J. Symbolic Computation (1996) **22**, 665–698

NatureTime: Temporal Granularity in Simulation of Ecosystems†

EDJARD MOTA‡, DAVID ROBERTSON‡ AND ALAN SMAILL‡

Department of Artificial Intelligence, The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, Scotland

(Received 31 October 1995)

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 simpler domains of application we may not need to consider it 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 (and sometimes cyclically) is very difficult to achieve reliably. We have proposed a new time granularity theory based on *modular temporal classes*, and have developed a temporal reasoning system to specify cyclical processes of simulation models in ecology at many levels of time.

1. Introduction

Temporal logics and reasoning systems usually treat time as a linear sequence of discrete points or linear intervals. Such abstract views of time have been used in the development of many specification languages for real-time systems, databases, planning, etc. One might expect that temporal logics would be a natural way to represent conventional simulation models, since these also represent change over time. However, when we attempt to do this we almost immediately encounter obstacles: different parts of the model may operate at different temporal granularities; processes may operate cyclically; the axioms familiar to temporal logicians may be far removed in programming style from those of simulationists. We encountered all these obstacles when applying temporal logics to ecological modelling. Moreover, many phenomena in nature cannot be easily understood in only one scale of time, and using different scales is essential to their comprehension.

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

† This work and the first author (on leave from Department of Computer Science, University of Amazonas-Brazil) are sponsored by the Brazilian Ministry of Education, grant no. 01723/93-8/CAPES.

‡ E-mail: {edjardm, dr, smaill}@aisb.ed.ac.uk

0747–7171/96/110665 + 34 $25.00/0

standardization 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.

In this paper we present a temporal logical reasoning framework to deal with time granularity based on an ontology of time called a *Linear-Cyclic Hierarchy*. Granularity of time is defined by means of a hierarchy of modular sets and an easy mechanism for specifying processes working at different time scales and which can also be cyclical is provided. The logic was used in the representation of seasonal cycles in agroforestry problems, and we show how it can be used as a programming language to develop simulation models working at different levels of time granularities, particularly for ecosystems.

2. Motivation

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 (Hobbs, 1985). In particular we are concerned with the representation of processes, cyclical or not, working at different time scales. A logical framework for representing and reasoning about this kind of knowledge should deal with the problems of:

(i) Definition of propositions at appropriate levels of temporal granularity.
(ii) Relationships between propositions defined over different time granularities.
(iii) Alignment of temporal domains, allowing events at different temporal granularities to be synchronized.
(iv) Dealing with the "next" temporal operator. For instance, "next time I will play football" may have different interpretations depending on the level of time we are talking about.

Along with these problems we address the need for dealing with cyclical events or processes, at different levels of time and which may interact. The following example shows how such a need arises in an ecosystem model, the agents involved, and the effect of their (possibly cyclical) actions.

> *Example 1*: A piece of forest is composed of 10 trees, each one allocated in a square with sides of 3 m, 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 growth of its neighbours because of the competition for nutrients and light, assuming constant absorption of light per unit area of the canopy of the tree.

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. A sophisticated simulation model for this problem should consider as a relevant part of the scenario the following processes.

(i) 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.

(ii) The rate of the absorption of nutrients (or uptake), e.g. carbon (photosynthesis) and nitrogen (roots). Another time scale may be needed for this.
(iii) The influence of the concentration of water, nutrients, etc, in the soil, according to the age of the forest. This will be responsible for stopping the increase of tree biomass. Another time scale may be needed.
(iv) 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.
(v) 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 us consider the process involving the leaves and the roots of a tree, and make an intuitive analysis of their effect on 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 represent their influence in the growth rate of a tree properly. Due to the great complexity of the processes involved, what is usually done, in practice, is either to keep individual models separately (maybe some parameter values or data sets are shared on an ad hoc basis), or one very large imperative model is built. In the latter case, the control structure of all models must be adapted.

The work presented in this paper proposes a temporal logical reasoning framework for problems of this nature.

3. NatureTime Logic Definition

In this section we will present the hierarchical theory of time originally proposed in Mota (1994), which 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. The basic idea of our logic is to separate the task of defining relations within the model from the task of computing when these relations hold. We view temporal reasoning as a problem of unification between temporal labels, where the flow of control of program execution is influenced by this. Although this can be seen as similar to unification between sorts in order sorted logics, the presentation of our time theory is not directly based on that approach. We leave this for future work, and any use of the term *sort* in this work does not assume any underlying sorted logic theory.

3.1. BASIC ASSUMPTIONS

Our main concern is to provide a logical framework for which the translation of temporal knowledge from a complex domain (in our case ecological modelling), is not a too painful process. Because the different levels of time are usually expressed by using a label for each one, such references about scales should be part of the model. For this reason, we decided not to "disturb" accepted ways of representing simulation models using computational logic, e.g. Robertson *et al.* (1991). That is, it would be ideal if the temporal

aspects of the program could be thought as labels for components of Prolog-like clause programs, so that we could distinguish the task of defining relations within the model from the task of saying when these relations hold. By taking domain examples in the form of English text (Sinclair *et al.*, 1993; Haggith *et al.*, 1992), we classified the *sorts* of expressions that are normally used to refer about inherently cyclical temporal classes, and that are hierarchically related. For instance,

(i) "the tree grows faster during the rainy season (say February) than in any other season" (from the example in Section 2)
(ii) "The effect of photosynthesis changes on a scale of minutes. On the other hand, the effects of water up take and nitrogen absorption by the roots are more effectively described on a hourly time scale. Both processes influence the growth rate of a tree".

We can see from the first example that it is necessary to talk about things which happen intermittently through the flow of time. From the second example we have processes which can also change at different temporal scales. The main issue in these examples is to merge specifications at different time scales coherently in such a way that the responses of lower levels can influence the responses of higher levels. There may exist some cases in which the opposite effect occurs. For instance, some ecological models consider a forest as "a big leaf", and the behaviour of the forest may affect the behaviour of each individual tree over very long periods of time. Another example from a very different field is inflation, in economics. It is a property of the overall economy, but which affects individuals in different ways. Thus, what is needed is a framework of time which allows us to define cyclical relations and relations at many levels of time granularity.

As an initial step we took actual phenomena in nature which led us to use references of time as in these simple examples. This is depicted in Figure 1, where we view the granularity of time as being a hierarchy of modular sets which are related by a kind of "inclusion relation". At all levels, except one (possibly the highest being considered), time is closed, i.e. time moments are isomorphic to points on a circle (Poidevin and MacBeath, 1995). According to this view of time we can understand, for instance, years as being modular cycles of months, [lunar] months of days, and so on.

The basic mathematical framework to model concepts like "seasons", or cyclical processes and such a *hierarchy of cycles* is modular arithmetic [also called clock arithmetic (Biggs, 1987)]. In this way, 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. This allows us to define the "previous" and "next" operators without ambiguity, 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.
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 values*. For instance, 60 is the modular value which sub-divides an

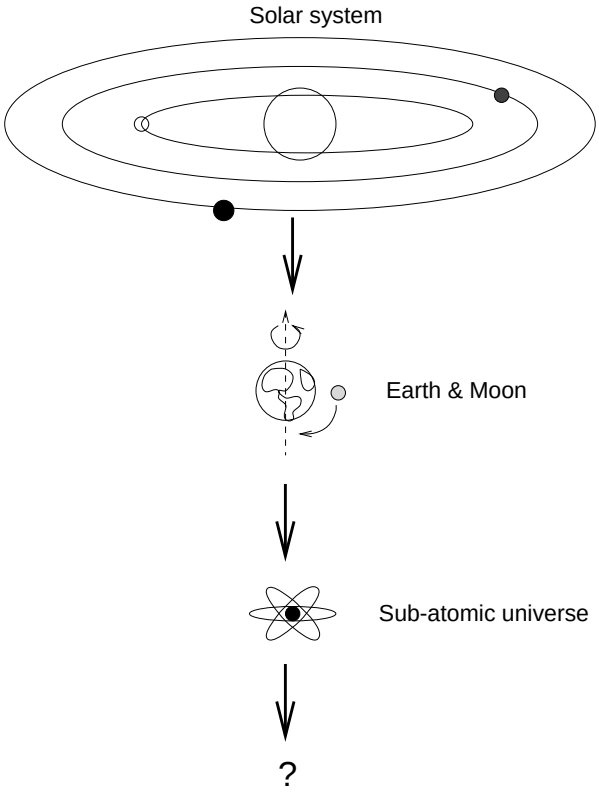

Figure 1. View of the natural events we use to refer about time. Years are cycles of months, months of days, and so on.

hour into minutes. This provides us with a facility to subdivide temporal classes in the way we need.

4. The specification of one MTC 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). This allows us to have multiply nested levels of time granularity.
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. This is to make the theory a tractable one, because without this restriction there would be no way to compute the operations over our time expressions
6. The highest level of the hierarchy is not circularly grouped to form a MTC, 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 for minutes which would be circularly ordered. This is to prevent the flow of time being an eternal cycle.

3.2. ELEMENTS OF THE LANGUAGE

We now present a logic programming language enhanced with some special symbols, terms, and a unification algorithm for temporal labels. In this work we also allow negation by failure under the closed world assumption. So, the syntax is basically Prolog-like but with some special features according to the following definition.

3.2.1. VOCABULARY

(i) Alongside Prolog variables, represented here by $\mathcal{L}_v$, there is a disjoint set $\mathcal{L}_{tv}$ of temporal variables, where x_i, y_i, z_i are variables of $\mathcal{L}_v$, s_i, t_i, u_i are variables of $\mathcal{L}_{tv}$.
(ii) a finite set $\mathcal{L}_c$ of constants.
(iii) a finite set $\mathcal{L}_f$ of non-temporal function symbols of the form f^n, where n is the arity of the function.
(iv) a finite set $\mathcal{L}_{tc}$ of special constants defined as $\{\mathit{lowest}, \mathit{flowtime}, \mathit{infinity}, \mathit{smallest}\} \cup \mathcal{T}_C$, where $\mathcal{T}_C = \{c_1, \ldots, c_n\}$, and each c_i is a special constant or names of temporal classes.
(v) a finite set of temporal function symbols $\mathcal{L}_{tf} = \{p^2, i^2, t^n \ldots^2, \mathit{plus}^2, \mathit{after}^2, \mathit{before}^2, \mathit{of}^2\}$, where $\ldots^2$, plus^2, after^2, before^2, of^2 are all of arity 2 but written using infix notation. For instance, p, *plus*, i, t, *after*, *of*, and *last* in $p(2, \mathit{hour})\ \mathit{plus}\ p(37, \mathit{minute})$, $i(2 \ldots 3, \mathit{month})$, $t(17, 6, 1994)$, $p(1, \mathit{day})\ \mathit{after}\ t(12, 3, 1995)$, $p(13, \mathit{month})\ \mathit{plus}\ p(25, \mathit{day})\ \mathit{before}\ t(25, 12, 1994)$, $\mathit{day}(1)\ \mathit{of}\ \mathit{week}$, $\mathit{last}(\mathit{day}(2)\ \mathit{of}\ \mathit{week}\ \mathit{of}\ \mathit{month}(2)\ \mathit{of}\ \mathit{year}(1996))$ are function symbols. The meta-term f_t^n will be used to refer to temporal function symbols with arity n.
(vi) a finite set $\mathcal{L}_p$ of predicate symbols p^n, where $n > 0$ is the arity of p, and $\{\mathit{subclasses}^2, \mathit{on}^2, \mathit{mod\_temp\_class}^3, \mathit{mod\_value}^2, \mathit{mod\_value}^3, \mathit{change\_mod\_value}^3\}$ is a special subset of $\mathcal{L}_p$.
(vii) the set $\mathbb{Z}$ of integers is also part of the vocabulary.
(viii) the propositional connectives for negation, conjunction, disjunction, and reverse implication are represented here by $\neg$, $\&$, $\vee$, and $\Longleftarrow$, respectively. The truth value for true is represented by $\top$, and false by $\bot$.
(ix) a temporal connective @.

3.2.2. 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.

(i) a *logical term* is defined as usual. Examples of logical terms are *maize*, *grass*, $\mathit{height}(\mathit{tree}(t_1))$, $\mathit{biomass}(\mathit{forest}(f_1))$.
(ii) a *temporal term* (TT) is

- a temporal variable $s \in \mathcal{L}_{tv}$.
- a temporal constant $c_t \in \mathcal{T}_C$.
- a temporal function symbol $f_t \in \mathcal{L}_{ft}$ in one of the following forms.

  * a *period*, which is recursively defined as

    · a *single period* $p(s, m)$ where $s \in \mathbb{Z}^+$ and $m \in \mathcal{T}_C$.

  - · a *composite period* P *plus* P', where P is *single period*, and P' is a *period* term at a higher scale than P.

* a *cyclical interval* $i(s \ldots t, m)$, 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 corresponds to exactly one element of $\mathcal{T}_C$. This can also be seen as a moment of time.
* a *linear interval* $s_1 \ldots s_2$, 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 is some $\alpha \in T_C$. For instance, the function symbol of *week*(1) corresponds to *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*.

A *ground temporal term* is a TT with no variables, e.g. $i(2 \ldots 2, month)$ is a GTT while $i(2 \ldots x, month)$ is not. Examples of TT are $p(3, month)$ *plus* $p(2, day)$, $i(10 \ldots 3, month)$, $t(15, 2, 1994), \ldots, t(3, 4, 1994)$, *minute*(43), *day*(5), *week*(3), *year*(12), *minute*(43) *of day*, *minute*(43) *of day*(3) *of week*, *day*(1) *of month of year*(1994). Note, that *collection interval* is more general than *cyclical interval* in the sense that it may define cyclical and non-cyclical intervals. For instance, *day*(2) *of week of month* is intended to represent all Mondays of all months, and so is cyclical. However, *day*(2) *of week of month*(2) *of year*(1996) is not cyclical because it represents the finite sequence of days 5th, 12th, 19th, and 26th of February 1996.

(i) a *first* and *last* TT of a *collection interval* are represented by *first*(X) and *last*(X), respectively, where X is a *collection interval*. For instance, suppose X is *day*(2) *of week of month*(2) *of year*(1996)), then *first*(X) corresponds to exactly the temporal entity $t(5, 2, 1996)$, while *last*(X) corresponds to $t(26, 2, 1996)$.

(ii) a pure temporal expression (PTE)

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

Examples of PTE are $p(3, month)$ *after* $i(10 \ldots 11, month)$, $p(24, year)$ *after* $t(17, 7, 1970)$, *hour*(4) *of day*(1) *of week*(1) *of month*(3). We say that one of the TT are *canonical forms* of PTE.

(iii) an atomic formula (AF) is

- $\top$ and $\bot$ are atomic formulae
- if $x_1, \ldots, x_n$ are logical terms and $p^n \in \mathcal{L}_p$, then $p(x_1, \ldots, x_n)$ is an atomic formula
- if A is an AF, so is $\neg A$

(iv) classical atomic formulae can be annotated with a PTE by using the temporal operators "@", which means that a classical logical formula is true throughout the whole interval, i.e.

- A, if A is an AF, then it is an atomic temporal formula (ATF).
- $A \,@\, T$, where A is an AF and T is a PTE, is an ATF

Some examples of ATFs are,

- *time_between*$(i(2\ldots4, month), p(3, month))$ which means that *the period of time between February and April, including both, is always equal to* 3 *months*
- *harvested*$(maize, highlands) \,@\, i(12\ldots1, month)$ which means that *maize is harvested in the high lands throughout the whole interval from December up to January.*

(v) *body* is in one of the forms $A \,\&\, B$, $A \vee B$, or C, where A and B are bodies and C is an ATF. A typical example of a body is *time_between*$(i(2\ldots4, month), p(3, month)) \,\&$ *harvested*$(maize, high\_lands) \,@\, i(12\ldots1, month)$.
When representing different propositions A and B which are true at the same time interval T, it will be required to write $(A \,\&\, B) \,@\, T$ rather than $A \,@\, T \,\&\, B \,@\, T$.

(vi) a well formed temporal formula (WFTF) is

- if A is a positive ATF (non-negated ATF), then A is a WFTF with an *empty body*
- if A is a positive ATF and $B \neq \top, \bot$, and B is a body then
  $A \Longleftarrow B$ is a WFTF, and A is called the *head*

Note that a WFTF where its *head* is an AF, and *body* is formed only by atomic formulae corresponds exactly to a Prolog clause with no temporal contents.
An example of a WFTF is

harvested$(tomatoes) \,@\, p(6, month)$ *after* $T \Longleftarrow$ *planted*$(tomatoes) \,@\, T$.

Which means *if tomatoes are planted throughout an interval* T*, then they are harvested* 6 *months after* T.

3.3. THE LINEAR-CYCLIC HIERARCHY STRUCTURE

The temporal structure of time, called *Linear-Cyclic Hierarchy*, is a 4-*tuple* $\mathcal{LC}_H =_{def} \langle \mathcal{E}, \prec, \mathcal{T}_h, \overset{t}{\succ} \rangle$, where $\mathcal{E}$ is a non-empty set of temporal entities (or units of time), $\prec$ is a binary relation of precedence over the set of moments of time in $\mathcal{E}$, $\mathcal{T}_h$ is a finite set $\{c_1, \ldots, c_n\}$ of modular temporal classes, and $\overset{t}{\succ}$ is a partial ordering relation over $\mathcal{T}_h$. This relation induces a special set $\mathcal{T}_{mh} \subseteq \mathcal{T}_h$, $\{mc_1, \ldots, mc_k\}$ and $k \leq n$, called the *main time hierarchy (MTH)*, defined by a sequence of relations as follows.

mod_temp_class$(flow\_time, c_k, infinity)$, where $c_k \in \mathcal{T}_C$
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 $m_v \in \mathbb{Z}^+$ or $m_v = x \ldots y$ and $x, y \in \mathbb{Z}^+$ and $x < y$.
mod_temp_class$(c_1, lowest, smallest)$, where $c_1 \in \mathcal{T}_C$.

Each pair c_i, c_j of $\mathcal{T}_h$ such that $c_j \overset{t}{\succ} c_i$ holds is related by *mod_temp_class*(c_j, c_i, m_v) and m_v is called the *modular value* of the modular set of c_i which defines c_j. Each c_i is called the name of the class. As $\overset{t}{\succ}$ (properties are in Appendix A is a partial ordering relationship, there will be some MTCs of the set $\mathcal{T}_h$ which will not hold such a relation between them. Note that the set $\mathcal{E}$ is a set of temporal entities which are the forms of TT we have presented, i.e. moment of time, cyclical, linear and collection interval. In the final case, these classes are out of the MTH and are said to be disjointed.

There are two things worth noticing. First, *flow_time* is defined as a special MTC which in fact does not belong to the hierarchy, but it is used to say that the class c_n is the highest class, and that the flow of time will be associated with infinite instances of c_n. Second, the fact that the lowest level of the hierarchy is defined in terms of a temporal constant symbol, i.e. *lowest*, allows us to interpret this structure as a hierarchy of discrete intervals. However, such an assumption is not necessary since we may consider the lowest level as belonging to the set of rationals, which would give us a dense model of time.

In the case that a non-regular MTC is defined, i.e. the modular value m_v is a range rather than a single value, then we need to specify the subclasses, the modular value for each one, and any relationship between them and other levels. Such a kind of MTC is useful for solving the problem related with the irregularity of the real calendar. This is done as follows.

(i) $subclasses(c, l)$, where c is a irregular MTC and c is a list of constant symbols representing the names of the subclasses of c.
(ii) $mod\_value(c, m)$, where c must be an element of a list of subclasses, as defined in the previous item, and $m \in \mathbb{Z}^+$.
(iii) $change\_mod\_value(c_i, s_1 \ldots s_2, p(d, c_j))$, where c_i is an irregular MTC which has its modular value ranging between s_1 and s_2 (positive integers), at every d units of time at the level of c_j, and c_j is a MTC defined by C_i.
(iv) $mod\_value(c_i, I_{c_j}, z)$, where c_i is an irregular MTC, I_{c_j} is an instance of the MTC c_j, which is defined by c_i, and z is the modular value of c_i according to the following recursive definition.

- $mod\_value(c_i, \_, z)$ iff there is some $Z \in \mathbb{Z}^+$ such that $mod\_value(c_i, z)$.
- $mod\_value(c_i, I_{c_j}, z)$ iff both $change(c_i, s_1 \ldots s_2, p(d, c_j))$ and $\mathcal{R}(I_{c_j}, d, s_1, s_2, c)$ hold, where $\mathcal{R}$ is a meta-predicate to represent a temporal relation between I_{c_j}, d, s_1, s_2 that will compute z.

For instance, the real calendar can be defined as follows.

$mod\_temp\_class(flow\_time, year, infinity)$.
$mod\_temp\_class(year, month, 12)$.
$mod\_temp\_class(month, day, 28 \ldots 31)$.
$mod\_temp\_class(day, lowest\_level, smallest)$.
$subclasses(month, [january, february, march, april, may, june, july,$
$\quad august, september, october, november, december])$.
$mod\_value(january, 31)$.
all other cases until $mod\_value(december, 31)$.
$change\_mod\_value(february, 28 \ldots 29, p(4, year))$.
$mod\_value(Class, \_, Z) \Longleftarrow$
$\quad mod\_value(Class, Z)$.
$mod\_value(Class, Year, Z) \Longleftarrow$
$\quad change\_mod\_value(Class, Z1 \ldots Z2, p(MF, year))$ &
$\quad Leap Year$ *is* $Year$ *mod* MF &
$\quad (Leap Year = 0$ &
$\quad Z = Z2$

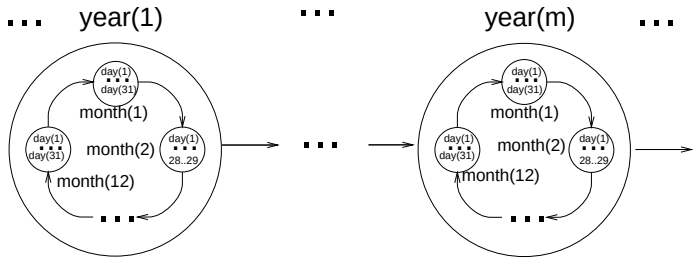

Figure 2. A view of the *Linear-Cyclic Hierarchy* of time. The inner circles represent lower levels of time granularity like *day* and *month*, while the outest circles represent the highest level, like *year*.

$$\vee$$
$$LeapYear > 0\,\&$$
$$Z = Z1).$$

A view of this hierarchy is depicted in Figure 2.

The temporal entity which we can represent through the MTH is the smallest interval $t(x_1, \ldots, x_n)$. In the calendar defined above the term $t(1, 1, 1996)$ represents the first day of the first month of the year 1996. Along with these MTC we may define other types which do not belong to MTH, e.g. week. This could be defined, for example, as follows.

$mod\_temp\_class(lunar\_month, week, 4)$.
$mod\_temp\_class(week, day, 7)$.
$mod\_temp\_class(labour\_week, day, 5)$.

The temporal entities of this MTCs are represented by collection intervals as described in Section 3.2.2. Although the logic has expressive power to represent such a kind of time interval, we will not explore it in this work since simulation models usually do not need such a kind of temporal reference.

3.4. PAST AND FUTURE RELATIONS

There are two relations for the notions of future and past. The first is for linear intervals, and it is used to implement the usual relations between linear intervals (Allen, 1983; Allen and Hayes, 1985). As linear intervals are represented by a pair of smallest temporal entities (moments of time), which is a structure relating instances of all MTCs of the MTH, then instead of using the less than ($<$) relation directly, as in the case of a single time scale, we have the following linear precedence relation.

DEFINITION 3.1. *Let $R = t(x_1, \ldots, x_k)$ and $S = t(y_1, \ldots, y_k)$ be two smallest intervals. We say that R is linearly precedent to S, written as $R \prec S$ if, and only if*

$x_k < y_k$, *or*
$x_i < y_i$ *and for all* x_j, y_j *such that* $i < j \leq k$ *and* $x_j \leq y_j$.

We also say S is a moment in the future in relation to R.

The second notion relates one TE in the past with another in the future, and also with the period of time between them. However, because a cyclical interval, at any level of granularity, is defined as a closed time structure, then every time "instant" is both before and after any other (including itself). Because of this, we use the expression *relative past* to mean that a given TE is the past of another one by a specific period of time, because there may exist many others that are also in the past but separated by different periods. Analogously, *relative future* means that a given TE is in the future of another one in relation to one specific period of time. Both concepts are captured by the predicate $future(S, P, T)$ to mean the future of S after P is T. In the case where both S and T are smallest intervals the definition is trivial and makes use of the temporal precedence relation as defined above and the period of time between them. The definition of this concept for cyclical intervals is defined as follows, where $\oplus$ is the modular sum operation of modular sets (Biggs, 1987).

DEFINITION 3.2. *Let S, T be two cyclical intervals of a MTC c_i, and suppose there is a MTC defined as $mod\_temp\_class(c_{i+1}, c_i, m)$, and P is a period of time of the level i. We say that S is the* relative past *of T, where the period of time between them is P, written as $future(S, P, T)$ iff $S \oplus P = T$. We say that T is the relative future of S.*

Now, we use these relations in order to map complex PTEs to canonical forms. This is done by using the operations of *up-wave* modular sum and subtraction defined in Appendix B.

DEFINITION 3.3. *Let $P = p(\Delta, c_i)$ be a period of time, T a temporal interval. Then P* after *T is the temporal entity in the future of T, defined as*

a- $i(s_1 \oplus \Delta \ldots s_2 \oplus \Delta, c_i)$, *if T is a cyclical interval* $i(s_1 \ldots s_2, c_i)$
b- $t(s'_1, \ldots, s'_n)\, \omega_\oplus\, P \ldots t(t_1, \ldots, t'_n)\, \omega_\oplus\, P$, *if T is a linear interval* $t(s_1, \ldots, s_n) \ldots t(t_1, \ldots, t_n)$.

The converse of this operator, *before* is easily defined if we change the *up-wave* modular sum by subtraction. Examples of how the *after* operator works are: $p(3, month)$ *after* $i(3 \ldots 4, month)$ is equivalent to $i(6 \ldots 7, month)$. $p(2, month)$ *after* $t(1, 10, 1990) \ldots t(12, 11, 1990)$ is equivalent to the interval $t(1, 12, 1990) \ldots t(12, 1, 1991)$.

3.5. DESIRED INFERENCES

Now, we are going to elaborate on the kind of temporal inferences that should be drawn when using **NatureTime** for dealing with cyclical events or for the specification of agents behaving at different levels of time granularity. In the first case, the mechanism is simple. In the second case, we will elaborate on a particular type of agent specification that, to our knowledge, has not been treated so far. This is for the specification of the behaviour of an agent is simulation models, specially those used for ecosystems. We will discuss why an executable temporal (logical) reasoning system is suitable for such a problem, and then what are the problems we may come across and how we can overcome them using **NatureTime**. Both cases may also be combined as will be shown in Section 5. In either case, the way in which a temporal sentence will being provable from a knowledge base will differ only in the way the proof is constructed.

3.5.1. TEMPORAL UNIFICATION

Before we describe the desired inferences we need to define the concept of temporal unification which is the core of our logic. Unification as traditionally understood is the process of determining whether two expressions can be made identical by performing appropriate substitutions for their variables.

Temporal unification is the process of determining whether two temporal entities (intervals or moments of time) have some temporal entity in common. In traditional temporal reasoning jargon this is equivalent to finding if they are equal, overlap or one is included into the other (i.e. during, starting or finishing). Note that we do not mention substitution of variables, it is implicit that if one (or both) of the temporal entities is (are) a variable, then one may substitute the other. A full description of how this unification works is given in the next section.

3.5.2. TEMPORAL QUERY AND PROVABILITY

We now give the notion of what one would expect as a correct answer for a given temporal query about a knowledge base. Formally, we have.

DEFINITION 3.4. *Let Δ be a set of temporal formulae (i.e. a knowledge base). For any query $\phi \,@\, \tau$ we want to know if there is an interval τ' (possibly more than one), where τ' occurs within τ (possibly equal to it), such that $\phi \,@\, \tau'$ holds.*

The concept underlying the process of proving a given query is *temporal provability* defined as follows.

DEFINITION 3.5. *Let Δ be a set of temporal formulae, $\phi \,@\, \tau$ a temporal formula. We say that $\phi \,@\, \tau$ is* temporally provable *(or* temporally derivable *from Δ, written $\Delta \vdash_t \phi \,@\, \tau$ if by systematically applying modus ponens, along with standard and temporal substitution, to the set of temporal assertions and temporal logical axioms of Δ we can find a temporal substitution τ' for τ such that we can derive the formula $\phi \,@\, \tau'$, written $\Delta \vdash_t \phi \,@\, \tau'$.*

Given a certain goal, the interpreter we are going to present in the next section searches systematically for derivations of temporal formulae according to the above definition.

Because it is well known that provability intuitively suggests how logical implication can be automated, then we may say that $\Delta \vdash_t \phi \,@\, \tau$ is equivalent to say that $\Delta \models_t \phi \,@\, \tau$, i.e. $\phi \,@\, \tau$ temporal logically follows from Δ.

3.5.3. REASONING ABOUT CYCLICAL EVENTS

Any inference about any cyclical event E can be obtained by, first, specifying an atomic temporal formula involving the event and the interval of time at which such a repetition happens at a given time granularity. In this case the temporal entity T should be a cyclical interval, e.g. $i(s_1 \ldots s_2, c_i)$. Then we write $E \,@\, i(s_1 \ldots s_2, c_i)$ to mean that *E is true throughout the cyclical interval $s_1 \ldots s_2$ at the level i of granularity defined by the MTC c_i.*

Example 2: Suppose we have the following sentences in a knowledge base

$harvest(corn) \,@\, i(5 \ldots 6, month)$.
$harvest(herb\_tea) \,@\, i(2 \ldots 10, month)$.
$harvest(coffee) \,@\, i(8 \ldots 4, month)$.
$harvest(rice) \,@\, I \Longleftarrow (harvest(tea) \,\&\, harevst(coffee)) \,@\, I$.

The types of inference cyclical reasoning about this KB could be:

Q - $harvest(corn) \,@\, i(6 \ldots 9, month)$.
A - $\{harvest(corn) \,@\, i(6 \ldots 6, month)\}$
Q - $harvest(corn) \,@\, i(3 \ldots 5, month)$.
A - $\{harvest(corn) \,@\, i(5 \ldots 5, month)\}$
Q - $harvest(rice) \,@\, T$.
A - $\{harvest(rice) \,@\, i(8 \ldots 10, month), harvest(rice) \,@\, i(2 \ldots 4, month)\}$
Q - $harvest(corn) \,@\, t(x_1, x_2, x_3) \ldots t(y_1, y_2, y_3)$
A - $harvest(corn) \,@\, t(1, 3, x_3) \ldots t(31, 5, y_3)$

Note that the last query is more specific in the levels of time required. This will be very useful when we have processes involved at different granularities. The process of reasoning is standard backward chaining. As shown in Mota (1994), it is suitable for representing entailed events (cyclical or not) of temporal knowledge for ecological domain and for reasoning about when an implication holds in a given knowledge base, i.e. for a knowledge base Δ we may be interested to know if $\Delta \vdash P \rightarrow Q$, where P and Q are atomic sentences. However, because backward chaining involves looking back in time for clauses for events, under certain circumstances, forward chaining can be computationally more attractive. One of these is the case of reasoning about simulation models of ecosystems (i.e. simulating the behaviour of some agent), where backward chaining does not seem to provide reliable solutions for testing more complicated problems as in the case of many agents interacting at different scales of time. Our next step will be to show why we also need forward chaining, and for what kind of inferences it is useful.

3.5.4. REASONING ABOUT AN AGENT'S BEHAVIOUR

There are some cases in which the behaviour of certain entities 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 the best for the domain of ecosystems. For example, if we want to represent the behaviour of agents which immigrate and emigrate from one population to another, then it is inconvenient 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 (sometimes with a fine grained "internal" time scale to approximate continuous change). This yields a discrete approximation to continuous models. However, a discrete approach does not usually allow us to compute the value of some attribute at a time in between two consecutive time steps. For instance, if we have the specification of some ecological entity working at a weekly scale of time, then because its attribute would be updated only at every week we could not obtain its value separately within a week. To overcome this, an additional mechanism seems to be needed beyond the standard way of proving a sentence.

One way of obtaining such a value is to assume that the attribute changes in a linear

fashion, and then using a linear equation to estimate it. For more complex processes a non-linear behaviour might be assumed and the value would be computed in the same way. When doing this, we have to assume that the process does not interact with other processes (possibly of other agents) that may affect the attributes in question. The reason for such an assumption is that the function used to estimate the desired value needs the values of the attribute at the end points of the interval between the consecutive time steps, i.e. the previous and the future values. However, if there is such an interaction, then we cannot estimate the future value because it may happen that the other process(es) have influenced so much that the effect over the attribute can be much more than the expected. Moreover, to estimate such a future value it is needed to know the wanted value which leads us to a situation of "deadlock".

For simplicity we avoid interpolation between time points. In this case every entity with its behaviour specified at a coarser level will have its state changed only at that level, i.e. any query about its state at any time between two consecutive time steps will always give the computation for the last one. Based on these assumptions, the temporal reasoning system should reason about queries like:

1. What is the value of an attribute of an entity at any given time?
2. When will any two entities (no relation between them) have the same value?
3. What is the value of a given entity at a given time which interacts with another entity which works at different time scale?

There are two important components to this problem. First we want the deduction process to be able to search for a value which satisfies one query, but this value should be determined by a given temporal entity. This time can be synchronous or asynchronous in relation to the updating time steps of the attributes of the ecological entity we want to know something about. Second, the computation of the value of an attribute of an agent at a given time i depends on the value at time $i-1$. Because of this, a simulation model can be interpreted as being a relation between the state of the agent in the past and its state in the future ruled by certain conditions of state transition.

This idea of constructing the future state based on the past is known as *declarative past and imperative future* (Gabbay, 1989). In this view programs should be expressed in a temporal language which could be read in a declarative way, and when the program runs it should construct a model for the temporal sentence it is intended to represent (Gabbay and Reynolds, 1995). Relating it to the usual simulation models of ecosystems are developed, this is equivalent to describing the behaviour of agents using a temporal logic. The simulation of the behaviour of the agent through the flow of time corresponds to the construction of a model, or running the program for a temporal sentence at each time step. As our language deals with explicit representation of time, the usual way of representing *past* $\rightarrow$ *future* in this work should consider the length or period of time between past and future (or present). As far as simulation models are concerned, such an imperative formulation can be represented in **NatureTime** as in the following schematic way.

$$A' \,@\, P \; \mathit{after}\; T_p \Longleftarrow A \,@\, T_p \,\&\, \mathcal{R}(A, A').$$

where A represents the information about an ecological species, T_p the previous time, P is normally in the form $p(1, C)$, C is an MTC, and the second order 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. However, Gabbay's approach proposes not to use the traditional execution mechanisms based upon resolution and refutation. In this paper we do not need to make such an assumption. If A and A' both simply refer to the same predicate (e.g. A is $value(a, X_p)$ and A' is $value(a, X_f)$), then we do not have to worry about branches in the search space. We simply follow a single chain of conclusions until we have reached some desired (possibly the final) time.

For instance, suppose we have the following hypothetical agent specification.

1 - $value(a, 100) \,@\, t(1, 1, 1)$.
2 - $value(a, X_f) \,@\, p(1, week)\ after\ T_p$
$\Longleftarrow$
$value(a, X_p) \,@\, T_p \,\&$
$X_f\ is\ X_p + 1$.

The execution of this program would generate the value of a for every week, but without using the traditional combination of backtracking and recursion until the initial state. Instead, the generation of the model for the sentence should get the previous state of the computation (past) in order to produce the next state (future). For this example we have the following table with the first steps of execution for a goal $value(a, X) \,@\, T$, showing the present state of computation, the sentence being matched for each backtracking step for the generation of the future state. After the initial state the sentence used is always the second, and to compute a new state it is not necessary to reach the initial state again as backward chaining would do.

Present	Sentence matched	Future
—	1	$value(a, 100) \,@\, t(1, 1, 1)$
$value(a, 100) \,@\, t(1, 1, 1)$	2	$value(a, 101) \,@\, t(1, 7, 1)$
$value(a, 101) \,@\, t(1, 7, 1)$	2	$value(a, 102) \,@\, t(1, 14, 1)$
$value(a, 102) \,@\, t(1, 14, 1)$	2	$value(a, 103) \,@\, t(1, 21, 1)$

4. 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.

4.1. A META-INTERPRETER FOR NATURETIME

The meta-interpreter is an extension of the one presented by Sterling and Shapiro (1986). Because we are interested in problems where the specification of the behaviour of agents through the flow of time is basically a clause where the state of the agent in the past is related to the (present or) future, then we also allow the meta-interpreter to deal efficiently with this kind of clause. This will be presented in Section 4.1.2.

4.1.1. EXTENDING A STANDARD META-INTERPRETER

The extensions are basically twofold. First, the predicate *clause* must also identify well-formed temporal formula as specified in Section 3.2.2; second, the introduction of a special unification for PTEs. Note that most of the unification will still be done by standard Prolog unification, except when involving PTEs. However, 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 (Sterling and Shapiro, 1986) is changed to be a binary relation $solve^2$. The meta-interpreter accepts queries which are temporal formulae. It succeeds if the query holds throughout some interval within the given PTE as specified in Definition 3.6. So, the first argument is the temporal formula to be solved, and the second the result of the computation for the first argument be true. In what follows, $system(X)$ succeeds if X is a system predicate but not a negation. The meta-interpreter is defined as follows.

$$
\begin{array}{ll}
1 - & solve(X, X) :- \\
 & \quad \neg temp\_formula(X), \\
 & \quad (clause(X \Longleftarrow Y), \\
 & \quad\ solve(Y, \_) \\
 & \quad\ ; \\
 & \quad\ system(X), \\
 & \quad\ call(X)). \\
2 - & solve(\neg X, \_) :- \\
 & \quad \neg solve(X, \_). \\
3 - & solve(A \,@\, T1, A \,@\, T3) :- \\
 & \quad \neg A = (\_ \,\&\, \_), \\
 & \quad clause(A \,@\, T2), \\
 & \quad temp\_unify(T1, T2, T3). \\
4 - & solve(A \,@\, T, A \,@\, RT) :- \\
 & \quad A = (A1 \,\&\, A2), \\
 & \quad solve(A1 \,@\, \_, A1 \,@\, T1), \\
 & \quad solve(A2 \,@\, \_, A2 \,@\, T2), \\
 & \quad temp\_unify(T1, T2, RT1), \\
 & \quad temp\_unify(T, RT1, RT). \\
5 - & solve(A \,\&\, B, A1 \,\&\, B1) :- \\
 & \quad solve(A, A1), \\
 & \quad solve(B, B1). \\
6 - & solve(A \vee B, R) :- \\
 & \quad (solve(A, R) \ ; \\
 & \quad solve(B, R)). \\
7 - & solve(A \,@\, T1, A \,@\, T3) :- \\
 & \quad clause(A \,@\, T2 \Longleftarrow Body), \\
 & \quad solve(Body, \_), \\
 & \quad temp\_unify(T1, T2, T3).
\end{array}
$$

The first clause deals with classical formula in the usual way. The second implements the negation by failure as using the standard way of negating if it cannot be proved. The third clause consults the Knowledge Base (KB) to check whether we can directly

match $A \,@\, T1$ to some unit clause or not. The fourth clause deals with composite events, where we want to know if there is some time interval throughout which they can happen all together. Clauses 5 and 6 are just another way of re-writing 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 the clauses of the meta-interpreter we need to understand the declarative interpretation of each one; they should be interpreted as follows.

3 - *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 $T2$.*
4 - *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 $RT1$.*
7 - *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 $T2$.*

In the first (1) and in the last clause (7), when the Y and *Body* are solved, any occurrence of temporal entities within them will be substituted by standard unification. The value computed by the second argument is ignored here. This can give unsound results if there are shared variables present. We restrict programs and queries to avoid this situation. The ideal solution is either if the solver carries a list L of temporal variables and binds them only after solving all temporal formulae in a wftf, or having a sort of memory for storing operations over PTE and combining those stored with the new one during the process of deduction. This is similar to the idea of *environment* (van Emden, 1984) used in the implementation of Prolog machines. However, this would require a more sophisticated meta-interpreter, and for the purposes of this work this simple solution provides a powerful mechanism for a significant subset of temporal problems.

4.1.2. DEALING WITH SIMULATION MODELS

As we have discussed in Section 3.5.4 it is interesting to deal with simulation clauses in a different way. The process of reasoning should be Forward Chaining rather than backward, and we also want to obtain the state of agent at any time in between two consecutive time steps of its behaviour, i.e. asynchronous information. Although the basic extension defined in the last section can be used for the specification of simple simulation models (Mota *et al.*, 1995a), we have shown (Mota *et al.*, 1995b) that for more complicated models, where it is needed to obtain asynchronous information, the *solver*[2] does not offer mechanisms for this. In this work we will provide the meta-interpreter with such a facility. Because we are considering discrete models, then every entity with its behaviour specified at coarser levels will have its state changed only at that level, i.e. any query about its state at any time between two consecutive time steps will always give the computation for the last one.

For a given goal $A \,@\, T$ the solving process should first check if the time requested is a fixed time. If it is the case, then verify if this formula can match with part of the body of a temporal formula like the schemata given in Section 3.5.4, which we call a *simulation clause*. After this, instead of using the meta-interpreter as above (which would apply

backward reasoning), the search should re-use, at every time step, the last value for the state of the entity in order to reason forwards.

What if the time in the goal is not a ground temporal term? We will use the same solution, except that the searching process does not need to check the next time with the previous one in order to stop the search.

In the more complicated case, the searching process basically checks if the solution found is that of the specified time. If it is not the case, then if the given time T is the future of T_p and the PTE P *after* T_p is not the future of T, a new instance of the simulation clause is used to continue the search. More formally we have.

DEFINITION 4.1. *Given an atomic temporal assertion $A_i \,@\, T_i$ (or the initial state for A), a goal $A_1 \,@\, T$, one (or more mutually exclusive) simulation clause schemata*

$$C = A' \,@\, P \textit{ after } T_p \Longleftarrow A \,@\, T_p \,\&\, \textit{Constraints},$$

where A, A', A_i and A_1 have the same predicate name and arity. Then, the forward computation which will search for a solution for the goal $A_1 \,@\, T$ is given as follows.

if T is ground temporal term then

reduce T to a canonical PTE, say T_1
$T'_i = T_i$
$A'_i = A_i$
while $T'_i \prec T_1$ then

get a new instance C' of C, matching $A \,@\, T_p$ with $A'_i \,@\, T'_i$, and
future(T'_i, P, T'_j) holds and
if $T_1 \prec T'_j$ holds then stop the search, otherwise
if $\neg\, T_1 \prec T'_j$ holds and Constraints is found to be true by the solver clauses then
$T'_i = T'_j$ and $A'_i = A'$

if T is a temporal variable then

get a new instance C' of C, matching $A \,@\, T_p$ with $A'_i \,@\, T'_i$, and
future(T'_i, P, T'_j) holds and
Constraints is found to be true by the solver clauses then
$T'_i = T'_j$ and $A'_i = A'$

return $A'_i \,@\, T'_i$

Note that this algorithm is guaranteed to terminate, in the case that T is a ground TT, because of the fact that the condition $T'_i \prec T_1$ will eventually fail, or $T_1 \prec T'_j$ will eventually hold. If more solutions are requested, then backtracking over the clause selection strategy is possible to find another one only if T is not a ground TT because there will always be a T'_j such that *future*(T'_i, P, T'_j) holds. But in the case $T1$ (the reduced form of T) is a ground TT, then once $\neg\, T'_i \prec T_1$ holds the computation will never get new instances of C, and so no more solution will be found.

4.2. GENERAL UNIFICATION ALGORITHM

The specialized unification algorithm we developed to treat PTEs consists of two steps. First, every complex PTE is reduced to a canonical form of temporal entities. Second, as

canonical forms of a temporal entity are in fact intervals of time, or collections of them, then they are unified according to the usual relations between time intervals. These are based on the future and past relations, and also in the relation between periods of time at different levels of granularity. The result of a unification between two temporal entities is not their most general unifier as is traditional, but rather the canonical form of temporal entity which is the result of their matching, and this can involve more than one temporal entity as an example will show. The *temporal unification* is defined as follows.

DEFINITION 4.2. *Let t_1, t_2 and t be three PTE. We write temp_unify(t_1, t_2, t) to mean that t_1 and t_2 are unifiable and t is the unified term iff either*

- $t_1 = t_2 = t'$ *and t is the reduction of t', written reduce(t, t'), or*
- $t_1 \neq t_2$, *and*
  t_1 *is reducible to* rt_1 - reduce(t_1, rt_1) *and*
  t_2 *is reducible to* rt_2 - reduce(t_2, rt_2) *and*
  t *is reducible to* t' - reduce(t, t') *and*
  t' *is the matching of* rt_1 *and* rt_2 - unify_units(rt_1, rt_2, t').

For instance, suppose we want to unify $i(2 \ldots 10, month)$ (or all intervals from February to October) with $i(8 \ldots 4, month)$ (or all intervals from August to April). In this case there will be two different instances of the canonical form of cyclical interval, i.e. $i(2 \ldots 4, month)$ and $i(8 \ldots 10, month)$. How unification works is related to the unification of temporal units as defined in Section 4.4.

4.3. REDUCTION OF TEMPORAL TERMS

The reduction algorithm is divided into two groups of clauses. The first consists of the clauses to deal with canonical forms of temporal expression. The second, deals with complex forms. First we introduce what we mean by temporal variable: a term t is a temporal variable if it is a logical variable, i.e. if $t \in \mathcal{L}_v$, or a canonical form of PTE where all its elements are logical variables. Based on it we have the following reduction algorithm.

DEFINITION 4.3. *Let t_1 be a PTE, and t_2 a canonical PTE. We say that t_1 is reducible to t_2, written reduce(t_1, t_2) iff one of the following holds.*

- $t_1 = t_2$, *and t_1 is a temporal variable, temp_var(t_1).*
- $\neg$*temp_var(t_1) and is in the form $t(x_1, \ldots, x_k)$, so $t_2 = t_1$*
- $t_1 = s_1 \ldots s_2$, *and* $t_2 = rs_1 \ldots rs_2$ *where*
  s_1 *is reducible to* rs_1 *and*
  s_2 *is reducible to* rs_2.
- $t_1 = t_2 = i(s \ldots t, c)$
- $t_1 = t$ *after* Δ *and*
  t_1 *is reducible to* rt_1 *and*
  t_2 *is reducible to* rt_2 *and*
  future(rt_1, Δ, rt_2) *holds.*

- $t_1 = t$ *before* Δ *and*
  t_1 *is reducible to* rt_1 *and*
  t_2 *is reducible to* rt_2 *and*
  $future(rt_2, \Delta, rt_1)$ *holds.*

For instance, $p(1, week)$ *after* $(p(1, week)$ *plus* $p(3, day)$ *after* $t(14, 2, 1994))$ is reduced to $t(1, 3, 1994)$.

4.4. UNIFICATION OF TIME UNITS

The unification on the level of units is just the matching of temporal entities. This has to deal with linear and cyclical intervals, and also intervals of both types. The linear unification is based on the $\prec$ ordering relationship of the bounding temporal entities of the interval, and the modular unification is based on the matching of circular intervals. In both cases we use the relations *during* and *overlap* (Allen and Hayes, 1985) taking into account that the "last" element of a modular set is followed by the first. Because of this, the linear match has four different cases, and the modular case six as described in Appendix C.

Its definition is as follows.

DEFINITION 4.4. *Let* s_1, s_2 *and* s_3 *be three canonical forms of PTEs. We say that* s_3 *is the unit matching from* s_1 *and* s_2, *written* $match\_units(s_1, s_2, s_3)$ *if one of the following holds.*

- $s_1 = s_2 = s_3$.
- s_1 *and* s_2 *are linear intervals, and* s_2 *is during* s_1, $s_3 = s_2$.
- s_1 *and* s_2 *are cyclical intervals of a MTC* c, *and*
  $mod\_temp\_class(c', c, mv)$ *holds and*
  s_3 *is the cyclical interval from the modular matching between* s_1 *and* s_2. *(could be more than one matching)*
- s_1 *is a cyclical interval and* s_2 *is a linear interval and*
  s'_1 *is a* linear instance *of* s_1, *written* $time\_instance(s_1, s'_1)$, *and*
  s_3 *is the linear matching between* s'_1 *and* s_2
- s_1 *is a linear interval and* s_2 *is a cyclical interval and* $match\_units(s_2, s_1, s_3)$ *holds.*

Note that *time_instance* creates a 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. For this reason, this is one of the most important features of the logic because without it, the concept of cyclical interval would be useless. In a pure linear model of time, it is necessary to introduce some "expert" computation over recurrent representation, in order to obtain reasoning about cyclical events. In our work, the use of cyclical interval provides an easy and elegant mechanism to represent and obtain reasoning about cyclical events and processes, since this new type of interval represents many instances in the linear level.

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

$harvest(corn) \,@\, i(5 \ldots 6, month)$.
$harvest(herb\_tea) \,@\, i(2 \ldots 10, month)$.
$harvest(coffee) \,@\, i(8 \ldots 4, month)$.
$harvest(rice) \,@\, I \Longleftarrow (harvest(tea) \,\&\, harevst(coffee)) \,@\, I$.

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 calls the *solve*[2], and shows the second argument of it as the answer.

: | $harvest(corn) \,@\, i(6 \ldots 9, month)$.
>> $harvest(corn) \,@\, i(6 \ldots 6, month)$
: | $harvest(corn) \,@\, i(3 \ldots 5, month)$.
>> $harvest(corn) \,@\, i(5 \ldots 5, month)$
: | *more.*
>> *Sorry, no further solution is possible.*
: | $harvest(rice) \,@\, T$.
>> $harvest(rice) \,@\, i(8 \ldots 10, month)$
: | *more.*
>> $harvest(rice) \,@\, i(2 \ldots 4, month)$
: | *more.*
>> *Sorry, no further solution is possible.*

The *more* command allows all possible solutions by backtracking. In the next section we shall show two examples of simulation models specified in our language, and how they are solved.

5. Simulation Models in NatureTime

This section presents the application of our logic language to develop a simulation model for the growing process of the trees of the example of Section 2, and we also present another example of two temporal entities working at different time scales and which interact with one another. After this, we point out the limitations of this implementation and of the language.

5.1. EXAMPLE OF TREE GROWING PROCESS

Usually, a discrete model of the height of a tree t_i at a given time $t + 1$ might be represented by an equation of the form

$$H_i(t+1) = H_i(t) + r_i H_i \left(1 - \frac{H_i}{MAX_{H_i}}\right), \tag{5.1}$$

where MAX_{H_i} is the maximum height that a tree t_i can reach, r_i is the intrinsic 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. However, it does not explicitly model the interaction among trees, which may affect this rate. 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. The increase of the height at every time step will be given as follows.

$$\Delta'_H = \left(r_s - \sum_{j=1}^{nb} k\frac{H_j}{d_{ij}}\right) H_i \left(1 - \frac{H_i}{MAX_{H_i}}\right), \tag{5.2}$$

where r_s is the standard growth rate of a tree if no influence is present, nb is the number of tree neighbours of t_i.

In our previous scenario, as the scale of time used was week, we could define another MTC within the hierarchy defined, that is $mod\_temp\_class(week, day, 7)$. The rainy season, assumed to be only in February, can be written as $season(rain) \,@\, i(2 \ldots 2, month)$. 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 (5.1), 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 (5.2). 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.

$height(Tree, H) \,@\, p(1, week)\ after\ T$
$\Longleftarrow$
$height(Tree, H1) \,@\, T\ \&$
$max\_height(Tree, MAX)\ \&$
$real\_gr(Tree, T, \ldots, (p(1, week)\ after\ T, RGr))\ \&$
$neighbours(Tree, NTrees)\ \&$
$influences(NTrees, T, R)\ \&$
$sum(R, TR)\ \&$
$(Gr\ \ is\ \ RGr - TR)\ \&$
$(C\ \ is\ \ Gr * H1 * (1 - H1/MAX))\ \&$
$(H\ \ is\ \ H1 + C).$

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

$real\_gr(Tree, T, Gr)$
$\Longleftarrow$
$season(rain) \,@\, T\ \&$
$growth\_rate(Tree, LowGr)\ \&$
$(Gr\ \ is\ \ 1.2 * LowGr).$
$real\_gr(Tree, T, Gr)$
$\Longleftarrow$
$\neg\, season(rain) \,@\, T\ \&$
$growth\_rate(Tree, Gr).$

Gr is the growth rate if T is a rainy season and $Lowgr$ is a standard growth rate, then $Gr = LowGr * 1.2$. Gr is the growth rate if it is not true that T is a rainy season. Note that we call the *solve* meta-interpreter to use its facilities of unification as we saw in Section 4. The complete knowledge base of our example is shown in the Appendix D.

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

$| : height(t1, H) @ T.$
$>> height(t1, 1) @ t(14, 2, 1994)$
$| : more.$
$>> height(t1, 1.01) @ t(21, 2, 1994)$
$| : more.$
$>> height(t1, 1.02) @ t(28, 2, 1994)$
$| : more.$
$>> height(t1, 1.03) @ t(5, 3, 1994)$
$| : more.$
$>> height(t1, 1.037) @ t(12, 3, 1994)$
$| : more.$
$>> height(t1, 1.045) @ t(19, 3, 1994)$
$| : more.$
$>> height(t1, 1.054) @ t(26, 3, 1994)$
$| : more$
$>> height(t1, 1.062) @ t(3, 4, 1994)$

5.2. TWO ECOLOGICAL SPECIES WORKING AT TWO TIME SCALES

The following example is part of our discussion on the granular aspects of time in simulation models of ecosystems (Mota *et al.*, 1995b).

> *Example 4*: The ecological entities are a tree growing, called simply *tree* and an insect (or a cloud of insects) called *bug*. The *tree* has its growth rate affected by the *bug* only if it flies on the top of the *tree*. The height reached by the *bug* will depend on the height of the *tree*, and the height of the *tree* depends on the time spent by the *bug* at a certain position, say 5 m. For simplicity, instead of using the logistic equation for the growing process, the tree grows 0.5 m every week. The time scale of the bug's movement is considered to be day.

There is an interaction between these two entities, and we will assume that the tree should get the "progress" of the bug's position in a period of one week. However, in the *tree*'s behaviour specification there is no need to explicitly represent the scale of the *bug*, because it could be another entity interacting with it. Then we shall use the predicate $scale(time, Object, Process, MTC)$ to specify at which scale of time the process of a given entity works. In this way we have the following facts in our KB.

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

$value(pos, bug, 6) \,@\, t(1, 1, 1)$.

Now we need a general way to capture the progress of an attribute of a given agent throughout a certain period of time. We will use the predicate *progress*[4] to represent 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 of all the values for *Att* during T. A simple specification for *progress*/4 is in Appendix E, and a more deep discussion on the specification of interacting agents working at different levels of time granularity is out of the scope of this work. More details can be found in Mota *et al.* (1995b). The specification of the *tree*'s growing process can be, for example, as follows.

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

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

$value(pos, bug, PB) \,@\, p(1, day)$ *after* T
$\Longleftarrow$
$value(pos, bug, PBi) \,@\, T$ &
$value(height, tree, H) \,@\, T$ &
$new\_pos(PBi, H, PB)$.

The *new_pos*[3] simply implements a change of the bug's position, and it is also in the Appendix E. For this specification we have the following results for the simulation of the bug's position.

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

For the tree's growing process we have.

$| : value(height, tree, H) \,@\, T$.
$>> value(height, tree, 9) \,@\, t(1, 1, 1)$
$| : more$.
$>> value(height, tree, 9.44) \,@\, t(8, 1, 1)$

...
$>>$ $value(height, tree, 11.2) \,@\, t(6, 2, 1)$
$|:$ $more.$
$>>$ $value(height, tree, 11.64) \,@\, t(13, 2, 1)$

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

$|:$ $value(height, tree, H) \,@\, t(10, 1, 1).$
$>>$ $value(height, tree, 9.44) \,@\, t(10, 1, 1)$
$|:$ $value(height, tree, H) \,@\, t(10, 2, 1).$
$>>$ $value(height, tree, 11.2) \,@\, t(10, 2, 1)$

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

6. Analysis and Related Work

6.1. Strengths of the NatureTime system

NatureTime is a comparatively simple logic which does not stray far from the traditional style of mainstream logic programming—yet it can deal with a wide range of the problems commonly encountered in ecological modelling and simulation. Moreover, when much of the ecosystem is stable (i.e. unchanging when events happen) then most of the updating would be redundant. By using a logic we have presented in this paper we only need to deduce changes to the relevant aspects of the ecosystem. Thus if our simulation needs to include things working at a very small temporal granularity (like a little insect) together with things which change slowly (like a tree), it is redundant to update the state of the tree with every event affecting the insect. Our system does not involve this redundancy as we have demonstrated in the Example 3 of Section 5.2.

In terms of representation, **NatureTime** has a simple and elegant mechanism for the representation of cyclical knowledge by using the concept of cyclical intervals. This allows us to write $harvested(coffee) \,@\, i(8 \ldots 4, month)$, without the necessity of saying that harvesting can occur every year, since the MTC month is included in the definition of year, and year is flow of time. This can also be used to create another MTC which are not nested within the main hierarchy. For instance,

$mod\_temp\_class(labour\_week, day, 5).$
$mod\_temp\_class(labour\_month, labour\_week, 4)$
$mod\_temp\_class(lunar\_month, week, 4).$

In this paper we did not present any mechanisms to deal with more complex cyclical intervals, and other temporal entities like collection intervals and fluctuation of temporal entities over other temporal entities. This is basically related to introduce mechanisms of inference to reason about sentences like "all Mondays of 1996", although it does not

seem to be useful in simulation models. Such a treatment, here, would distract from the main point of this work.

The temporal operator *after* allows a more legible reading of some temporal statements, and it represents well enough the relation between past and future at different levels of time granularity. This leaves the user free to write his/her inference rules without the necessity of using properties and relations between temporal units. It is just needed to understand what a temporal formula is intended to mean and how we can map temporal knowledge to the forms of expressions of the language. For instance, the sentence *grass is free to grow up 1 month* after *a certain month X if sheep use the meadow up to X* can be easily translated to our logic as follows.

$$grow\_free(grass, meadow) \,@\, p(1, month) \;\; after\; i(X \ldots X, month)$$
$$\Longleftarrow$$
$$use(sheep, meadow) \,@\, i(Z \ldots X, month).$$

6.2. LIMITATIONS OF THE NATURETIME SYSTEM

The main limitation of **NatureTime** with earlier approaches using traditional computational logic, is its exponential search requirement in the case we have many agents interacting. However, because it uses a forward chaining strategy to re-use the previous computation of a given attribute, then the reduction of search space is considerable.

Another limitation which does not seem to affect the application for the more precise problems we proposed to tackle, is the lack of a suitable temporal connective for the representation of sentences like "it rained from 3 pm to 5 pm sometime last week". This could be the operator $\diamond$ usually known as "sometimes".

6.3. RELATED WORK

One early investigation on the representation of time clock on any scale was proposed in Ladkin (1986a), where interval calculus (Allen, 1983) is extended to achieve a time framework where different time units (TU) can be specified. This extends the idea of *convex* to *union-of-convex* intervals (Ladkin, 1986b), where there may exist gaps between *convex* intervals. The representation of *Basic Time Units* is a sequence like $[year, month, day, hour, \ldots]$. In Ladkin (1987) it was shown that by introducing appropriate relations between intervals such a sequence gives a suitable representation for a convex rational interval structure. We provide a similar entity that we called *smallest interval* which also has as many elements as desired, but we reach this representation from different concepts, i.e. this will depend on the number of MTCs defined. In **NatureTime** it seems to be a bit easier to define calendars, and we incorporate circularity within the model. Although *union-of-convex* intervals can be compared to our cyclical interval, Ladkin did not explore the subject of dealing with cyclical events.

A similar approach was proposed in Leban *et al.* (1986), where the basic idea 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 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 really deal with different granularities of time because it does not seem to be clear

how we can we specify relationships between propositions defined over different time scales. Furthermore, the way in which circularity is obtained is very much dependent on the implementation of circular list rather than the concept itself, although it gives a close idea of it.

Another approach for representing cyclical events was proposed by Koomen (1989), which is based on Allen's system (Allen, 1984; Allen and Hayes, 1985). The idea is to define a recurrent event e by stating explicitly that it is true repeatedly over an interval I, i.e. $\mathrm{RT}(I, e)$. However, it is not clear whether the interval I is *convex* or not, and it is not obvious how to use the mechanisms of inference to model cyclical events in a "natural" way to obtain appropriate inferences. As this approach does not deal with metric information, no representation of propositions at different time scales is possible.

A more pragmatic approach for dealing with time granularity was proposed in Dean (1989), in order to speed up the information retrieval on a large temporal data base maintained by the *time map system*. This work proposes a hierarchical framework of time such that events at different levels of abstraction can be easily represented and retrieved by using a structure similar to the usual calendar. The hierarchy over a linear structure of time is obtained by the concept of a *partitioning scheme*, which is a sequence of partitions $P_1, P_2, \ldots, P_n$ of the set of reals, in such a way that for each $i < n$ if an interval I belongs to a partition P_i, then there is a set of time intervals in P_{i+1} such that I is partitioned by it. Although this approach shows to be very successful in the context of data base maintenance, the aspect of representing and reasoning about cyclical events was not explored. Moreover, it does not seem to be a suitable approach for modelling more complex problem in the real world. As we often want to obtain some prediction via inference rules, and not only retrieve assertions about temporal knowledge, then this approach is not suitable for the type of problem we are dealing with.

A more recent approach (Ciapessoni *et al.*, 1993; Montanari, 1994) 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 (Rescher and Urquhart, 1971) (i.e. standard propositional logic added with a parameter operator $P\alpha$, where $P\alpha(p)$ is intended to mean "proposition p is realized at the position α"). The first identifies the domain or level of time granularity at which a given formula has to be considered. The second is used to constrain formulae to different domains. The hierarchy of time is a linear structure called the universe of domains, where a *granularity ordering* relationship is imposed over this universe. Also a partial ordering of *disjointedness* is provided to relate domains at different levels of granularity. This is similar to the *partitioning scheme* of Dean's approach (*op. cit.*). Temporal domains are related to our concept of *modular temporal class*. Their concept of locally temporally valid is related to the meaning of the throughout temporal connective. Finally, because we define grains of time based on modular chain of modular sets, cyclicity of events happening at different granularities is more easily obtained.

In the context of reactive systems specification and reasoning, Fiadeiro and Maibaum (1994) proposes a hierarchical (vertical) decomposition (or abstract implementation), of object specification in temporal logic. Such objects are seen as building blocks of the design process of reactive systems. At each layer of such a hierarchy there is a logic dealing with a single time scale, isomorphic to the set of natural numbers, and there is a collection of objects that may be used for composing complex objects (systems)at higher levels of abstraction. In this way, temporal execution of an abstract action is done by the temporal execution of concrete actions of the level below. The interface between

both levels is given by axioms which says when the concrete actions start, are being executed or have finished. This work does not intend to represent or reason about time explicitly. However, close observation shows us that the granularity of time is embodied within the specification of actions at each level. We could not see how cyclical processes might be represented in such a framework. Although it is allowed to represent interaction between abstract and concrete actions, the opposite direction of relation does seem to be straightforward. Our approach, though based on explicit reference of time and different mechanisms, is more general because we allow interaction in both direction.

An approach which allows many granularities in the same logic, for specifying asynchronous execution of agents is proposed in Fisher (1995). In this work, granularity (though not mentioned) is achieved by providing each agent with its own local clock represented by the predicate $tick(O)$. The problems with the "next" operator is solved by using the auxiliary predicates $next\text{-}tick(O, X)$—which is true if X is satisfied within the next O tick, analogously $last\text{-}tick(O, X)$. No mechanism is proposed for dealing with interacting agents working at different ticks of the global clock.

The systems we have mentioned so far do not provide mechanisms for representing events, or actions at different levels of time granularity and cyclicity in the same logical framework. Only recently, in a parallel work to ours, Cukierman and Delgrand (1995) propose a framework of time based on the notion of *calendars* which are regarded as being cyclic temporal objects, and are related to our concept of MTC. Since TUs are formally represented in a linear hierarchy, recurrent activities are dealt with *non-convex* intervals as suggested in Ladkin's approach (*op. cit.*). While granularity is obtained by decomposing all TUs into contiguous partially ordered sequences of other TUs, we take a more abstract way by defining a chain of MTCs and thus obtaining both concepts at once. TUs are related to our temporal terms using a different notation, but the set-based language for specifying TUs generates complex expressions to be read when representing concepts like *collection intervals*. They do not explore mechanisms to obtain inferences about processes working at different time scales.

What seems to be common to almost all of these approaches is that they start from linear structure, and then try to achieve granularity by imposing a hierarchy among different time intervals, and cyclicity representation by using the concepts of *convex* and *non-convex*. Although we have not started from the same point, we could interpret our time intervals in the same way, but we do not need to do that for the understanding of the principles of the theory proposed here. Furthermore, the basic assumptions of these theories do not include cyclical aspects of time in their models. Such a need has also been addressed in Pachet *et al.* (1995) for dealing with musical objects.

Finally, the style in which we present our logic is very close to proposed in Fruhwirth (1996), where temporal reasoning is treated as an application of Annotated Constraint Logic Programming. The reason is because we also view a logical formula as a classical formula annotated with a PTE. Fruhwirth's work even uses a notation for time which is similar to our smallest interval. However, there are no special mechanisms for granularity of time and circular time as we have, although it seems to be possible.

7. Conclusion and Future Work

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

ecosystems. In particular, the theory offers a simple and elegant mechanism to specify as many collections of time intervals as wanted, that is the concept of MTC.

As a specification language for simulation models which interact, the **NatureTime** logic was shown to be a powerful tool. It offers a very expressive way to define executable simulation models to be tested, mainly in the case of ecosystem domain.

Other possible branches of research from this work are:

To investigate the expansion of the logic for full resolution.
To adequate the temporal unification process for different types of temporal connectives. For instance, the present version would not deal properly with a temporal connective like "sometime".
To propose a more general proof procedure which can deal with PTE.
To extend to a multi-agent framework (Mota, 1995).

Acknowledgements

The first author would like to thank Wamberto Vasconcelos for his encouraging ideas during many "academic coffees". We would like to thank Robert Muetzelfeldt and Paulo Salles who have contributed to our understanding on ecological modelling and simulation. We also thank to Mandy Haggith for proof-reading this work and previous contribution that helped the present paper. Finally, we thank the referees who carefully read and gave valuable comments for this final version.

References

Allen, J.F., Hayes, P.J. (1985). A common sense theory of time. In *Proc. IJCAI*, pp. 528–531.
Allen, J.F. (1983). Maintaining knowledge about temporal intervals. *Comm. ACM*, **26**(11).
Allen, J.F. (1984). Towards a general theory of action and time. *Artificial Intelligence*, **23**(1).
Biggs, N.L. (1987). *Discrete Mathematics*. Oxford Science Publication.
Ciapessoni, E., Corsetti, E., Montanari, A., San Pietro, P. (1993). Embedding time granularity in a logical specification language for synchronous real-time systems. *Science of Computer Programming*, **20**(1):141–171.
Cukierman, D., Delgrand, J. (1995). A language to express time intervals and repetition. In *Proc. 2nd International Workshop on Temporal Representation and Reasoning*, Melbourne Beach, Florida, USA, April.
Dean, T. (1989). Using temoral hierarchies to efficietly maintain large temporal databases. *J. ACM*, **36**(4):687–718.
Fiadeiro, J.L., Maibaum, T. (1994). SOMETIMES "TOMOROW" IS SOMETIME action refinement in a temporal logic of objects. In *First International Conference on Temporal Logic*, pp. 48–66, Bonn, Germany, July. Springer-Verlag.
Fisher, M. (1995). Towards a semantics for concurrent metatem. In Fisher, M., Owens, R., eds, *Executable Modal and Temporal Logics*, pp. 86–102. Springer-Verlag. V(897).
Fruhwirth, T. (1996). Temporal annotated constraint logic programming. *J. Symbolic Computation* **22**(5/6):555–583.
Gabbay, D., Reynolds, M. (1995). Towards a computational treatment of time. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4 Epistemic and Temporal Reasoning. Oxford University Press.
Gabbay, D. (1989). The declarative past and imperative future. In Barringer, H., ed., *Coloquim on Temporal Logics and Specification*, pp. 409–448. Springer-Verlag. V(398).
Haggith, M., Robertson, D., Walker, D., Sinclair, F., Muetzelfeldt, R. (1992). TEAK—tools for eliciting agroforestry knowledge. In *British Computer Society Symposium of IT—Enabled Charge in Developing Countries*.
Hobbs, J.R. (1985). Granularity. In *Proc. IJCAI*, pp. 1–4.
Koomen, J.A.G.M. (1989). Reasoning about recurrence. Technical Report Technical Report 307, Department of Computer Science—University of Rochester, Rochester, USA.
Ladkin, P. (1986a). Primitives units for time specification. In *Proc. AAAI*, pp. 354–359.

Ladkin, P. (1986b). Time representation: A taxonomy of interval relations. In *Proc. AAAI*, pp. 360–366.
Ladkin, P. (1987). The completeness of a natural system for reasoning with time intervals. In *Proc. AAAI*, pp. 462–467.
Leban, B., McDonald, D.D., Foster, D.R. (1986). A representation for collections of temporal intervals. In *Proc. AAAI*, pp. 367–371.
Montanari, A. (1994). A metric and layered temporal logic for time granularity, syncrony and asyncrony. Unpublished work of the First International Conference on Temporal Logic, July.
Mota, E. (1994). Temporal representation of ecological domains. DAI TP-31, Department of Artificial Intelligence, University of Edinburgh.
Mota, E. (1995). Time granularity in simulation models within a multi-agent system. DAI Discussion Paper 158, Department of Artificial Intelligence, University of Edinburgh, April.
Mota, E., Haggith M., Smaill, A., Robertson, D. (1995a). Time granularity in simulation models of ecological systems. In *Workshop on Excutable Temporal Logics—Montreal, Canada*. DAI RP-740, Edinburgh University.
Mota, E., Robertson, D., Muezelfeldt, R. (1995b). On the granular aspects of time in simulation models. TP-39, Detartment of Artificial Intelligence/University of Edinburgh.
Pachet, F., Ramalho, G., Carrive, J., Cornic, G. (1995). Representing temporal musical objects and reasoning in the muses system. In *International Congress on Music and Artificial Intelligence*, pp. 33–48.
Le Poidevin, R., MacBeath, M., eds (1995). *The Philosophy of Time*, chapter IX, pp. 149–167. Oxford Readings in Philosophy. Oxford University Press.
Rescher, N., Urquhart, A. (1971). *Temporal Logic*. Springer-Verlag.
Robertson, D., Bundy, A., Muetzefeldt, R., Haggith, M., Uschold, M. (1991). *Eco-Logic Logic-Based Approaches to Ecological Modelling*. The MIT Press.
Sinclair, F, Robertson, D., Muetzelfeldt, R., Walker, D., Haggith, M., Kendon, G. (1993). Formal representation and use of indigenous ecological knowledge about agroforestry. ODA Forestry and Agroforestry Research Strategy—Project R4731 Second Annual Report, University of Edinburgh.
Sterling, L., Shapiro, E. (1986). *The Art of Prolog*. The MIT Press.
van Emden, M.H. (1984). An interpreting algorithm for prolog programs. In *Prolog Implementations*, Ellis Horwood Series ARTIFICIAL INTELLIGENCE, pp. 93–110. Ellis Horwood Ltd.

Appendix A. Properties of $\overset{t}{\succ}$

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 $\overset{t}{\succ}$ relation has the following properties.

The $\overset{t}{\succ}$ relation has the following properties.

transitive—if $\mathcal{C}_i^{mi}$, $\mathcal{C}_j^{mj}$ and $\mathcal{C}_k^{mk}$ are MTCs and $\mathcal{C}_k^{mk} \overset{t}{\succ} \mathcal{C}_j^{mj}$ and $\mathcal{C}_j^{mj} \overset{t}{\succ} \mathcal{C}_i^{mi}$, then $\mathcal{C}_k^{mk} \overset{t}{\succ} \mathcal{C}_i^{mi}$.

reflexive—if $\mathcal{C}_i^{mi}$ is a MTC then $\mathcal{C}_i^{mi} \overset{t}{\succ} \mathcal{C}_i^{mi}$ (every MTC can be subdivided in itself)

anti-symmetric—if $\mathcal{C}_i^{mi}, \mathcal{C}_j^{mj}$ are MTCs, and $i \neq j$, and $\mathcal{C}_j^{mj} \overset{t}{\succ} \mathcal{C}_i^{mi}$, then $\mathcal{C}_i^{mi} \overset{t}{\not\succ} \mathcal{C}_j^{mj}$.

Appendix B. Up-Wave Modular Sum and Subtraction

DEFINITION B.1. *Let $P = p(\Delta, c_i)$, where c_i defines another MTC as mod_temp_class(c_{i+1}, c_i, m), and $I = t(s_1, \ldots, s_k)$. The* up-wave modular sum *between P and I, $I \,\omega_\oplus\, P$, is defined as*

$$I \,\omega_\oplus\, P = t(s_1, \ldots, s_i + \Delta, \ldots, s_k), \text{ iff } s_i + \Delta < m$$

$I \,\omega_\oplus\, P = t(s_1, \ldots, s_i \oplus \Delta, \ldots, s_k) \,\omega_\oplus\, P'$, *iff $s_i + \Delta \geq m$, and $P' = p(\Delta', c_{i+1})$, where $\Delta' = s_i + \Delta$ div m.*

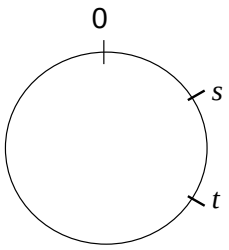

Figure 3. Time interval in a modular temporal structure.

Figure 4. Correspondent interval at any arbitrary interval in the infinte line.

DEFINITION B.2. *Let $P = p(\Delta, c_i)$, and c_i defines another MTC as mod_temp_class(c_{i+1}, c_i, m), and $I = t(s_1, \ldots, s_k)$. We call the* up-wave modular subtraction *between P and I, $I \,\omega_{\ominus}\, P$, defined as*

$$I \,\omega_{\ominus}\, P = t(s_1, \ldots, s_i - \Delta, \ldots, s_k), \text{ iff } \Delta < s_i$$

$I \,\omega_{\ominus}\, P = t(s_1, \ldots, s_i \ominus \Delta, \ldots, s_k) \,\omega_{\ominus}\, P'$, *iff* $\Delta \geq s_i$ *and* $P' = p(\Delta', c_{i+1})$, *where* $\Delta' = \Delta$ *div* m.

B.1. DIAMETER FUNCTION

DEFINITION B.3. *Let S be an interval in a Linear-Cyclic hierarchy of time, P a period of time. We call* diameter of time*, written $\oslash$, to the function which maps S to P. More generally, $\oslash : \mathcal{E} \rightarrow \mathcal{P}$, where $\mathcal{E}$ is the set of all temporal entities and $\mathcal{P}$ the set of all periods, or length of time.*

B.2. LINEAR REALIZABILITY OF A CIRCULAR TIME STRUCTURE

This section presents what was called *Linear Realizability* in Rescher and Urquhart (1971) in which it was established a relationship between a course of history in a circular structure of time and the same course in a linear time. Consider a temporal structure which is one-dimension, finite and closed $\mathcal{C}^m$, where m is the number of elements in the circular set C. Then any of the possible courses of history realized in C can also be realized on the line. We can see this if we consider the arbitrary interval $s \ldots t$ in the circle C^m as shown in Figure 3.

By putting it in correspondence with an arbitrary interval (with the same size), in the infinite line, as shown in Figure 4.

Now, at any distance m forwards from s and t in the circular structure C^m in correspondence the forwards oints at the same distance from the linear interval $s \ldots t$; and analogously backwards. Thus we put the circle into correspondence over and over again with an *equally long* segment on the line as depicted in Figure 5.

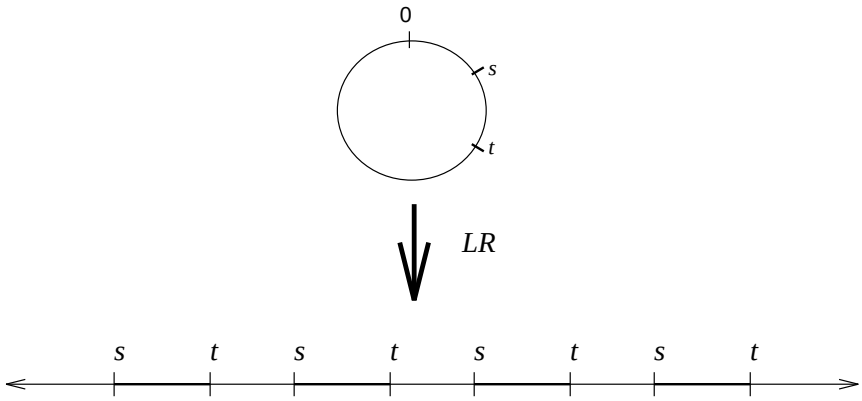

Figure 5. Linear equivalent intervals of the circular interval.

Appendix C. Matching Between Cyclical Intervals

DEFINITION C.1. *Given the cyclical intervals* $S = i(s_1 \dots s_2, c_i)$ *and* $T = i(t_1 \dots t_2, c_i)$ *of a MTC* c_i*. Then, the matching between these intervals results in* $i(t_1 \dots t_2, c_i)$ *in the case that one of the following holds.*

$s_2 \geq t_2$ *and* $t_2 > t_1$ *and* $t_1 \geq s_1$*, or*
$t_2 > t_1$ *and* $t_1 \geq s_1$ *and* $s_1 > s_2$*, or*
$s_1 > s_2$ *and* $s_2 \geq t_2$ *and* $t_2 > t_1$*, or*
$t_1 \geq s_1$ *and* $s_1 > s_2$ *and* $s_2 \geq t_2$*.*

In this case we say that T *is* included *in (or is during)* S*.*

DEFINITION C.2. *Given the cyclical intervals* $S = i(s_1 \dots s_2, c_i)$ *and* $T = i(t_1 \dots t_2, c_i)$ *of a MTC* c_i*. Then, the matching between these intervals results in*

$i(t_1 \dots s_2, c_i)$ *if one of the following holds*

$t_2 > s_2$ *and* $s_2 \geq t_1$ *and* $t_1 > s_1$*, or*
$s_2 \geq t_1$ *and* $t_1 \geq s_1$ *and* $s_1 > t_2$*, or*
$t_1 > s_1$ *and* $s_1 > t_2$ *and* $t_2 > s_2$*, or*
$s_1 > t_2$ *and* $t_2 \geq s_2$ *and* $s_2 \geq t_1$

$i(t_1 \dots s_2, c_i)$ *and also* $i(s_1 \dots t_2, c_i)$ *if one of the following holds*

$t_2 \geq s_1$ *and* $s_1 > s_2$ *and* $s_2 \geq t_1$*, or*
$t_1 > t_2$ *and* $t_2 \geq s_1$ *and* $s_1 > s_2$*.*

In both cases S *and* T *overlap.*

Appendix D. KB Definition for the Example 1

D.1. NEIGHBOURS DEFINITION

$neighbours(t1, [(t2, 4.24), (t3, 4.24), (t5, 6)])$.

$neighbours(t2, [(t1, 4.24), (t3, 6), (t4, 4.24), (t5, 4.24)]).$
$neighbours(t3, [(t1, 4.24), (t2, 6), (t5, 4.24), (t6, 4.24), (t7, 6)]).$
$neighbours(t4, [(t2, 4.24), (t5, 6), (t8, 4.24), (t9, 3)]).$
$neighbours(t5, [(t1, 6), (t2, 4.24), (t3, 4.24), (t4, 6), (t6, 6), (t7, 4.24), (t8, 4.24)]).$
$neighbours(t6, [(t3, 4.24), (t5, 6), (t7, 4.24)]).$
$neighbours(t7, [(t3, 6), (t5, 4.24), (t6, 4.24), (t8, 6)]).$
$neighbours(t8, [(t2, 6), (t4, 4.24), (t5, 4.24), (t7, 6), (t9, 3), (t10, 3)]).$
$neighbours(t9, [(t4, 3), (t8, 3), (t10, 2.24)]).$
$neighbours(t10, [(t8, 3), (t9, 4.24)]).$

D.2. initial and maximum height, and growth rate of each tree

$height(t1, 1) @ t(14, 2, 1994).$	$max\_height(t1, 7).$	$growth\_rate(t1, 0.01).$
$height(t2, 1) @ t(14, 2, 1994).$	$max\_height(t2, 10).$	$growth\_rate(t2, 0.012).$
$height(t3, 1) @ t(14, 2, 1994).$	$max\_height(t3, 12).$	$growth\_rate(t3, 0.015).$
$height(t4, 1) @ t(14, 2, 1994).$	$max\_height(t4, 5).$	$growth\_rate(t4, 0.0009).$
$height(t5, 1) @ t(14, 2, 1994).$	$max\_height(t5, 5).$	$growth\_rate(t5, 0.008).$
$height(t6, 1) @ t(14, 2, 1994).$	$max\_height(t6, 7).$	$growth\_rate(t6, 0.011).$
$height(t7, 1) @ t(14, 2, 1994).$	$max\_height(t7, 12).$	$growth\_rate(t7, 0.015).$
$height(t8, 1) @ t(14, 2, 1994).$	$max\_height(t8, 6).$	$growth\_rate(t8, 0.01).$
$height(t9, 1) @ t(14, 2, 1994).$	$max\_height(t9, 4).$	$growth\_rate(t9, 0.006).$
$height(t10, 1) @ t(14, 2, 1994).$	$max\_height(t10, 13).$	$growth\_rate(t10, 0.018).$

D.3. influence of other trees in one tree

$influences([], \_, []).$
$influences([Y|T], Time, [R1|TR]) :-$
  $ind\_influence(Y, Time, R1),$
  $influences(T, Time, TR).$
$ind\_influence((Tree, D), T, I) :-$
  $solve(height(Tree, H) @ T, \_),$
  $(I \; is \; H/(D * 1000)), !.$

Appendix E. KB Definition for the Example 2

E.1. progress and influence specification

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, is represented in a list composed with the values for *Att* during T. This list is computed as defined in the meta-language as follows.

$progress(value(Att, Obj, V)@T, P, [V|R]) :-$
  $change(Att, Proc),$
  $scale(time, Obj, Proc, C),$
  $solve(value(Att, Obj, V) @ T, \_),$
  $future(T, P, Tf),$
  $progression(value(Att, Obj, V) @ T, Tf, C, R).$
$progression(\_ @ T, Tf, C, []) :-$
  $next(T, C, Tf).$
$progression(value(Att, Obj, Vi) @ Ti, Tf, C, [Vj|R]) :-$
  $\neg next(Ti, C, Tf),$
  $value(Att, Obj, Vj) @ p(1, C) \; after \; Ti \Longleftarrow value(Att, Obj, Vi) @ Ti \; \& \; Constraints,$
  $solve(Constraints, \_),$
  $next(Ti, C, Tj),$

$progression(value(Att, Obj, Vj) @ Tj, Tf, C, R).$

$influence(GR, [], GR).$
$influence(GR, [Pos|R], RealGR) :-$
$Pos > 7,$
$NGR\ is\ GR - 0.02,$
$influence(NGR, R, RealGR).$
$influence(GR, [Pos|R], RealGR) :-$
$Pos \leq 7,$
$influence(GR, R, RealGR).$

E.2. BUG'S POSITION

$new\_d(Pos1, M, H, D1, D2) :-$
$Pos2\ is\ Pos1 + M,$
$Pos2 \leq H,$
$D2\ is\ D1 * (-1).$
$new\_d(Pos1, M, H, D1, D2) :-$
$Pos2\ is\ Pos1 + M,$
$Pos2 > H,$
$D2\ is\ D1 * (-1).$