.0009).                        % rate growth of t4
growth_rate(t5,0.008).                         % rate growth of t5
growth_rate(t6,0.011).                         % rate growth of t6
growth_rate(t7,0.015).                         % rate growth of t7
growth_rate(t8,0.01).                          % rate growth of t8
growth_rate(t9,0.006).                         % rate growth of t9
growth_rate(t10,0.018).                        % rate growth of t10
% ------------------------------------------------------------------
%           INFLUENCE OF OTHER TREES IN ONE TREE
influences([],_,[]).                           % 0 tree has no influence
influences([Y|T],Time,[R1|TR]):-               % the influence of each
  ind_influence(Y,Time,R1),                    % individual tree and
  influences(T,Time,TR).                       % the influence of rest

ind_influence((Tree,D),T,I):-                  % Tree's influence is I
  solve(height(Tree,H) @ T,_),                 % H is Tree's height @ T
  (I is H/(D*1000)), !.                        % I = H/(D*1000)).
```

This article was processed using the LaTeX macro package with LLNCS style