Modeling of Agent Behavior Using Behavioral Specifications

Alexei Sharpanskykh (sharp@few.vu.nl)
Department of Artificial Intelligence, Vrije Universiteit Amsterdam,
De Boelelaan 1081a, NL-1081 HV Amsterdam, The Netherlands

Jan Treur (treur@few.vu.nl)
Department of Artificial Intelligence, Vrije Universiteit Amsterdam,
De Boelelaan 1081a, NL-1081 HV Amsterdam, The Netherlands

Abstract

The behavioral dynamics of a cognitive agent can be considered both from an external and an internal perspective. From the external perspective, behavior is described by specifying (temporal) correlations between input and output states of the agent. From the internal perspective the agent's dynamics can be characterized by direct (causal) temporal relations between internal, mental states of the agent. The latter type of specifications can be represented in executable format, which allows performing simulations of the agent's behavior under different (environmental) circumstances. For enabling simulations when only given an external behavioral specification, this has to be transformed first into some type of executable format. An automated procedure for such a transformation is proposed in this paper. The application of the transformation procedure is demonstrated by two simulation examples addressing delayed response behavior and adaptive behavior.

Introduction

The behavior of a cognitive agent can be considered both from an external and an internal perspective. From the external perspective, behavior of the agent can be described by correlations of a certain complexity between its input and output states over time, expressed in some (temporal) language, without any reference to internal or mental states of the agent. Such descriptions can be successfully used for modeling relatively simple types of behavior (e.g., stimulus-response behavior (Skinner, 1935)). For less simple types of behavior (e.g., adaptive behavior based on conditioning (Balkenius & Moren, 1999)) an external behavioral specification often consists of more complex temporal relations that can not be directly used for simulations.

From the internal perspective the behavior of the agent can be characterized by a specification of more direct (causal) temporal relations between mental states of the agent, based on which an externally observable behavioral pattern is generated. Executability is an important advantage of such a specification over an external one. By means of executable specifications it is possible to perform automated simulations of different scenarios of the agent's behavior. For enabling automated analysis of an external behavioral specification by simulation, it should be (automatically) translated into executable format. This is the main problem addressed in this paper. As a solution an automated procedure for transformation of the external behavioral specification first into a synthetic executable specification, and subsequently into a general state transition system format is proposed. The executable specification is based on direct executable temporal relations between certain (synthetic, postulated) states. These states correspond to sensory representation memory and preparation states of the agent. The justification of such a transformation is based on the theorem that an external behavioral specification entails a certain dynamic property if and only if the generated executable internal specification entails the same property (Sharpanskykh & Treur, 2005).

Using the generated transition system specification of an agent's behavior, a developed simulation software tool generates a trace, representing changes of internal (mental) states of the agent over time according to a given scenario.

In the next section the concepts for formal modeling of externally observable agent behavior and for specifying a generated executable internal specification are introduced. Next, the transformation procedure from an external into an executable internal specification and subsequently into a general description of a finite state transition system is described in detail. The explanation of the procedure is illustrated by a running example. After that the proposed approach is applied for generating a simulation trace for an example scenario of agent behavior. The paper ends with a discussion.

Formal Modeling of Agent Behavior

From an external perspective an agent can be seen as an autonomous entity that interacts with a dynamic environment via its *input* and *output* (interface) states. At its input the agent receives observations from the environment whereas at its output it generates actions that can change a state of the environment.

States and Traces

A state of the agent as used here can be considered as indication of which properties of the agent (e.g., observations and actions) are true (hold) at a certain point in time. Externally observable state properties of the agent are formalized using the interaction ontology InteractionOnt(A). In general, an ontology is defined as a specification (in order-sorted logic) of a vocabulary that comprises finite sets of

sorts, constants within these sorts, and relations and functions over these sorts. Specifically, InteractionOnt can be seen as a union of input and output state ontologies of the agent (resp., InputOnt and OutputOnt) to define corresponding input and output agent state properties. The state properties are specified using n-ary predicates (with $n \geq 0$). For example, observed(a) means that an agent has an observation of state property a and performing_action(b) represents an action b performed by an agent in an environment. To describe dynamics an explicit reference is made to time: a *trace or trajectory* over an ontology Ont is a time-indexed sequence of states over Ont.

Expressing Dynamic Properties

To express properties of the internal and the external dynamics of the agent, dynamic properties can be formulated that relate properties of states at certain points in time. Consider a simple example of externally observable behavior of an agent:

> "At any point in time if agent A observes food present at position p, then there exists a later point in time, when agent A goes to p."

To express such dynamic properties, and other, more sophisticated ones, the temporal trace language TTL is used; cf. (Jonker & Treur & Wijngaards, 2003; Sharpanskykh & Treur, 2005). TTL allows explicit references to time points and traces in dynamic properties expressions. Thus, the sorted predicate logic temporal trace language TTL is built on atoms referring to traces, time and state properties. For example, "in the output state of agent A in trace γ at time t property p holds" is formalized by state(γ, t, output(A)) |= p. Here |= is a predicate symbol in the language, usually used in infix notation, which is comparable to the Holds-predicate in situation calculus (Reiter, 2001). Sometimes this relation will be used without the third argument: state(γ, t) |= p.

Dynamic properties are expressed by temporal statements built using the usual logical connectives and quantification (for example, over traces, time and state properties). For example the following dynamic property is expressed:

> "in any trace γ, if at any point in time t1 agent A observes that it is dark in the room, then there exists a point in time t2 after t1 such that at t2 in the trace agent A switches on a lamp".

In formalized form:

$$\forall t1 \, [\, state(\gamma, t1, input(A)) \models observed(dark\_in\_room) \Rightarrow \exists t2 \geq t1 \; state(\gamma, t2, output(A)) \models performing\_action(switch\_on\_lamp) \,]$$

Notice that also within states statements about time can be made (e.g., in state properties representing memory). To relate time within a state property to time external to states a state atom present_time(t) is used. This atom is assumed to have correctness and uniqueness properties (Sharpanskykh & Treur, 2005).

Dynamic properties to model a behavioral specification are assumed to be specified as a logical implication from temporal input patterns described by a past statement and an interval statement to a temporal output pattern described by a future statement. For simplicity, the consequent part has a format that prevents non-determinism in behavior.

Past, Interval and Future Statements

a) A *past statement* for a trace γ and a time point t over state ontology Ont is a temporal statement $\varphi_p(\gamma,t)$ in TTL, such that each time variable different from t is restricted to the time interval before t. In other words, for every time quantifier for a time variable s a restriction of the form $s \leq t$, or $s < t$ is required within the statement.

b) An *interval statement* for a trace γ and time points t_1 and t_2 over state ontology Ont is a temporal statement $\varphi(\gamma, t_1, t_2)$ in TTL, such that each time variable different from t_1 and t_2 is restricted to the time interval after t_1 and before t_2. In other words, for every time quantifier for a time variable s a restriction of the form $s \geq t_1$ or $s > t_1$ and $s \leq t_2$, or $s < t_2$ is required within the statement.

c) A *future statement* for a trace γ and a time point t over state ontology Ont is a temporal statement $\varphi_f(\gamma,t)$ in TTL, such that for every time quantifier for a time variable s, different from t a restriction of the form $s \geq t$, or $s > t$ is required within the statement.

Postulated Internal States and their Relations

An executable specification of the dynamics of an agent, generated from an external behavioral specification consists of a set of dynamic properties in an executable temporal language, representing temporal relations between a number of postulated internal (or mental) states.

Internal states of a component A are described using a postulated internal state ontology InternalOnt(A). In cognitive science it is often assumed that an agent maintains a memory in the form of some internal model of the history. As in most literature in the cognitive area we assume that internal states are formed on the basis of (input) observations (sensory representations). For this the predicate memory is used. Therefore, a state ontology InternalOnt should contain sorts and predicates for defining memories about input states. For example, memory(t, observed(a)) expresses that at time point t the agent has memory that it observed a state property a. Before performing an action it is postulated that an agent creates an internal preparation state (e.g., a preparation to perform an action).

Table 1: Executable format of dynamic properties.

If a conjunction of state properties X holds in trace γ at time point t, *(Step Property)* $\forall t \; state(\gamma, t) \models X$	**then** a(nother) conjunction of state properties Y will hold in trace γ at time point t+c, with c>0 an integer constant $\Rightarrow state(\gamma, t+c) \models Y$
If a conjunction of state properties X and a condition property C hold in trace γ at time point t, *(Conditional Persistency Property)* $\forall t \; state(\gamma, t) \models X \,\&\, C$	**then** this conjunction of state properties X will hold in trace γ at the next time point $\Rightarrow state(\gamma, t+1) \models X$
If a conjunction of state properties X holds in trace γ at time point t, *(State Relation Property)* $\forall t \; state(\gamma, t) \models X$	**then** a(nother) conjunction of state properties Y will hold in trace γ at the same time point t $\Rightarrow state(\gamma, t) \models Y$

For example, preparation_for(b) represents a preparation of an agent to perform an action b. Both memory and preparation states belong to a type of states, which are much better understood and physically grounded in neurobiological research than intentional states such as beliefs, desires, and intentions. This motivates the choice that has been made with respect to an internal representation of externally observable behavior of an agent.

Each dynamic property in the internal specification of an agent is specified in one of the *executable* forms, given in Table 1.

Transformation into an Executable Format

The procedure described in this section achieves the transformation of an external behavioral specification for an agent into the executable format and subsequently into the representation of a finite state transition system. Although in this paper this transformation is used for enabling simulations of different scenarios of agent behavior, this procedure can be also applied for verification and validation of agent systems. An external behavioral specification of an agent is defined as follows.

External Behavioral Specification

An *external behavioral specification* for an agent consists of dynamic properties $\varphi(\gamma, t)$ expressed in TTL of the form $[\varphi_p(\gamma, t) \Rightarrow \varphi_f(\gamma, t)]$, where $\varphi_p(\gamma, t)$ is a past statement over the interaction ontology and $\varphi_f(\gamma, t)$ is a future statement. The future statement is represented in the form of a conditional action specification: $\varphi_f(\gamma, t) \Leftrightarrow \forall t_1 > t\ [\varphi_{cond}(\gamma, t, t_1) \Rightarrow \varphi_{act}(\gamma, t_1)]$, where $\varphi_{cond}(\gamma, t, t_1)$ is an interval statement over the interaction ontology, which describes a condition for some specified action(s) and $\varphi_{act}(\gamma, t_1)$ is a (conjunction of) future statement(s) for t_1 over the output ontology of the form state(γ, t_1+c) |= performing_action(a), for some integer constant c and action a.

When the past formula $\varphi_p(\gamma, t)$ is true, within γ at time t a potential to perform one or more action(s) occurs. This potential is actualized at time t_1 when the condition formula $\varphi_{cond}(\gamma, t, t_1)$ is true, which leads to the action(s) being performed in γ at the time point(s) t_1+c indicated in $\varphi_{act}(\gamma, t_1)$.

For any (non-executable) dynamic property $\varphi(\gamma, t)$ from an external behavioral specification for an agent A, the following transformation is performed to obtain an executable representation.

The Transformation Procedure

(1) Identify executable temporal properties, which describe transitions from interaction states to memory states.
(2) Identify executable temporal properties, which describe transitions from memory states to preparation states for performing an action.
(3) Specify executable properties, which describe the transition from preparation states to the corresponding action performance states.
(4) From the executable properties, identified during the steps 1-3, construct a part of the specification $\pi(\gamma, t)$, which describes the internal dynamics of agent A, corresponding to the property $\varphi(\gamma, t)$.
(5) Apply the steps 1-4 to all properties in the external behavioral specification of the agent A. In the end add to the executable specification the dynamic properties, which were initially specified in executable form using an ontology, different than InteractOnt(A).
(6) Translate the identified during the steps 1-5 executable rules into the transition system representation.

The details of the described procedure are explained by means of an example, in which delayed-response behavior of a laboratory mouse is analyzed; e.g., Hunter (1912).

The initial situation for the conducted experiment is as follows: the mouse is placed in front of a transparent screen that separates it from a piece of food that is put behind the screen. The mouse is able to observe the position of food and of the screen. At some moment after food has been put, a cup is placed covering the food, which makes food invisible for the mouse. After some time the screen is raised and the animal is free to go to any position. If the mouse comes to the position, where the food is hidden, then it will be capable to lift up the cup and get the food.

The behavioral specification for the conducted experiment consists of environmental properties and externally observable behavioral properties of the mouse. For the purposes of illustration of the proposed transformation procedure the dynamic property that describes the delayed-response behavior of the mouse has been chosen. Informally this property expresses that the mouse goes to the position with food if it observes that there is no screen and at some point in the past the mouse observed food and since then did not observe the absence of food. According to the definition of an external behavioral specification the considered property can be represented in the form $[\varphi_p(\gamma, t) \Rightarrow \varphi_f(\gamma, t)]$, where $\varphi_p(\gamma, t)$ is a formula

$\exists t2\ [\text{state}(\gamma, t2, \text{input(mouse)}) \models \text{observed(food)} \wedge$
$\forall t3, t \geq t3 > t2\ \text{state}(\gamma, t3, \text{input(mouse)}) \models \text{not(observed(not(food)))}]$

and $\varphi_f(\gamma, t)$ is a formula

$\forall t4 > t\ [\text{state}(\gamma, t4, \text{input(mouse)}) \models \text{observed(not(screen))} \Rightarrow$
$\text{state}(\gamma, t4+c, \text{output(mouse)}) \models \text{performing\_action(goto\_food)}\ ]$

with $\varphi_{cond}(\gamma, t, t4)$ is

$\text{state}(\gamma, t4, \text{input(mouse)}) \models \text{observed(not(screen))}$

and $\varphi_{act}(\gamma, t4)$ is

$\text{state}(\gamma, t4+c, \text{output(mouse)}) \models \text{performing\_action(goto\_food)},$

where t is the present time point with respect to which the formulae are evaluated.

Step 1. ***From interaction states to memory states***

The formula $\varphi_{mem}(\gamma, t)$ obtained by replacing all occurrences in $\varphi_p(\gamma, t)$ of subformulae of the form state(γ, t') |= p by state(γ, t) |= memory(t', p) is called the *memory formula for* $\varphi_p(\gamma, t)$.

Thus, a memory formula defines a sequence of past events (i.e., a history) (e.g., observations of an external world, actions) for the present time point t. The time interval for generation of an internal memory state of an agent from its observation is assumed to be incommensurably smaller than time intervals between external events (i.e., stimuli). Therefore, in the proposed model both an observation state and a corresponding memory state are created at the same time point.

According to Lemma 1 (Sharpanskykh & Treur, 2005) $\varphi_{mem}(\gamma, t)$ is equivalent to some formula $\delta^*(\gamma, t)$ of the form state(γ, t) |= $q_{mem}(t)$, where $q_{mem}(t)$ is called *the normalized memory state formula for* $\varphi_{mem}(\gamma, t)$, which uniquely describes the present state at the time point t by a certain history of

events. Moreover, q_{mem} is the state formula $\forall t'$ [present_time(t') $\Rightarrow$ $q_{mem}(t')$]. For the considered example $q_{mem}(t)$ for $\varphi_{mem}(\gamma, t)$ is specified as:

$\exists t2$ [memory(t2, observed(food)) $\wedge$
$\forall t3, t \geq t3 > t2$ memory(t3, not(observed(not(food))))]

Additionally, memory state persistency properties are composed for all memory atoms. For example, for the atom memory(t2, observed(food)) the corresponding persistency property is defined as:

$\forall t''$ state(γ, t'', internal(mouse)) |= memory(t', observed(food)) $\Rightarrow$
state(γ, t''+1, internal(mouse)) |= memory(t', observed(food))

Rules that describe creation and persistence of memory atoms are given in *the executable theory from observation states to memory states* $Th_{o\rightarrow m}$. For the considered example:

$\forall t'$ state(γ, t', input(mouse)) |= observed(food) $\Rightarrow$
state(γ, t', internal(mouse)) |= memory(t', observed(food))
$\forall t'$ state(γ, t', input(mouse)) |= not(observed(not(food))) $\Rightarrow$
state(γ, t', internal(mouse)) |= memory(t', not(observed(not(food))))
$\forall t'$ state(γ, t', input(mouse)) |= observed(not(food)) $\Rightarrow$
state(γ, t', internal(mouse)) |= memory(t', observed(not(food)))
$\forall t''$ state(γ, t'', internal(mouse)) |= memory(t', observed(food)) $\Rightarrow$
state(γ, t''+1, internal(mouse)) |= memory(t', observed(food))
$\forall t''$ state(γ, t'', internal(mouse)) |= memory(t', not(observed(not(food)))) $\Rightarrow$
state(γ, t''+1, internal(mouse)) |= memory(t', not(observed(not(food))))
$\forall t''$ state(γ, t'', internal(mouse)) |= memory(t', observed(not(food))) $\Rightarrow$
state(γ, t''+1, internal(mouse)) |= memory(t', observed(not(food)))

Step 2. ***From memory states to preparation states***

Obtain $\varphi_{cmem}(\gamma, t, t_1)$ by replacing all occurrences in $\varphi_{cond}(\gamma, t, t_1)$ of state(γ, t') |= p by state(γ, t_1) |= memory(t', p). The condition memory formula $\varphi_{cmem}(\gamma, t, t_1)$ contains a history of events, between the time point t, when $\varphi_p(\gamma, t)$ is true and the time point t_1, when the formula $\varphi_{cond}(\gamma, t, t_1)$ becomes true. Again by Lemma 1 $\varphi_{cmem}(\gamma, t, t_1)$ is equivalent to the state formula state(γ, t_1) |= $q_{cond}(t, t_1)$, where $q_{cond}(t, t_1)$ is called *the normalized condition state formula for* $\varphi_{cmem}(\gamma, t, t_1)$. Moreover, $q_{cond}(t)$ is the state formula $\forall t'$ [present_time(t') $\Rightarrow$ $q_{cond}(t, t')$].

For the considered example $q_{cond}(t, t4)$ for $\varphi_{cmem}(\gamma, t)$ is obtained as: memory(t4, observed(not(screen))) and $q_{cond}(t)$: $\forall t'$ [present_time(t') $\Rightarrow$ memory(t', observed(not(screen)))].

Obtain $\varphi_{prep}(\gamma, t_1)$ by replacing in $\varphi_{act}(\gamma, t_1)$ any occurrence of state(γ, t_1+c) |= performing_action(a) by state(γ, t_1) |= preparation_for(action(t_1+c, a)), for some number c and action a. The preparation state is created at the same time point t_1, when the condition for an action $\varphi_{cond}(\gamma, t, t_1)$ is true. By Lemma 1 $\varphi_{prep}(\gamma, t_1)$ is equivalent to the state formula state(γ, t_1) |= $q_{prep}(t_1)$, where $q_{prep}(t_1)$ is called *the normalized preparation state formula for* $\varphi_{cond}(\gamma, t_1)$. Moreover, q_{prep} is the state formula $\forall t'$ [present_time(t')] $\Rightarrow$ $q_{prep}(t')$]. For the considered example $q_{prep}(t4)$ is composed as preparation_for(action(t4+c, goto_food)).

Rules, which describe generation and persistence of condition memory states, a transition from the condition to the preparation state, and the preparation state generation and persistence, are given in *the executable theory from memory states to preparation states* $Th_{m\rightarrow p}$. For the considered example:

$\forall t'$ state(γ, t', input(mouse)) |= observed(not(screen)) $\Rightarrow$
state(γ, t', internal(mouse)) |= [memory(t', observed(not(screen))) $\wedge$
stimulus_reaction(observed(not(screen)))]
$\forall t''$ state(γ, t'', internal(mouse)) |= memory(t', observed(not(screen))) $\Rightarrow$
state(γ, t''+1, internal(mouse)) |= memory(t', observed(not(screen)))
$\forall t'$ state(γ, t') |= $\forall t''$ [present_time(t'') $\rightarrow$
$\exists t2$ [memory(t2, observed(food)) $\wedge$
$\forall t3, t'' \geq t3 > t2$ memory(t3, not(observed(not(food))))]] $\Rightarrow$
state(γ, t') |= $\forall t'''$ [present_time(t''') $\rightarrow$ [$\forall t4 > t'''$ [memory(t4,
observed(not(screen))) $\rightarrow$ preparation_for(action(t4+c, goto_food))]]]
$\forall t'$, t state(γ, t') |= [$\forall t'''$ [present_time(t''') $\rightarrow$ [$\forall t4 > t'''$
[memory(t4, observed(not(screen))) $\rightarrow$
preparation_for(action(t4+c, goto_food))]]] $\wedge$
$\forall t''$ [present_time(t'') $\rightarrow$ memory(t'', observed(not(screen)))] $\wedge$
stimulus_reaction(observed(not(screen)))] $\Rightarrow$
state(γ, t', internal(mouse)) |= $\forall t4$ [present_time(t4) $\rightarrow$
preparation_for(action(t4+c, goto_food))]
$\forall t'$ state(γ, t') |= [stimulus_reaction(observed(not(screen))) $\wedge$
not(preparation_for(action(t'+c, goto_food)))] $\Rightarrow$
state(γ, t'+1) |= stimulus_reaction(observed(not(screen)))
$\forall t'$ state(γ, t', internal(mouse)) |= [preparation_for(action(t'+c, goto_food)) $\wedge$
not(performing_action(goto_food))] $\Rightarrow$
state(γ, t'+1, internal(mouse)) |= preparation_for(action(t'+c, goto_food)).

The auxiliary atoms stimulus_reaction(a) are used to reactivate agent preparation states for generating recurring actions.

Step 3. ***From preparation states to action states***

The preparation state preparation_for(action(t_1+c, a)) is followed by the action state, created at the time point t_1+c. Rules that describe a transition from preparation to action states are given in *the executable theory from the preparation to the action state(s)* $Th_{p\rightarrow a}$. For the considered example the following rule holds:

$\forall t'$ state(γ, t', internal(mouse)) |= preparation_for(action(t'+c, goto_food)) $\Rightarrow$
state(γ, t'+c, output(mouse)) |= performing_action(goto_food).

Step 4. ***Constructing an executable specification***

An executable specification $\pi(\gamma, t)$ for agent A is defined by a union of the dynamic properties from the executable theories $Th_{o\rightarrow m}$, $Th_{m\rightarrow p}$ and $Th_{p\rightarrow a}$, identified during the steps 1-3. For the purposes of simulations of agent behavior the non-executable external behavioral specification is replaced by the executable behavioral specification.

Step 5. ***Constructing an executable specification for the whole external behavioral specification of an agent***

Other non-executable dynamic properties from the agent behavioral specification are substituted by executable ones by applying the same sequence of steps 1-4. In the end the executable properties for generating observation states from the states of the external world are added:

$\forall t'$ state(γ, t', world) |= [food $\wedge$ not(cup)] $\Rightarrow$
state(γ, t', input(mouse)) |= observed(food)
$\forall t'$ state(γ, t', world) |= [not(food) $\wedge$ not(cup)] $\Rightarrow$
state(γ, t', input(mouse)) |= observed(not(food))
$\forall t'$ state(γ, t', world) |= not(screen) $\Rightarrow$
state(γ, t', input(mouse)) |= observed(not(screen))
$\forall t'$ state(γ, t', world) |= screen $\Rightarrow$
state(γ, t', input(mouse)) |= observed(screen)

It is assumed that an observation state is generated at the same time point, when a corresponding state of the external world is active.

Step 6. ***Translation of an executable specification into a description of a transition system***

For the purpose of practical simulation a behavioral specification based on executable temporal logical properties generated at Step 5 is translated into a finite state transition system description. A finite state transition system is described by a tuple $\langle Q, Q_0, \Sigma, \twoheadrightarrow \rangle$, where Q is a finite set of states of an agent, $Q_0 \subseteq Q$ is a set of initial states, Σ is a set

of labels or events, which trigger the transition and $\rightarrow \subseteq Q \times \Sigma \times Q$ is a set of transitions. From the description of agent behavior in the form of a transition system the same traces are generated as by executing the dynamic properties.

For translating the executable specification of agent behavior into the transition system representation the introduced earlier predicate present_time(t) is used. This predicate is only true in a state for the current time point t. The executable properties from the executable specification, translated into the transition rules for the considered example are given below:

food ∧ not(cup) ⟶ observed(food)
not(food) ∧ not(cup) → observed(not(food))
screen ⟶ observed(screen)
not(screen) ⟶ observed(not(screen))
present_time(t) ∧ observed(food) ⟶ memory(t, observed(food))
present_time(t) ∧ not(observed(not(food))) ⟶
memory(t, not(observed(not(food))))
present_time(t) ∧ observed(not(food)) ⟶
memory(t, observed(not(food)))
present_time(t) ∧ observed(not(screen)) ⟶
memory(t, observed(not(screen))) ∧
stimulus_reaction(observed(not(screen)))
memory(t, observed(food)) ⟶ memory(t, observed(food))
memory(t, not(observed(not(food)))) ⟶
memory(t, not(observed(not(food))))
memory(t, observed(not(food))) ⟶
memory(t, observed(not(food)))
memory(t, observed(not(screen))) ⟶
memory(t, observed(not(screen)))
present_time(t) ∧ ∃t2 [memory(t2, observed(food)) ∧
∀t3, t ≥ t3 > t2 memory(t3, not(observed(not(food))))] ⟶
conditional_preparation_for(action(goto_food))
present_time(t) ∧ conditional_preparation_for(action(goto_food)) ∧
memory(t, observed(not(screen))) ∧
stimulus_reaction(observed(not(screen))) ⟶
preparation_for(action(t+c, goto_food))
present_time(t) ∧ stimulus_reaction(observed(not(screen))) ∧
not(preparation_for(action(t+c, goto_food))) ⟶
stimulus_reaction(observed(not(screen)))
preparation_for(action(t+c, goto_food)) ∧
not(performing_action(goto_food)) ⟶
preparation_for(action(t+c, goto_food))
preparation_for(action(t+c, goto_food)) ∧ present_time(t+c-1) ⟶
performing_action(goto_food).

For performing simulations a special software tool has been developed. Based on a specification of agent behavior in form of a transition system and using a sequence of external events (i.e., stimuli) as input, the program generates a trace (i.e., a sequence of agent states over time). The generated in such way traces can be used for analysis of external and internal dynamics of the agent in different experimental settings. Furthermore, a transition system representation can be used for construction of graphical models of agent dynamics. A graphical model for the considered example is shown in Figure 1. The state description literals with names started with a capital letter denote variables, which allow a concise representation of sets of states

The model in Figure 1 has been built manually. However, there exist tools, as one described in van Ham, van de Wetering, and van Wijk (2002), which allow for automatic visualization of finite state transition systems and can be used for graphical analysis of executable models.

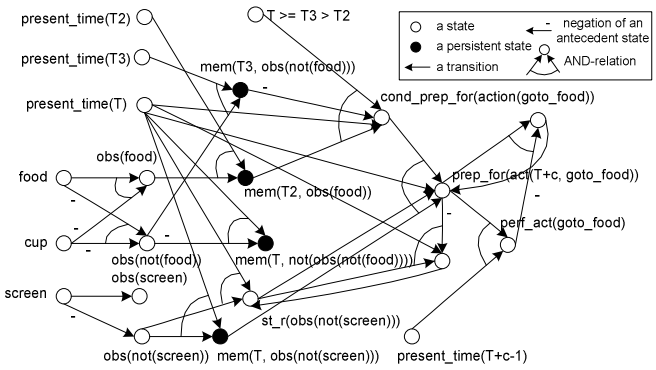

Figure 1: Graphical model, which describes delayed-response behavior of a mouse in executable form.

Simulation Example

In this Section the proposed transformation procedure is applied for simulating adaptive behavior of *Aplysia Californica* (a sea hare). In neurobiology *Aplysia* has been often used for investigating classical and operant conditioning (Carew & Walters & Kandel, 1981). Consider a slightly simplified classical conditioning experiment of the *Aplysia*'s defensive withdrawal reflex. Before a learning phase a strong noxious stimulus (an electric shock) on the *Aplysia*'s tail produces a defensive reflex (a contraction), while a light tactile stimulus on *Aplysia*'s siphon does not lead to contraction.

During the learning phase a light tactile stimulus on the Aplysia's siphon is repeatedly paired with an electric shock on its tail. After a few trials (for this example three temporal pairings are assumed) the animal reacts by contraction to the light tactile stimulus. The property that describes the learning process of the animal from the external perspective can be represented in the form $[\varphi_p(\gamma, t) \Rightarrow \varphi_f(\gamma, t)]$, where $\varphi_p(\gamma, t)$ is the formula:

∃t2, t3, t4, t5, t6, t7 [t2 < t3 ∧ t3 < t4 ∧ t4 < t5 ∧ t5 < t6 ∧ t6 < t7 ∧ t7 < t ∧
state(γ, t2, input(aplysia)) |= observed(touch_siphon) ∧
state(γ, t3, input(aplysia)) |= observed(tail_shock) ∧
state(γ, t4, input(aplysia)) |= observed(touch_siphon) ∧
state(γ, t5, input(aplysia)) |= observed(tail_shock) ∧
state(γ, t6, input(aplysia)) |= observed(touch_siphon) ∧
state(γ, t7, input(aplysia)) |= observed(tail_shock)]

and $\varphi_f(\gamma, t)$ is the formula

∀t8 ≥ t [state(γ, t8, input(aplysia)) |= observed(touch_siphon) ⇒
state(γ, t8+c, output(aplysia)) |= performing_action(contraction)]

with $\varphi_{cond}(\gamma, t, t8)$ is

∀t8 ≥ t state(γ, t8, input(aplysia)) |= observed(touch_siphon)

and $\varphi_{act}(\gamma, t8)$ is

state(γ, t8+c, output(aplysia)) |= performing_action(contracts),

For this experiment c is assumed to be equal to two time units in a relative time scale.

Using the automated procedure, from the external behavioral specification of *Aplysia* a transition system was generated. This transition system was used to simulate a scenario of the animal's behavior with the following stimuli: touch the siphon (time points 0, 5, 9 and 15) and shock on

the tail (time points 1, 6 and 10). The results of the simulation in form of a partial trace are given in Table 2.

Table 2: Partial simulation trace illustrating adaptive behavior of *Aplysia Californica*.

0: present_time(0) world(0,touch_siphon)	11: not(preparation_for(action(3, contracts)))
1: not(world(0,touch_siphon)) observed(0,touch_siphon)	……
2: memory(observed(0,touch_siphon)) not(observed(0,touch_siphon)) stimulus_reaction(observed(touch_siphon))	39: present_time(15) world(15,touch_siphon)
3: present_time(1) world(1,tail_shock)	40: not(world(15,touch_siphon)) observed(15,touch_siphon)
4: not(world(1,tail_shock)) observed(1,tail_shock)	41: memory(observed(15,touch_siphon)) not(observed(15,touch_siphon)) stimulus_reaction(observed(touch_siphon))
5: memory(observed(1,tail_shock)) not(observed(1,tail_shock)) stimulus_reaction(observed(tail_shock))	42: preparation_for(action(17,contracts))
6: conditional_preparation_for(action(contracts))	43: not(stimulus_reaction(observed(touch_siphon))) not(stimulus_reaction(observed(tail_shock)))
7: preparation_for(action(3,contracts))	44: present_time(16)
8: not(stimulus_reaction(observed(touch_siphon))) not(stimulus_reaction(observed(tail_shock)))	45: present_time(17) performing_action(contracts)
9: present_time(2)	46: not(preparation_for(action(17,contracts)))
10: present_time(3) performing_action(contracts)	47: present_time(18)

In the given trace the process of conditioning starts at the state 0 (time point 0) and finishes at the state 36 (time point 12). After that the animal reacts to a light tactile stimulus (state 39) by producing a defensive reflex (states 42-45).

Discussion

Behavior of organisms comes in a variety of forms and complexities. Simple forms of behavior such as stimulus-response patterns can be formalized in relatively simple terms, based on direct stimulus-action associations that can be considered as associations between an input state and a subsequent output state of the organism. A description of an organism's behavior in terms of such stimulus-action associations can directly be used as a basis to model and simulate this behavior. For more complex behavior, however, the picture is not so simple. To describe behavior from the external perspective, in general, an input-output correlation (cf. Kim, 1996) has to be specified which indicates how a pattern of input states over time relates to a pattern of output states over time. With increasing complexity of the behavior considered, specification of such an input-output correlation will become more complex, and not take the form of direct stimulus-action associations anymore. The question arises how such more complex descriptions of behavior can be expressed and handled, and, in particular, how such behavior can be simulated and analyzed. The answer on this question developed in this paper is twofold. First, a formal language is put forward that allows specifying behavior from an external perspective in terms of dynamic properties involving input states and output states over time. Secondly, it is shown how external behavior specifications expressed in such a language can be automatically transformed into executable specifications that easily can be used to perform simulation and analysis. This transformation creates a specification based on postulated internal states (in particular memory states and preparation states), and their direct temporal relationships.

Sometimes, when a structure of a neurological circuit of an organism is known, it is possible to relate postulated internal states to certain real neurological states of an organism. The neurological model of *Aplysia Californica*, suggested by Roberts and Glanzman (2003) allows finding some correspondences between the postulated internal states described in the example of this paper and the real physical states of the organism. The observation states from our model can be related to activation states of sensory neurons, whereas the memory states (to some extent) can be put into correspondence with an enhancement of the strength of the synaptic connection between the sensory and motor neurons and with an associative increase in the excitability of the siphon sensory neurons of *Aplysia* (as followed from a correspondence with professor Glanzman)

References

Balkenius, C., & Moren, J. (1999). Dynamics of a classical conditioning model. *Autonomous Robots, 7*, 41-56.

Carew, T.J. & Walters, E.T., & Kandel, E.R. (1981). Classical conditioning in a simple withdrawal reflex in *Aplysia Californica*. *The Journal of Neuroscience, 1(12)*, 1426-1437.

Jonker, C.M., & Treur J., & Wijngaards W.C.A. (2003). A temporal-modelling environment for internally grounded beliefs, desires, and intentions. *Cognitive Systems Research Journal, 4(3)*, 191-210.

Hunter, W.S. (1912). The delayed reaction in animals. *Behavioral Monographs, 2*, 1-85.

Kim, J. (1996). *Philosophy of Mind*. Westview Press

Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. Cambridge MA: MIT Press.

Roberts, A.C., & Glanzman, D.L. (2003). Learning in Aplysia: looking at synaptic plasticity from both sides. *Trends in Neurosciences, 26*, 662-670.

Skinner, B.F. (1935). The generic nature of the concepts of stimulus and response. *Journal of General Psychology, 12*, 40-65.

Sharpanskykh, A. & Treur, J. (2005). *Modeling of Agent Behavior Using Behavioral Specifications* (Tech. Rep. 06-02ASRAI; http://hdl.handle.net/1871/9123). Vrije Universiteit, Amsterdam.

van Ham, F., & van de Wetering, H., & van Wijk, J.J. (2002). Interactive Visualization of State Transition Systems. *IEEE Transactions on Visualization and Computer Graphics, 8(4)*, IEEE CS Press, 319-329.