Integrating Agent Models and Dynamical Systems

Tibor Bosse, Alexei Sharpanskykh, and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV, The Netherlands
{tbosse, sharp, treur}@cs.vu.nl
http://www.cs.vu.nl/~{tbosse, sharp, treur}

Abstract. Agent-based modelling approaches are usually based on logical languages, whereas in many areas dynamical system models based on differential equations are used. This paper shows how to model complex agent systems, integrating quantitative, numerical and qualitative, logical aspects, and how to combine logical and mathematical analysis methods.

1 Introduction

Existing models for complex systems are often based on quantitative, numerical methods such as Dynamical Systems Theory (DST) [23], and more in particular, differential equations. Such approaches often use numerical variables to describe global aspects of the system and how they affect each other over time; for example, how the number of predators affects the number of preys. An advantage of such numerical approaches is that numerical approximation methods and software environments are available for simulation.

The relatively new agent-based modelling approaches to complex systems take into account the local perspective of a possibly large number of separate agents and their specific behaviours in a system; for example, the different individual predator agents and prey agents. These approaches are usually based on qualitative, logical languages. An advantage of such logical approaches is that they allow (automated) logical analysis of the relationships between different parts of a model, for example relationships between global properties of the (multi-agent) system as a whole and local properties of the basic mechanisms within (agents of) the system. Moreover, by means of logic-based approaches, declarative models of complex systems can be specified using knowledge representation languages that are close to the natural language. An advantage of such declarative models is that they can be considered and analysed at a high abstract level. Furthermore, automated support (e.g., programming tools) is provided for manipulation and redesign of models.

Complex systems, for example organisms in biology or organisations in the socio-economic area, often involve both qualitative aspects and quantitative aspects. In particular, in the area of Cognitive Science, the lower-level cognitive processes of agents (e.g., sensory or motor processing) are often modelled using DST-based approaches. Furthermore, at the global level the dynamics of the environment, in which agents are situated, is often described by continuous models (i.e., models based

on differential equations); e.g., dynamic models of markets, or natural environmental oscillations. Yet agent-based (logical) languages are often used for describing high-level cognitive processes of agents (e.g., processes related to reasoning) and agent interaction with the environment (e.g., agent actions, execution of tasks).

It is not easy to integrate both types of approaches in one modelling method. On the one hand, it is difficult to incorporate logical aspects in differential equations. For example, qualitative behaviour of an agent that depends on whether the value of a variable is below or above a threshold is difficult to describe by differential equations. On the other hand, quantitative methods based on differential equations are not usable in the context of most logical, agent-based modelling languages, as these languages are not able to handle real numbers and calculations.

This paper shows an integrative approach to simulate and analyse complex systems, integrating quantitative, numerical and qualitative, logical aspects within one expressive temporal specification language. Some initial ideas behind the simulation approach proposed in this paper were described in [6]. The current paper elaborates upon these ideas by proposing more extensive means to design precise, stable, and computationally effective simulation models for hybrid systems (i.e., comprising both quantitative and qualitative aspects). Furthermore, it proposes techniques for analysis of hybrid systems, which were not previously considered elsewhere. The developed simulation and analysis techniques are supported by dedicated tools.

In Section 2, this language (called LEADSTO) is described in detail, and is applied to solve an example differential equation. In Section 3, it is shown how LEADSTO can solve a system of differential equations (for the case of the classical Predator-Prey model), and how it can combine quantitative and qualitative aspects within the same model. Section 4 demonstrates how existing methods for approximation (such as the Runge-Kutta methods) can be incorporated into LEADSTO, and Section 5 shows how existing methods for simulation with dynamic step size can be incorporated. Section 6 demonstrates how interlevel relationships can be established between dynamics of basic mechanisms (described in LEADSTO) and global dynamics of a process (described in a super-language of LEADSTO). Finally, Section 7 is a discussion.

2 Modelling Dynamics in LEADSTO

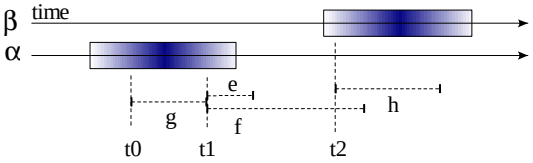
Dynamics can be modelled in different forms. Based on the area within Mathematics called calculus, the Dynamical Systems Theory [23] advocates to model dynamics by continuous state variables and changes of their values over time, which is also assumed continuous. In particular, systems of differential or difference equations are used. This may work well in applications where the world states are modelled in a quantitative manner by real-valued state variables. The world's dynamics in such application show continuous changes in these state variables that can be modelled by mathematical relationships between real-valued variables. However, not for all applications dynamics can be modelled in a quantitative manner as required for DST. Sometimes qualitative changes form an essential aspect of the dynamics of a process. For example, to model the dynamics of reasoning processes usually a quantitative approach will not work. In such processes states are characterised by qualitative state

properties, and changes by transitions between such states. For such applications often qualitative, discrete modelling approaches are advocated, such as variants of modal temporal logic, e.g. [20]. However, using such non-quantitative methods, the more precise timing relations are lost too. For the LEADSTO language described in this paper, the choice has been made to consider the timeline as continuous, described by real values, but for state properties both quantitative and qualitative variants can be used. The approach subsumes approaches based on simulation of differential or difference equations, and discrete qualitative modelling approaches. In addition, the approach makes it possible to combines both types of modelling within one model. For example, it is possible to model the exact (real-valued) time interval for which some qualitative property holds. Moreover, the relationships between states over time are described by either logical or mathematical means, or a combination thereof. This will be explained in more detail in Section 2.1. As an illustration, in Section 2.2 it will be shown how the logistic model for population growth in resource-bounded environments [4] can be modelled and simulated in LEADSTO.

2.1 The LEADSTO Language

Dynamics is considered as evolution of states over time. The notion of state as used here is characterised on the basis of an ontology defining a set of properties that do or do not hold at a certain point in time. For a given (order-sorted predicate logic) ontology Ont, the propositional language signature consisting of all *state ground atoms* (or *atomic state properties*) based on Ont is denoted by APROP(Ont). The *state properties* based on a certain ontology Ont are formalised by the propositions that can be made (using conjunction, negation, disjunction, implication) from the ground atoms. A *state* S is an indication of which atomic state properties are true and which are false, i.e., a mapping S: APROP(Ont) $\rightarrow$ {true, false}.

To specify simulation models a temporal language has been developed. This language (the LEADSTO language [7]) enables to model direct temporal dependencies between two state properties in successive states, also called *dynamic properties*. A specification of dynamic properties in LEADSTO format has as advantages that it is executable and that it can often easily be depicted graphically. The format is defined as follows. Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h non-negative real numbers. In the LEADSTO language the notation $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$ (also see Fig. 1), means:



If state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h.

Fig. 1. Timing relationships for LEADSTO expressions.

An example dynamic property that uses the LEADSTO format defined above is the following: "observes(agent_A, food_present) $\rightarrow\!\!\!\!\rightarrow_{2, 3, 1, 1.5}$ beliefs(agent_A, food_present)".

Informally, this example expresses the fact that, if agent A observes that food is present during 1 time unit, then after a delay between 2 and 3 time units, agent A will belief that food is present during 1.5 time units. In addition, within the LEADSTO language it is possible to use sorts, variables over sorts, real numbers, and mathematical operations, such as in "has_value(x, v) $\rightarrow_{e, f, g, h}$ has_value(x, v*0.25)". Next, a *trace* or *trajectory* γ over a state ontology Ont is a time-indexed sequence of states over Ont (where the time frame is formalised by the real numbers). A LEADSTO expression $\alpha \rightarrow\!\!\!\rightarrow_{e, f, g, h} \beta$, holds for a trace γ if:

$\forall t1\ [\forall t\ [t1-g \leq t < t1 \Rightarrow \alpha \text{ holds in } \gamma \text{ at time } t\ ] \Rightarrow \exists d\ [e \leq d \leq f\ \&\ \forall t'\ [t1+d \leq t' < t1+d+h \Rightarrow \beta \text{ holds in } \gamma \text{ at time } t'\ ]$

To specify the fact that a certain event (i.e., a state property) holds at every state (time point) within a certain time interval a predicate holds_during_interval(event, t1, t2) is introduced. Here event is some state property, t1 is the beginning of the interval and t2 is the end of the interval.

An important use of the LEADSTO language is as a specification language for simulation models. As indicated above, on the one hand LEADSTO expressions can be considered as logical expressions with a declarative, temporal semantics, showing what it means that they hold in a given trace. On the other hand they can be used to specify basic mechanisms of a process and to generate traces, similar to Executable Temporal Logic [3]. More details on the semantics of the LEADSTO language can be found in [7].

2.2 Solving the Initial Value Problem in LEADSTO: Euler's method

Often behavioural models in the Dynamical Systems Theory are specified by systems of differential equations with given initial conditions for continuous variables and functions. A problem of finding solutions to such equations is known as an initial value problem in the mathematical analysis. One of the approaches for solving this problem is based on discretisation, i.e., replacing a continuous problem by a discrete one, whose solution is known to approximate that of the continuous problem. For this methods of numerical analysis are usually used [22]. The simplest approach for finding approximations of functional solutions for ordinary differential equations is provided by Euler's method. Euler's method for solving a differential equation of the form $dy/dt = f(y)$ with the initial condition $y(t_0)=y_0$ comprises the difference equation derived from a Taylor series:

$$y(t) = \sum_{n=0}^{\infty} \frac{y^{(n)}(t_0)}{n!} * (t - t_0)^n ,$$

where only the first member is taken into account: $y_{i+1}=y_i+h^* f(y_i)$, where $i \geq 0$ is the step number and $h>0$ is the integration step size. This equation can be modelled in the LEADSTO language in the following way:

- Each integration step corresponds to a state, in which an intermediate value of y is calculated.
- The difference equation is modelled by a transition rule to the successive state in the LEADSTO format.
- The duration of an interval between states is defined by a step size h.

Thus, for the considered case the LEADSTO simulation model comprises the rule:

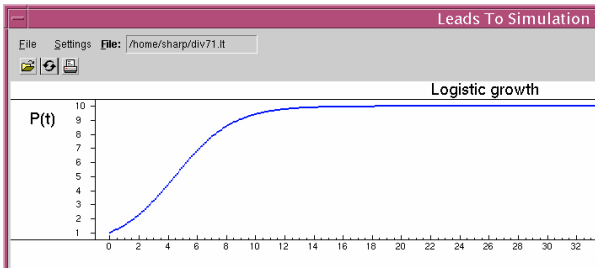
has_value(y, v1) $\rightarrow\!\!\!\!\rightarrow_{0,\,0,\,h,\,h}$ has_value(y, v1+h* f(v1))

The initial value for the function y is specified by the following LEADSTO rule:

holds_during_interval(has_value(y, y_0), 0, h)

By performing a simulation of the obtained model in the LEADSTO environment an approximate functional solution to the differential equation can be found.

To illustrate the proposed simulation-based approach based on Euler's method in LEADSTO, the logistic growth model or the Verhulst model [4] which is often used to describe the population growth in resource-bounded environments, is considered: dP/dt = r*P(1-P/K), where P is the population size at time point t; r and K are some constants. This model corresponds to the following LEADSTO simulation model: has_value(y, v1) $\rightarrow\!\!\!\!\rightarrow_{0,\,0,\,h,\,h}$ has_value(y, v1+ h*r* v1*(1-v1/K)).



The result of simulation of this model in the LEADSTO environment with the parameters r=0.5 and K=10 and the initial value P(0)=1 is given in Figure 2.

Fig. 2. Logistic growth function modelled in LEADSTO with parameters r=0.5, K=10, P(0)=1.

3 Modelling the Predator-Prey Model in LEADSTO

The proposed simulation-based approach can be applied for solving a system of ordinary differential equations. In order to illustrate this, the classical Lotka-Volterra model (also known as a Predator-Prey model) [21] is considered. The Lotka-Volterra model describes interactions between two species in an ecosystem, a predator and a prey. The model consists of two equations: the first one describes how the prey population changes and the second one describes how the predator population changes. If x(t) and y(t) represent the number of preys and predators respectively, that are alive in the system at time t, then the Lotka-Volterra model is defined by: dx/dt = a*x - b*x*y ; dy/dt = c*b*x*y - e*y where the parameters are defined by: a is the per capita birth rate of the prey, b is a per capita attack rate, c is the conversion efficiency of consumed prey into new predators, and e is the rate at which predators die in the absence of prey. To solve this system, numerical methods derived from a Taylor series up to some order can be used. In the following section it will be shown how Euler's (first-order rough) method can be used for creating a LEADSTO simulation model for finding the approximate solutions for the Predator-Prey problem. After that, in Section 3.2 it will be demonstrated how the generated LEADSTO simulation model can be extended by introducing qualitative behavioural aspects in the standard predator-prey model.

3.1 The LEADSTO language

Using the technique described in Section 2.2, the Lotka-Volterra model is translated into a LEADSTO simulation model as follows:

has_value(x, v1) ∧ has_value(y, v2) $\rightarrow_{0, 0, h, h}$ has_value(x, v1+h*(a*v1-b*v1*v2))
has_value(x, v1) ∧ has_value(y, v2) $\rightarrow_{0, 0, h, h}$ has_value(y, v2+h*(c*b*v1*v2-e*v2))

The initial values for variables and functions are specified as for the general case. Although Euler's method offers a stable solution to a stable initial value problem, a choice of initial values can significantly influence the model's behaviour. More specifically, the population size of both species will oscillate if perturbed away from the equilibrium. The amplitude of the oscillation depends on how far the initial values of x and y depart from the equilibrium point. The equilibrium point for the considered model is defined by the values x=e/(c*b) and y=a/b. For example, for the parameter settings a=1.5, b=0.2, c=0.1 and e=0.5 the equilibrium is defined by x=25 and y=7.5. Yet a slight deviation from the equilibrium point in the initial values (x_0=25, y_0=8) results in the oscillated (limit cycle) behaviour.

3.2 Extending the Standard Predator-Prey Model with Qualitative Aspects

In this section, an extension of the standard predator-prey model is considered, with some qualitative aspects of behaviour. Assume that the population size of both predators and preys within a certain eco-system is externally monitored and controlled by humans. Furthermore, both prey and predator species in this eco-system are also consumed by humans. A control policy comprises a number of intervention rules that ensure the viability of both species. Among such rules could be following:

- in order to keep a prey species from extinction, a number of predators should be controlled to stay within a certain range (defined by pred_min and pred_max);
- if a number of a prey species falls below a fixed minimum (prey_min), a number of predators should be also enforced to the prescribed minimum (pred_min);
- if the size of the prey population is greater than a certain prescribed bound (prey_max), then the size of the prey species can be reduced by a certain number prey_quota (cf. a quota for fish catch).

These qualitative rules can be encoded into the LEADSTO simulation model for the standard predator-prey case by adding new dynamic properties and changing the existing ones in the following way:

has_value(x, v1) ∧ has_value(y, v2) ∧ v1< prey_max $\rightarrow_{0, 0, h, h}$ has_value(x, v1+h*(a*v1-b*v1*v2))
has_value(x, v1) ∧ has_value(y, v2) ∧ v1 ≥ prey_max $\rightarrow_{0, 0, h, h}$
has_value(x, v1+h*(a*v1-b*v1*v2) - prey_quota)
has_value(x, v1) ∧ has_value(y, v2) ∧ v1 ≥ prey_min ∧ v2 < pred_max $\rightarrow_{0, 0, h, h}$
has_value(y, v2+h* (c*b*v1*v2-e*v2))
has_value(x, v1) ∧ has_value(y, v2) ∧ v2 ≥ pred_max $\rightarrow_{0, 0, h, h}$ has_value(y, pred_min)
has_value(x, v1) ∧ has_value(y, v2) ∧ v1 < prey_min $\rightarrow_{0, 0, h, h}$ has_value(y, pred_min)

The result of simulation of this model using Euler's method with the parameter settings: a=4; b=0.2, c=0.1, e=8, pred_min=10, pred_max=30, prey_min=40, prey_max=100, prey_quota=20, x_0=90, y_0=10 is given in Fig. 3. More examples of the LEADSTO simulation models combining quantitative and qualitative aspects of behaviour can be

found in [5] and [6]. In [6], a LEADSTO model for classical conditioning is presented, based on Machado's differential equation model [18]. This model integrates quantitative aspects such as levels of preparation with qualitative aspects such as the occurrences of certain stimuli. In [5], a LEADSTO model for eating regulation disorders is presented. This model integrates quantitative aspects such as a person's weight with qualitative aspects such as the decision to eat.

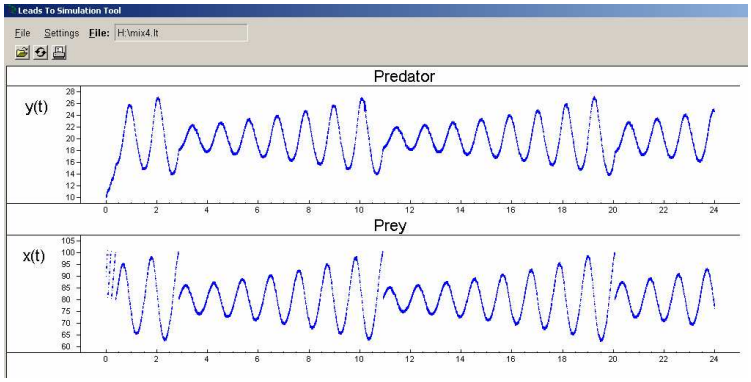


Fig. 3. Simulation results for the Lotka-Volterra model combined some qualitative aspects.

4 Simulating the Predator-Prey Model by the Runge-Kutta Method

As shown in [22], within Euler's method the local error at each step (of size h) is $O(h^2)$, while the accumulated error is $O(h)$. However, the accumulated error grows exponentially as the integration step size increases. Therefore, in situations in which precision of a solution is required, high order numerical methods are used. For the purpose of illustration of high-order numerical approaches the fourth-order Runge-Kutta method is considered. This method is derived from a Taylor expansion up to the fourth order. It is known to be very accurate (the accumulated error is $O(h^4)$) and stable for a wide range of problems. The Runge-Kutta method for solving a differential equation of the form $dx/dt = f(t, x)$ is described by the following formulae:

$x_{i+1} = x_i + h/6 *(k_1 + 2*k_2 + 2*k_3 + k_4)$,

where $i \geq 0$ is the step number, $h>0$ is the integration step size, and

$k_1 = f(t_i, x_i)$, $k_2 = f(t_i + h/2, x_i + h/2 *k_1)$, $k_3 = f(t_i + h/2, x_i + h/2 *k_2)$, $k_4 = f(t_i + h, x_i + h* k_3)$.

Now, using the Runge-Kutta method, the classical Lotka-Volterra model considered in the previous section is described in the LEADSTO format as follows:

has_value(x, v1) $\wedge$ has_value(y, v2) $\rightarrow\!\!\!\!\rightarrow_{0, 0, h, h}$ has_value(x, v1 + h/6 *(k_{11} + 2*k_{12} + 2*k_{13} + k_{14}))
has_value(x, v1) $\wedge$ has_value(y, v2) $\rightarrow\!\!\!\!\rightarrow_{0, 0, h, h}$ has_value(y, v2 + h/6 *(k_{21} + 2*k_{22} + 2*k_{23} + k_{24})),

where:

k_{11} = a*v1-b*v1*v2, k_{21} = c*b*v1*v2 - e*v2, k_{12} = a*(v1 + h/2 *k_{11}) - b*(v1 + h/2 *k_{11})*(v2 + h/2 *k_{21}), k_{22} = c*b*(v1 + h/2 *k_{11})*(v2 + h/2 *k_{21}) - e*(v2 + h/2 *k_{21}), k_{13} = a*(v1 + h/2 *k_{12}) - b*(v1 + h/2 *k_{12})*(v2 + h/2 *k_{22}), k_{23} = c*b*(v1 + h/2 *k_{12})*(v2 + h/2 *k_{22}) - e*(v2 + h/2 *k_{22}), k_{14} = a*(v1 + h *k_{13}) - b*(v1 + h *k_{13})*(v2 + h *k_{23}), k_{24} = c*b*(v1 + h *k_{13})*(v2 + h *k_{23}) - e*(v2 + h *k_{23}).

5 Simulation with Dynamic Step Size

Although for most cases the Runge-Kutta method with a small step size provides accurate approximations of required functions, this method can still be computationally expensive and, in some cases, inaccurate. In order to achieve a higher accuracy together with minimum computational efforts, methods that allow the dynamic (adaptive) regulation of an integration step size are used. This section shows how such methods can be incorporated in LEADSTO.

To illustrate the use of methods for dynamic step size control, the biochemical model of [13], summarised in Table 1, is considered.

Table. 1. Glycolysis model by [13].

Variables	Moiety conservation	Rate equations
W: Fructose 6-phosphate X : phosphoenolpyruvate Y : pyruvate N1 : ATP; N2 : ADP; N3 : AMP Differential equations X'[t] == 2*Vpfk - Vxy Y'[t] == Vxy - Vpdc N1'[t] == Vxy + Vak - Vatpase N2'[t] == -Vxy - 2*Vak + Vatpase	N1[t] + N2[t] + N3 = 20 Initial conditions N1[0] == 10 N2[0] == 9 Y[0] == 0 X[0] == 0 Fixed metabolites W = 0.0001; Z = 0	Vxy = 343*N2[t]*X[t]/((0.17 + N2[t])*(0.2 + X[t])) Vak = -(432.9*N3*N1[t] - 133*N2[t]^2) Vatpase = 3.2076*N1[t] Vpdc = 53.1328*Y[t]/(0.3 + Y[t]) (*10.0*Y[t]*) Vpfk = 45.4327*W^2/(0.021*(1 + 0.15*N1[t]^2/N3^2 + W^2))

This model describes the process of glycolysis in *Saccharomyces cerevisiae*, a specific species of yeast. This model is interesting to study, because the concentrations of some of the substances involved (in particular ATP and ADP) are changing at a variable rate: sometimes these concentrations change rapidly, and sometimes they change very slowly. Using the technique described in Section 2.2 (based on Euler's method), this model can be translated to the following LEADSTO simulation model:

has_value(x, v1) ∧ has_value(y, v2) ∧ has_value(n1, v3) ∧ has_value(n2, v4) →$_{0, 0, h, h}$
has_value(x, v1+ (2* (45.4327*w^2/ (0.021* (1+0.15*v3^2/ (20-v3-v4)^2+w^2)))-343*v4*v1/ ((0.17+v4)* (0.2+v1)))*h)

has_value(x, v1) ∧ has_value(y, v2) ∧ has_value(n1, v3) ∧ has_value(n2, v4) →$_{0, 0, h, h}$
has_value(y, v2+ (343*v4*v1/ ((0.17+v4)* (0.2+v1))-53.1328*v2/ (0.3+v2))*h)

has_value(x, v1) ∧ has_value(y, v2) ∧ has_value(n1, v3) ∧ has_value(n2, v4) →$_{0, 0, h, h}$
has_value(n1, v3+ (343*v4*v1/ ((0.17+v4)* (0.2+v1))+ (- (432.9* (20-v3-v4)*v3-133*v4^2))-3.2076*v3)*h)

has_value(x, v1) ∧ has_value(y, v2) ∧ has_value(n1, v3) ∧ has_value(n2, v4) →$_{0, 0, h, h}$
has_value(n2, v4+ (-343*v4*v1/ ((0.17+v4)* (0.2+v1))-2* (- (432.9* (20-v3-v4)*v3-133*v4^2))+3.2076*v3)*h)

The simulation results of this model (with a static step size of 0.00001) are shown in Fig. 4. Here the curves for N1 and N2 are initially very steep, but become flat after a while. As demonstrated by Figure 4, for the first part of the simulation, it is necessary to pick a small step size in order to obtain accurate results. However, to reduce computational efforts, for the second part a bigger step size is desirable. To this end, a number of methods exist that allow the dynamic adaptation of the step size in a simulation. Generally, these approaches are based on the fact that the algorithm

signals information about its own truncation error. The most straightforward (and most often used) technique for this is *step doubling* and *step halving*, see, e.g. [Gear 1971]. The idea of step doubling is that, whenever a new simulation step should be performed, the algorithm compares the result of applying the current step twice with the result of applying the double step (i.e., the current step * 2) once. If the difference between both solutions is smaller than a certain threshold ε, then the double step is selected. Otherwise, the algorithm determines whether step halving can be applied: it compares the result of applying the current step once with the result of applying the half step (i.e., the current step * 0.5) twice. If the difference between both solutions is smaller than ε, then the current step is selected. Otherwise, the half step is selected.

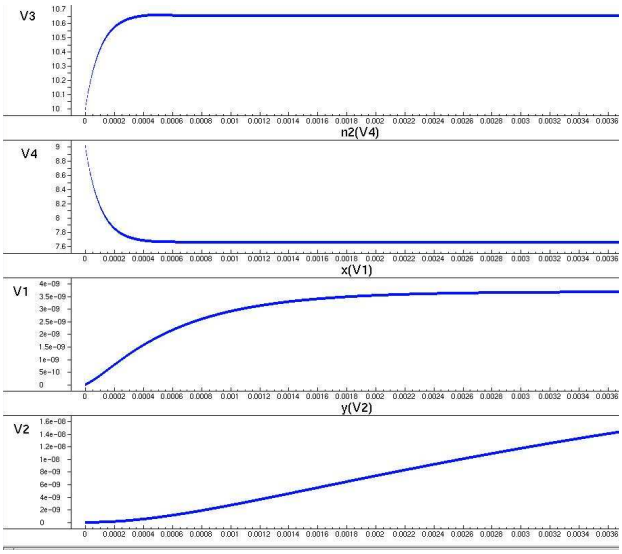


Fig. 4. Simulation results of applying Euler's method to [13]'s glycolysis model.

Since its format allows the modeller to include qualitative aspects, it is not difficult to incorporate step doubling and step halving into LEADSTO. To illustrate this, consider the general LEADSTO rule shown in Section 2.2 for solving a differential equation of the form dy/dt = f(y) using Euler's method:

has_value(y, v1) $\rightarrow_{0, 0, h, h}$ has_value(y, v1+h* f(v1))

Adding step doubling and step halving to this rule yields the following three rules:

step(h) ∧ has_value(y, v1) ∧ |(v1+2h* f(v1)) - ((v1+h* f(v1))+h* f(v1+h* f(v1)))| ≤ ε
$\rightarrow_{0, 0, 2h, 2h}$ has_value(y, v1+2h* f(v1)) ∧ step(2h)

step(h) ∧ has_value(y, v1) ∧ |(v1+2h* f(v1)) - ((v1+h* f(v1))+h* f(v1+h* f(v1)))| > ε ∧
|(v1+h* f(v1)) - ((v1+0.5h* f(v1))+0.5h* f(v1+0.5h* f(v1)))| ≤ ε
$\rightarrow_{0, 0, h, h}$ has_value(y, v1+h* f(v1)) ∧ step(h)

step(h) ∧ has_value(y, v1) ∧ |(v1+h* f(v1)) - ((v1+0.5h* f(v1))+0.5h* f(v1+0.5h* f(v1)))| ≤ ε
$\rightarrow_{0, 0, 0.5h, 0.5h}$ has_value(y, v1+0.5h* f(v1)) ∧ step(0.5h)

Besides step doubling, many other techniques exist in the literature for dynamically controlling the step size in quantitative simulations. Among these are several

techniques that are especially aimed at the Runge-Kutta methods, see, e.g., [24], Chapter 16 for an overview. Although it is possible to incorporate such techniques into LEADSTO, they are not addressed here because of space limitations.

6 Analysis In Terms of Local-Global Relations

Within the area of agent-based modelling, one of the means to address complexity is by modelling processes at different levels, from the global level of the process as a whole, to the local level of basic elements and their mechanisms. At each of these levels dynamic properties can be specified, and by interlevel relations they can be logically related to each other; e.g., [14], [27]. These relationships can provide an explanation of properties of a process as a whole in terms of properties of its local elements and mechanisms. Such analyses can be done by hand, but also software tools are available to automatically verify the dynamic properties and their interlevel relations. To specify the dynamic properties at different levels and their interlevel relations, a more expressive language is needed than simulation languages based on causal relationships, such as LEADSTO. The reason for this is that, although the latter types of languages are well suited to express the basic mechanisms of a process, for specifying global properties of a process it is often necessary to formulate complex relationships between states at different time points. To this end, the formal language TTL has been introduced as a super-language of LEADSTO; cf. [8]. It is based on order-sorted predicate logic and, therefore, inherits the standard semantics of this variant of predicate logic. That is, the semantics of TTL is defined in a standard way, by interpretation of sorts, constants, functions and predicates, and variable assignments. Furthermore, TTL allows representing numbers and arithmetical functions. Therefore, most methods used in Calculus are expressible in TTL, including methods based on derivatives and differential equations. In this section, first (in Section 6.1) it is shown how to incorporate differential equations in the predicate-logical language TTL that is used for analysis. Next, in Section 6.2 a number of global dynamic properties are identified, and it is shown how they can be expressed in TTL. In Section 6.3 a number of local dynamic properties are identified and expressed in TTL. Finally, Section 6.4 discusses how the global properties can be logically related to local properties such that a local property implies the global property.

6.1 The LEADSTO language

As mentioned earlier, traditionally, analysis of dynamical systems is often performed using mathematical techniques such as the Dynamical Systems Theory. The question may arise whether or not such modelling techniques can be expressed in the Temporal Trace Language TTL. In this section it is shown how modelling techniques used in the Dynamical Systems approach, such as difference and differential equations, can be represented in TTL. First the discrete case is considered. As an example consider again the logistic growth model: $dP/dt = r*P(1-P/K)$. This equation can be expressed in TTL on the basis of a discrete time frame (e.g., the natural numbers) in a straightforward manner:

$\forall t\ \forall v$ state(γ , t) $\models$ has_value(P, v) $\Rightarrow$ state(γ , t+1) $\models$ has_value(P, v + h • r • v • (1 - v/K))

The traces γ satisfying the above dynamic property are the solutions of the difference equation. However, it is also possible to use the dense time frame of the real numbers, and to express the differential equation directly. To this end, the following relation is introduced, expressing that x = dy/dt:

is_diff_of(γ, x, y) :
$\forall t,w\ \forall \varepsilon>0\ \exists \delta>0\ \forall t',v,v'$ [0 < dist(t',t) < δ & state(γ, t) $\models$ has_value(x, w) &
state(γ, t) $\models$ has_value(y, v) & state(γ, t') $\models$ has_value(y, v') $\Rightarrow$ dist((v'-v)/(t'-t),w) < ε]

where γ is the trace that describes the change of values of x and y over time, dist(u,v) is defined as the absolute value of the difference, i.e. u-v if this is ≥ 0, and v-u otherwise. Using this, the differential equation can be expressed by is_diff_of(γ , r • P (1 - P/K), P).

The traces γ for which this statement is true are (or include) solutions for the differential equation. Models consisting of combinations of difference or differential equations can be expressed in a similar manner. This shows how modelling constructs often used in DST can be expressed in TTL. Thus, TTL on the one hand subsumes modelling languages based on differential equations, but on the other hand enables the modeller to express more qualitative, logical concepts as well.

6.2 Mathematical Analysis in TTL: Global Dynamic Properties

Within Dynamical Systems Theory and Calculus, also for global properties of a process more specific analysis methods are known. Examples of such analysis methods include mathematical methods to determine equilibrium points, the behaviour around equilibrium points, and the existence of limit cycles [10]. Suppose a set of differential equations is given, for example a predator prey model: dx/dt = f(x, y) ; dy/dt = g(x, y), where f(x, y) and g(x, y) are arithmetical expressions in x and y. Within TTL the following abbreviation is introduced as a definable predicate:

point(γ, t, x, v, y, w) $\Leftrightarrow$ state(γ, t) |= has_value(x, v) $\wedge$ has_value(y, w)

Using this predicate, the following global properties can for example be specified:

Monotonicity
monotic_increase_after(γ, t, x) $\Leftrightarrow$
$\forall t1, t2$ [t ≤ t1 < t2 & point(γ, t1, x, v1, y, w1) & point(γ, t2, x, v2, y, w2) $\Rightarrow$ v1<v2]

Bounded
upward_bounded_after_by(γ, t, M) $\Leftrightarrow \forall t1$ [t ≤ t1 & point(γ, t1, x, v1, y, w1) $\Rightarrow$ v1≤M]

Equilibrium points
These are points in the (x, y) plane for which, when they are reached by a solution, the state stays at this point in the plane for all future time points. This can be expressed as a global dynamic property in TTL as follows:
has_equilibrium(γ, x, v, y, w) $\Leftrightarrow \forall t1$ [point(γ, t1, x, v, y, w) $\Rightarrow \forall t2 \geq t1$ point(γ, t2, x, v, y, w)]
occurring_equilibrium(γ, x, v, y, w) $\Leftrightarrow \exists t$ point(γ, t, x, v, y, w) & has_equilibrium(γ, x, v, y, w)

Behaviour Around an Equilibrium
attracting(γ, x, v, y, w, ε0) $\Leftrightarrow$ has_equilibrium(γ, x, v, y, w) &
ε0>0 $\wedge \forall t$ [point(γ, t, x, v1, y, w1) $\wedge$ dist(v1, w1, v, w) < ε0 $\Rightarrow$
$\forall \varepsilon>0\ \exists t1 \geq t\ \forall t2 \geq t1$ [point(γ, t2, x, v2, y, w2) $\Rightarrow$ dist(v2, w2, v, w) < ε]]

Here, dist(v1, w1, v2, w2) denotes the distance between the points (v1, w1) and (v2, w2) in the (x, y) plane.

Limit cycle
A limit cycle is a set S in the x, y plane such that
$\forall t, v, w$ point(γ, t, x, v, y, w) & (v, w) $\in$ S $\Rightarrow$ $\forall t' \geq t, v', w'$ [point(γ, t', x, v', y, w') $\Rightarrow$ (v', w') $\in$ S]

In specific cases the set can be expressed in an implicit manner by a logical and/or algebraic formula, e.g., an equation, or in an explicit manner by a parameterisation. For these cases it can be logically expressed that a set S is a limit cycle.

(1) When S is defined in an implicit manner by a formula $\varphi(v, w)$ with $S = \{ (v, w) \mid \varphi(v, w) \}$, then it is defined that S is a limit cycle as follows:

$\forall t, v, w$ point(γ, t, x, v, y, w) & $\varphi(v, w) \Rightarrow$ $\forall t' \geq t, v', w'$ [point(γ, t', x, v', y, w') $\Rightarrow \varphi(v', w')$]

E.g., when S is a circle defined by a formula of the form $S = \{ (v, w) \mid v^2 + w^2 = r^2 \}$

(2) When a set S in the plane is parameterised by two functions $c1, c2: [0, 1] \rightarrow \Re$, i.e., $S = \{ (c1(u), c2(u)) \mid u \in [0, 1] \}$, then S is a limit cycle if

$\forall t, u$ point(γ, t, c1(u), c2(u)) $\Rightarrow \forall t' \geq t \; \exists u'$ point(γ, t', c1(u'), c2(u'))

An example of a parameterising for S in the shape of a circle is as follows:

$c1(u) = r \cos 2\pi u$, $c2(u) = r \sin 2\pi u$

In many cases, however, the set S cannot be expressed explicitly in the form of an equation or an explicitly defined parameterisation. What still can be done often is to establish the existence of a limit cycle within a certain area, based on the Poincaré-Bendixson Theorem [16].

6.3 Mathematical Analysis in TTL: Local Dynamic Properties

The global dynamic properties described above can also be addressed from a local perspective. For example, the property of monotonicity (which was expressed above for a whole trace after a certain time point t), can also be expressed for a certain interval (with duration d) around t, as shown below.

Local monotonicity property
monotic_increase_around(γ, t, x, d) $\Leftrightarrow$
$\forall t1, t2$ [t-d $\leq$ t1 < t < t2 $\leq$ t+d & point(γ, t1, x, v1, y, w1) & point(γ, t2, x, v2, y, w2) $\Rightarrow$ v1< v2]

In terms of f and g:
monotic_increase_around(γ, t, x, d) $\Leftrightarrow$ point(γ, t, x, v1, y, w1) $\Rightarrow$ f(v1, w1) > 0

Local bounding property
upward_bounding_around(γ, t, M, δ, d) $\Leftrightarrow$
[point(γ, t, x, v1, y, w1) $\Rightarrow \forall t'$ [t$\leq$t'$\leq$t+d & point(γ, t', x, v2, y, w2) $\Rightarrow$ M-v2 $\geq$ (1-δ)*(M-v1)]

In terms of f and g from the equations dx/dt = f(x, y) and dy/dt = g(x, y):
upward_bounding_around(γ, t, M, δ, d) $\Leftrightarrow$ point(γ, t, x, v1, y, w1) $\Rightarrow$ f(v1, w1) $\leq$ δ/d (M - v1)

Local equilibrium property
From the local perspective of the underlying mechanism, equilibrium points are those points for which dx/dt = dy/dt = 0, i.e., in terms of f and g for this case f(x, y) = g(x, y) = 0.
equilibrium_state(v, w) $\Leftrightarrow$ f(v, w) = 0 & g(v, w) = 0

Local property for behaviour around an equilibrium:
attracting(γ, x, v, y, w, δ, ε0, d) ⇔ has_equilibrium(γ, x, v, y, w) &
ε0>0 ∧ 0< δ <1 ∧ d≥0 ∧ ∀t [point(γ, t, x, v1, y, w1) ∧ dist(v1, w1, v, w) < ε0 ⇒
∀t' [t+d≤t'≤t+2d & point(γ, t', x, v2, y, w2) ⇒ dist(v2, w2, v, w) < δ*dist(v1, w1, v, w)]]

In terms of f and g, this can be expressed by relationships for the eigen values of the matrix of derivatives of f and g.

Local limit cycle property
Let a set S in the plane be parameterised by two explicitly given functions c1, c2: [0, 1] → ℜ, i.e., S = { (c1(u), c2(u)) | u ∈ [0, 1] }, and d1(u) = dc1(u)/du, d2(u) = dc2(u)/du. Then S is a limit cycle if:
∀t, u point(γ, t, c1(u), c2(u)) ⇒ d1(u)*g(c1(u), c2(u)) = f(c1(u), c2(u))*d2(u)

6.4 Logical Relations between Local and Global Properties

The properties of local and global level can be logically related to each other by general interlevel relations, for example, the following ones:

∃d>0 ∀t'≥t monotic_increase_around(γ, t', x, d)	⇒ monotic_increase_after(γ, t, x)
∃d>0, δ>0 ∀t'≥t upward_bounding_around(γ, t, M, δ, d)	⇒ upward_bounded_after_by(γ, t, M)
∀t [state(γ, t) |= equilibrium_state(v, w)	⇒ has_equilibrium(γ, x, v, y, w)
∃d>0, δ>0 attracting(γ, x, v, y, w, δ, ε0, d)	⇒ attracting(γ, x, v, y, w, ε0)

These interlevel relations are general properties of dynamic systems, as explained, e.g., in [10]. Full proofs for these relations fall outside the scope of this paper. However, to make them a bit more plausible, the following sketch is given. The first interlevel relation involving monotonicity can be based on induction on the number of d-intervals of the time axis between two given time points t1 and t2. The second interlevel relation, involving boundedness is based on the fact that local bounding implies that in any d-interval, if the value at the start of the interval is below M, then it will remain below M in that interval. The third interlevel relation, on equilibrium points, is based on the fact that if at no time point the value changes, then at all time points after this value is reached, the value will be the same. For the fourth interlevel relation, notice that local attractiveness implies that for any d-interval the distance of the value to the equilibrium value at the end point is less than δ times the value at the starting point. By induction over the number of d-intervals the limit definition as used for the global property can be obtained.

7 Discussion

The LEADSTO approach discussed in this paper provides means to simulate models of dynamic systems that combine both quantitative and qualitative aspects. A dynamic system, as it is used here, is a system, which is characterised by states and transitions between these states. As such, dynamic systems as considered in [23], which are described by differential equations, constitute a subclass of the dynamic systems considered in this paper. Systems that incorporate both continuous

components and discrete components are sometimes called *hybrid systems*. Hybrid systems are studied in both computer science [9], [19] and control engineering [17]. They incorporate both continuous components, whose dynamics is described by differential equations and discrete components, which are often represented by finite-state automata. Both continuous and discrete dynamics of components influence each other. In particular, the input to the continuous dynamics is the result of some function of the discrete state of a system; whereas the input of the discrete dynamics is determined by the value of the continuous state. In the control engineering area, hybrid systems are often considered as switching systems that represent continuous-time systems with isolated and often simplified discrete switching events. Yet in computer science the main interest in hybrid systems lies in investigating aspects of the discrete behaviour, while the continuous dynamics is often kept simple.

Our LEADSTO approach provides as much place for modelling the continuous constituent of a system, as for modelling the discrete one. In contrast to many studies on hybrid systems in computer science (e.g., [25]), in which a state of a system is described by assignment of values to variables, in the proposed approach a state of a system is defined using a rich ontological basis (i.e., typed constants, variables, functions and predicates). This provides better possibilities for conceptualising and formalising different kinds of systems (including those from natural domains). Furthermore, by applying numerical methods for approximation of the continuous behaviour of a system, all variables in a generated model become discrete and are treated equally as finite-state transition system variables. Therefore, it is not needed to specify so-called *control points* [19], at which values of continuous variables are checked and necessary transitions or changes in a mode of a system's functioning are made. Moreover, using TTL, a super-language of LEADSTO, dynamical systems can be analysed by applying formalised standard techniques from mathematical calculus.

Since LEADSTO has a state-based semantics and allows a high ontological expressivity for defining state properties, many action-based languages (*A*, *B*, *C* [12], *L* [2] and their extensions) can be represented in (or mapped to) the LEADSTO format. In particular, trajectories that define the world evolution in action languages correspond to traces in LEADSTO, fluents evaluated in each state can be represented by state properties, and transitions between states due to actions can be specified by LEADSTO rules that contain the corresponding actions within the antecedents. Furthermore, to represent actions, observations, and goals of agents and facts about the world, the state ontology of LEADSTO includes corresponding sorts, functions and predicates. LEADSTO allows representing both static and dynamic laws as they are defined in [12], and non-deterministic actions with probabilities. To represent and reason about temporal aspects of actions, LEADSTO includes the sort TIME, which is a set of linearly ordered time points.

The expressions of query languages used to reason about actions [2], [12] can be represented in TTL, of which LEADSTO is a sublanguage. TTL formulae can express causality relations of query languages by implications and may include references to multiple states (e.g., histories of temporally ordered sequences of states). Using a dedicated tool [TTL], TTL formulae can be automatically checked on traces (or trajectories) that represent the temporal development of agent systems.

Concerning other related work, in [26], a logic-based approach to simulation-based modelling of ecological systems is introduced. Using this approach, continuous

dynamic processes in ecological systems are conceptualised by system dynamics models (i.e., sets of compartments with flows between them). For formalising these models and performing simulations, the logical programming language Prolog is used. In contrast to this, the LEADSTO approach provides a more abstract (or high-level) logic-based language for knowledge representation.

Also within the area of cognitive modelling, the idea to combine qualitative and quantitative aspects within one modelling approach is not uncommon. A number of architectures have been developed in that area, e.g., ACT-R [1] and SOAR [15]. Such cognitive architectures basically consist of a number of different modules that reflect specific parts of cognition, such as memory, rule-based processes, and communication. They have in common with LEADSTO that they are hybrid approaches, supporting both qualitative (or *symbolic*) and quantitative (or *subsymbolic*) structures. However, in LEADSTO these qualitative and quantitative concepts can be combined within the same expressions, whereas in ACT-R and SOAR separate modules exist to express them. In these cognitive architectures, often the role of the subsymbolic processes is to control the symbolic processes. For example, the subsymbolic part of ACT-R is represented by a large set of parallel processes that can be summarised by a number of mathematical equations, whereas its symbolic part is fulfilled by a production system. Here, the subsymbolic equations control many of the symbolic processes. For instance, if multiple production rules in ACT-R's symbolic part are candidates to be executed, a subsymbolic utility equation may estimate the relative cost and benefit associated with each rule and select the rule with the highest utility for execution.

Accuracy and efficiency of simulation results for hybrid systems provided by the proposed approach to a great extend depend on the choice of a numerical approximation method. Although the proposed approach does not prescribe usage of any specific approximation method (even the most powerful of them can be modelled in LEADSTO), for most of the cases the fourth-order Runge-Kutta method can be recommended, especially when the highest level of precision is not required. For simulating system models, for which high precision is demanded, higher-order numerical methods with an adaptive step size can be applied.

References

1. Anderson, J.R., Lebiere, C. The atomic components of thought. Lawrence Erlbaum Associates, Mahwah, NJ (1998)
2. Baral, C., Gelfond, M., Provetti, A. Representing Actions: Laws, Observation and Hypothesis. Journal of Logic Programming, 31(1-3) (1997) 201-243
3. Barringer, H., Fisher, M., Gabbay, D., Owens, R., Reynolds, M. The Imperative Future: Principles of Executable Temporal Logic, Research Studies Press Ltd. and John Wiley & Sons (1996)
4. Boccara, N. Modeling Complex Systems. Graduate Texts in Contemporary Physics series, Springer-Verlag (2004)
5. Bosse, T., Delfos, M.F., Jonker, C.M., Treur, J. Modelling Adaptive Dynamical Systems to analyse Eating Regulation Disorders. Simulation Journal: Transactions of the Society for Modeling and Simulation International, 82 (2006) 159-171

6. Bosse, T., Jonker, C.M., Los, S.A., Torre, L. van der, Treur, J. Formalisation and Analysis of the Temporal Dynamics of Conditioning. In: Mueller, J.P. and Zambonelli, F. (eds.), Proceedings of the Sixth International Workshop on Agent-Oriented Software Engineering, AOSE'05 (2005) 157-168
7. Bosse, T., Jonker, C.M., Meij, L. van der, Treur, J. LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. In: Eymann, T. et al. (eds.), Proc. MATES'05. LNAI 3550. Springer Verlag (2005) 165-178. Extended version in: International Journal of Artificial Intelligence Tools. To appear, 2007
8. Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., Treur, J. Specification and Verification of Dynamics in Cognitive Agent Models. In: Nishida, T. (ed.), Proceedings of the Sixth International Conference on Intelligent Agent Technology, IAT'06. IEEE Computer Society Press (2006) 247-254
9. Davoren, J.M., Nerode, A. Logics for Hybrid Systems. In Proceedings of the IEEE, 88 (7) (2000) 985-1010
10. Edwards, C.H., Penney, D. L. Calculus with Analytic Geometry. Prentice Hall, London, 5th edition (1998)
11. Gear, C.W. Numerical Initial Value Problems in Ordinary Differential Equations. Englewood Cliffs, NJ: Prentice-Hall (1971)
12. Gelfond, M., Lifschitz, V. Action languages, Electronic Transactions on AI, 3(16) (1998)
13. Hynne F, Dano S, Sorensen PG., Full-scale model of glycolysis in Saccharomyces cerevisiae. Biophys. Chem., 94 (1-2) (2001) 121-63
14. Jonker, C.M., Treur, J. Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. International Journal of Cooperative Information Systems 11 (2002) 51-92.
15. Laird, J.E., Newell, A., and Rosenbloom, P.S. Soar: an architecture for general intelligence. Artificial Intelligence 33 (1) (1987) 1-64.
16. Lefschetz, S. Differential equations: geometric theory. Dover Publications (2005)
17. Liberzon, D., Morse, A. S. Basic problems in stability and design of switched systems, IEEE Control Systems Magazine 19 (5) (1999) 59-70
18. Machado, A. Learning the Temporal Dynamics of Behaviour. Psychological Review, vol. 104 (1997) 241-265
19. Manna, Z., Pnueli, A. Verifying Hybrid Systems. In Hybrid Systems, LNCS 736, Springer-Verlag, (1993) 4-35
20. Meyer, J.J.Ch., Treur, J. (volume eds.). Agent-based Defeasible Control in Dynamic Environments. Series in Defeasible Reasoning and Uncertainty Management Systems (D. Gabbay and Ph. Smets, series eds.) vol. 7, Kluwer Academic Publishers (2002)
21. Morin P.J. Community Ecology. Blackwell Publishing, USA (1999)
22. Pearson, C.E.. Numerical Methods in Engineering and Science. CRC Press (1986)
23. Port, R.F., Gelder, T. van (eds.). Mind as Motion: Explorations in the Dynamics of Cognition. MIT Press, Cambridge, Mass (1995)
24. Press, W. H., Teukolsky, S. A., Vetterling,W. T., Flannery, B. P. Numerical recipes in C: the art of scientific computing. Cambridge university press, second edition (1992)
25. Rajeev, A., Henzinger, T.A., and Wong-Toi, H. Symbolic analysis of hybrid systems. In Proceedings of the 36th Annual Conference on Decision and Control (CDC), IEEE Press (1997) 702-707
26. Robertson, D., Bundy, A., Muetzelfeldt, R., Haggith, M., Ushold, M. Eco-Logic: Logic-Based Approaches to Ecological Modelling. MIT Press, Cambridge, Mass (1991)
27. Sharpanskykh, A., Treur, J. Verifying Interlevel Relations within Multi-Agent Systems. In: Brewka, G., Coradeschi, S., Perini, A., and Traverso, P. (eds.), Proc. of the 17th European Conference on Artificial Intelligence, ECAI'06, IOS Press (2006) 290-294