Modeling of Agent Behavior Using Behavioral Specifications

Technical Report 06-02ASRAI

Alexei Sharpanskykh and Jan Treur
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 executable format, and subsequently into a transition system description. 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.

1. 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 (Hunter, 1912; 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[1] that an external behavioral specification entails a certain dynamic property if and only if the generated executable internal specification entails the same property.

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.

[1] For the philosophical embedding of the introduced procedure and the theoretical results (i.e., lemmas, propositions and the theorem) for this paper we refer to Appendixes A and B

2. 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.

2.1 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.

2.2 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; Appendix B). 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:

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

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 predicate is assumed to have the properties of correctness and uniqueness (see Appendix B).

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.

2.3 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.

2.4 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). 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.

Table 1: Executable format of dynamic properties.

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

3. 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.

Definition (External behavioral specification)

An *external behavioral specification* for an agent system 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: $\varphi_f(\gamma, t) \Leftrightarrow \forall t_1 > t\ [\varphi_{cond}(\gamma, t, t_1) \Rightarrow \varphi_{act}(\gamma, t_1)]$, 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 a past formula $\varphi_p(\gamma, t)$ is true for γ at time t, a potential to perform one or more action(s) exists. This potential is realized at time t_1 when the condition formula $\varphi_{cond}(\gamma, t, t_1)$ becomes true, which leads to the action(s) being performed at the time point(s) t_1+c indicated in $\varphi_{act}(\gamma, t_1)$.

Let $\varphi(\gamma, t)$ be a non-executable dynamic property from an external behavioral specification for agent A, for which an executable representation should be found.

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.

Let us explain the details of the described procedure by means of an example, in which a delayed-response behavior of a laboratory mouse is investigated.

The initial situation for the conducted experiment is defined 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 (see Appendix B) $\varphi_{mem}(\gamma, t)$ 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:

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

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:

∀t'' state(γ, t'', internal(mouse)) |= memory(t', observed(food)) ⇒
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* $\mathrm{Th}_{o\to m}$. For the considered example:

∀t' state(γ, t', input(mouse)) |= observed(food) ⇒
state(γ, t', internal(mouse)) |= memory(t', observed(food))

∀t' state(γ, t', input(mouse)) |= not(observed(not(food))) ⇒
state(γ, t', internal(mouse)) |= memory(t', not(observed(not(food))))

∀t' state(γ, t', input(mouse)) |= observed(not(food)) ⇒
state(γ, t', internal(mouse)) |= memory(t', observed(not(food)))

∀t'' state(γ, t'', internal(mouse)) |= memory(t', observed(food)) ⇒
state(γ, t''+1, internal(mouse)) |= memory(t', observed(food))

∀t'' state(γ, t'', internal(mouse)) |= memory(t', not(observed(not(food)))) ⇒
state(γ, t''+1, internal(mouse)) |= memory(t', not(observed(not(food))))

∀t'' state(γ, t'', internal(mouse)) |= memory(t', observed(not(food))) ⇒
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 ∀t' [present_time(t') ⇒ $q_{cond}(t, t')$].

For the considered example $q_{cond}(t, t4)$ for $\varphi_{cmem}(\gamma, t)$ is obtained as: memory(t4, observed(not(screen))) and $q_{cond}(t)$: ∀t' [present_time(t') ⇒ 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 ∀t' [present_time(t')] ⇒ $q_{prep}(t')$]. 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* $\mathrm{Th}_{m\to p}$. For the considered example:

∀t' state(γ, t', input(mouse)) |= observed(not(screen)) ⇒
state(γ, t', internal(mouse)) |= [memory(t', observed(not(screen))) ∧
stimulus_reaction(observed(not(screen)))]

∀t'' state(γ, t'', internal(mouse)) |= memory(t', observed(not(screen))) ⇒
state(γ, t''+1, internal(mouse)) |= memory(t', observed(not(screen)))

∀t' state(γ, t') |= ∀t'' [present_time(t'') →
∃t2 [memory(t2, observed(food)) ∧
∀t3, t'' ≥ t3 > t2 memory(t3, not(observed(not(food))))]] ⇒
state(γ, t') |= ∀t''' [present_time(t''') → [∀t4 > t''' [memory(t4,
observed(not(screen))) → preparation_for(action(t4+c, goto_food))]]]

∀t', t state(γ, t') |= [∀t''' [present_time(t''') → [∀t4 > t'''
[memory(t4, observed(not(screen))) →
preparation_for(action(t4+c, goto_food))]]] ∧
∀t'' [present_time(t'') → memory(t'', observed(not(screen)))] ∧
stimulus_reaction(observed(not(screen)))] ⇒
state(γ, t', internal(mouse)) |= ∀t4 [present_time(t4) →
preparation_for(action(t4+c, goto_food))]

∀t' state(γ, t') |= [stimulus_reaction(observed(not(screen))) ∧

not(preparation_for(action(t'+c, goto_food)))] ⇒

state(γ, t'+1) |= stimulus_reaction(observed(not(screen)))

∀t' state(γ, t', internal(mouse)) |= [preparation_for(action(t'+c, goto_food)) ∧

not(performing_action(goto_food))] ⇒

state(γ, t'+1, internal(mouse)) |= preparation_for(action(t'+c, goto_food)).

The auxiliary atoms stimulus_reaction(a) are used for reactivation of 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)* $Th_{p\to a}$. For the considered example the following rule holds:

∀t' state(γ, t', internal(mouse)) |= preparation_for(action(t'+c, goto_food)) ⇒
state(γ, t'+c, output(mouse)) |= performing_action(goto_food).

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

An executable specification π(γ, t) for agent A is defined by a union of the dynamic properties from the executable theories $Th_{o\to m}$, $Th_{m\to p}$ and $Th_{p\to a}$, 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. The justification of such substitution is based on the theorem in Appendix B.

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:

∀t' state(γ, t', world) |= [food ∧ not(cup)] ⇒

state(γ, t', input(mouse)) |= observed(food)

∀t' state(γ, t', world) |= [not(food) ∧ not(cup)] ⇒

state(γ, t', input(mouse)) |= observed(not(food))

∀t' state(γ, t', world) |= not(screen) ⇒

state(γ, t', input(mouse)) |= observed(not(screen))

∀t' state(γ, t', world) |= screen ⇒

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 ⟨Q, Q_0, Σ, ↠⟩, 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 ↠ ⊆ Q x Σ x 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.

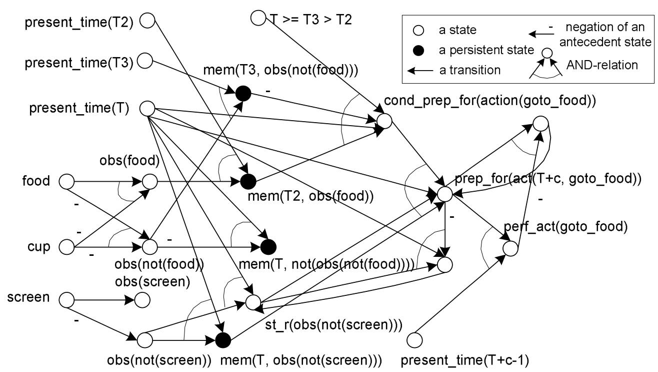

Figure 1: Graphical model, which describes delayed-response behavior of a mouse in executable form.

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.

4. 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 a 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 a 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 $\forall t8 \geq t$ 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 is generated[2]. This transition system is used for simulating a scenario of 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).

5. 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

[2] For the external, executable and transition system specifications, the complete trace and a graphical model for this example see Appendixes C, D

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)

6. 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.

Heil, J. (2000). *Philosophy of Mind*. Routledge.

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

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

Priest, S. (1991). *Theories of the Mind*. Penguin.

Putman, H. (1975). *Mind, Language, and Reality: Philosophical papers*, vol.2. Cambridge: Cambridge University 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.

Skinner, B.F. (1953). *Science and human behavior*. New York: Macmillan.

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.

Watson, J. B. (1913). Psychology as the Behaviorist Views It. *Psychological review, 20*, 158-177.

Appendix A: Philosophical embedding

To consider the behavior of an actor (an agent) from an external or an internal perspective is also a theme within Philosophy of Mind: the distinction between behaviorist and functionalist perspectives; e.g., Kim (1996). 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 properties of the agent; these are called input-output correlations by Kim (1996). Such view is considered within the philosophical perspective of behaviorism (Kim, 1996). The states of the agent are required to be publicly observable and the statements that describe these states should be intersubjectively verifiable (Heil, 2000). According to the apologists of behaviorism J. Watson (1913) and B. Skinner (1953), internal states of the agent (mental or inner states) are considered to be methodologically intractable and unnecessary, since they are based on a personal subjective experience and evaluations and can not be used for analysis and predictions of the agent behavior. However, for less simple types of behavior, a specification of externally observable behavior of the agent often consists of complex temporal relations that can not be directly automated and 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. Such perspective is taken within the philosophical perspective of functionalism (Kim, 1996). From this perspective mental states are described by their functional or causal roles. In contrast to behaviorism, a mental state is characterized by its direct temporal or causal relations with input, output and other mental states. Functionalism was originally formulated by Putnam in terms of a “Turing machine” (Putman, 1975), an abstract machine to give a mathematically precise definition of an algorithm or an automatic procedure. However, in general other executable (temporal) languages can be applied to specify functional roles.

Executability is an important advantage of functionalist specifications over behaviorist ones. By means of executable specifications it is possible to perform automated simulations of different scenarios and analysis of agent behavior. For enabling automated analysis of a behaviorist model, it should be (automatically) translated into executable format, i.e., into functionalist representation using certain (synthetic, hypothetical) internal mental states. So, the main problem addressed in this paper can be expressed as to transform behaviorist specifications into functionalist ones. Indeed a description of internal dynamics is provided that conforms to the postulates of functionalism and can be considered from the positions of this perspective. Thus, by means of the formalized and automated procedure proposed a relationship between the perspectives of behaviorism and functionalism is established.

As in the literature on logical behaviorism (Priest, 1991) it is claimed that any meaningful psychological statement can be transformed into a behaviorist style of statement (e.g., Kim, 1996), a converse translation can also be considered. This will be done in future work, as an extension of the work presented in this paper. The combination of both translations can be considered as well, thus obtaining a form of normalization of a functionalist-style specification, or the same for a behaviorist-style specification.

Appendix B: Theoretical results

TTL Syntax and Semantics

The language TTL, short for Temporal Trace Language, is a variant of order-sorted predicate logic. Whereas standard multi-sorted predicate logic is a language to reason about static properties only, TTL is an extension of such languages with facilities for reasoning about the dynamic properties of arbitrary systems expressed by static languages.

Language of TTL

The **language of TTL** is defined as follows.

An **alphabet** for TTL consists of a signature and standard logical connectives.

A **signature** consists of the symbols of the following classes:

(1) several standard sorts, among which TIME (a set of all time points), STATE (a set of all state names), TRACE (a set of all trace names; a trace can be considered as a timeline), STATPROP (a set of all state property expressions), TRUTH_VALUE (a set with two elements: false and true) and STATOM (STATOM is a subsort of STATPROP). All other sorts, which depend on an application domain, represent subsorts of the general sort OBJECT. Sorts STATE_INSTANCE and STATE_VARIABLE are subsorts of the sort OBJECT.

(2) countably infinite number of individual variables of each sort. We shall use t with subscripts and superscripts for variables of the sort TIME; γ with subscripts and superscripts for variables of the sort TRACE; s with subscripts and superscripts for variables of the sort STATE.

(3) a number of constants for sorts TIME, STATE, TRACE, STATPROP, TRUTH_VALUE, OBJECT and STATOM, among which $\otimes$ of the sort STATE, and true and false of the sort TRUTH_VALUE. The constant $\otimes$ designates an undefined state in the trace for a certain time value.

(4) function symbols, among which:
   - a) for each $n \geq 1$, a finite or countably infinite number of function symbols of type $(\text{OBJECT})^n \rightarrow$ STATOM.
   - b) a binary function symbol *state* of type TRACE x TIME $\rightarrow$ STATE
   - c) a binary function symbol *truth_value* of type STATE x STATOM $\rightarrow$ TRUTH_VALUE
   - d) a binary function symbol *and* of type STATPROP x STATPROP $\rightarrow$ STATPROP
     - the same for *or*, *implication* and *equivalence* function symbols
   - e) a unary function symbol *not* of type STATPROP $\rightarrow$ STATPROP
   - f) a binary function symbol *forall* of type STATE_VARIABLE x STATPROP $\rightarrow$ STATPROP
     - the same for the *exists* function symbol

(5) predicate symbols
   - a) a predicate symbol *holds* of type STATE x STATPROP.
   - b) =: TRACE x TRACE
   - c) =: STATE x STATE
   - d) =: TIME x TIME
   - e) <: TIME x TIME
   - f) $\subseteq$: TRACE x TRACE

Standard logical connectives

(1) standard connectives: $\vee$, $\wedge$, $\Leftrightarrow$, $\Rightarrow$, $\neg$

(2) standard quantifiers: $\forall$, $\exists$

(3) standard punctuation symbols:), (, ‘,’

A **sorted term** is defined as usually in sorted predicate logic.

TTL- formulae are defined inductively as follows:

A. The set of **atomic TTL-formulae** is defined as:

(1) If v_1 is a variable of sort STATE, and u_1 is a term of the sort STATPROP, then holds(v_1, u_1) is an atomic TTL formula.
(2) if t_1, t_2 are terms of sort TIME, then = (t_1, t_2) is an atomic TTL formula. (further we shall use this predicate in form $t_1 = t_2$)
(3) if t_1, t_2 are terms of sort TIME, then < (t_1, t_2) is an atomic TTL formula. (further we shall use this predicate in form $t_1 < t_2$, furthermore we shall use $t_1 \leq t_2$ for $t_1<t_2 \wedge t_1=t_2$)
(4) if γ_1, γ_2 are terms of sort TRACE, then = (γ_1, γ_2) is an atomic TTL formula. (further we shall use this predicate in form $\gamma_1= \gamma_2$)

(5) if s_1, s_2 are terms of sort STATE, then = (s_1, s_2) is an atomic TTL formula. (further we shall use this predicate in form $s_1 = s_2$)

B. The set of **well-formed TTL-formulae** is defined as

(1) Any atomic TTL-formula is a well-formed TTL-formula
If F and G are well-formed TTL-formulae, then so are ~F, $(F\vee G)$, $(F\wedge G)$, $(F\Rightarrow G)$ and $(F\Leftrightarrow G)$.
(2) If F is a well-formed TTL-formula containing x as a free variable, where x is a variable of any sort, then $(\forall x\ F)$ and $(\exists x\ F)$ are well-formed TTL-formulae.

The Semantics of TTL

An interpretation of a TTL formula is defined by the standard interpretation of order sorted predicate logic formulae. That is, the interpretation associates each sort symbol to a certain subdomain, such that a subdomain is included in another one, if the sort symbol designating the included subdomain is a subsort of the sort symbol designating the first one. Furthermore, an interpretation associates each function symbol with a mapping and each predicate symbol with a relation defined on the subdomains given by the interpretation for the respective sort name symbols and obtaining an element of the subdomain associated with the range sort name in the case of a functional symbol. Each variable assignment assigns a variable of a certain sort to an element of the subdomain associated with that sort. The semantics of connectives and quantifiers is defined in the standard way.

Subdomain of TRUTH_VALUE: true, false

Subdomain of TIME: N (the set of natural numbers, or the set of real numbers)

Subdomain of STATE: $\otimes$, $state_1$, …, $state_n$ (the set of state names)

Subdomain of TRACE: $trace_1$, …, $trace_n$ (the set of trace names)

Subdomain of STATOM: the set of state atom expressions

Subdomain of STATPROP: the set of state property expressions

Subdomain of STATE_INSTANCE: the set of state instances terms

Subdomain of STATE_VARIABLE: the set of state variables names

TTL Axioms

(1) Congruence of traces:
$\forall \gamma_1, \gamma_2\ [\forall t\ [state(\gamma_1, t) = state(\gamma_2, t)\ ] \Rightarrow \gamma_1 = \gamma_2]$

(2) Order relation between traces:
$\forall \gamma_1, \gamma_2\ [\forall t\ [state(\gamma_1, t) = state(\gamma_2, t) \vee state(\gamma_1, t) = \otimes\ ] \Rightarrow \gamma_1 \subseteq \gamma_2]$

(3) Equality of states:
$\forall s_1, s_2\ [\forall a{:}STATOMS\ [truth\_value(s_1, a) = truth\_value(s_2, a)] \Rightarrow s_1{=}s_2]$

(4) State property semantics
a. $holds(s, and(p_1, p_2)) \Leftrightarrow holds(s, p_1) \wedge holds(s, p_2)$
b. $holds(s, or(p_1, p_2)) \Leftrightarrow holds(s, p_1) \vee holds(s, p_2)$
c. $holds(s, not(p_1)) \Leftrightarrow \neg holds(s, p_1)$
d. $holds(s, exists(p_1, F)) \Leftrightarrow \exists p_1\ holds(s, F)$
e. $forall(s, forall(p_1, F)) \Leftrightarrow \forall p_1\ holds(s, F)$

(5) Partial order axioms for the time sort:
a. $\forall t\ t \leq t$
b. $\forall t_1, t_2\ [\ t_1 \leq t_2 \wedge t_2 \leq t_1\ ] \Rightarrow t_1 = t_2$
c. $\forall t_1, t_2, t_3\ [\ t_1 \leq t_2 \wedge t_2 \leq t_3\ ] \Rightarrow t_1 \leq t_3$

Properties of time

Uniqueness of time

This expresses that present_time(t) is true for at most one time point t:

$\forall t, t''\ state(\gamma, t) \models present\_time(t'') \Rightarrow \forall t', t' \neq t''\ \neg\ state(\gamma, t) \models present\_time(t')$

Correctness of time

This expresses that present_time(t) is true for the current time point t:

$\forall t\ state(\gamma, t) \models present\_time(t)$

Executable theories

For a given $\varphi(\gamma, t)$ ***the executable theory from observation states to memory states*** $Th_{o\to m}$ consists of the formulae:

For any state atom p occurring in $\varphi_p(\gamma, t)$, expressed in the InteractionOnt(A) for agent A:

$\forall t'$ state(γ, t') |= p $\Rightarrow$ state(γ, t') |= memory(t', p),

$\forall t''$ state(γ, t'') |= memory(t', p) $\Rightarrow$ state(γ, t''+1) |= memory(t', p),

state(γ, 0) |= present_time(0),

$\forall t$ state(γ, t) |= present_time(t) $\Rightarrow$ state(γ, t+1) |= present_time(t+1),

The last two rules are assumed to be included into the other theories $Th_{m\to p}$ and $Th_{p\to a}$ as well.

For a given $\varphi(\gamma, t)$ ***the executable theory from memory states to preparation states*** $Th_{m\to p}$ consists of the formulae:

For any state atom p occurring in $\varphi_{cond}(\gamma, t, t_1)$, expressed in the InteractionOnt(A) for agent A[3]::

$\forall t'$ state(γ, t') |= p $\Rightarrow$ state(γ, t') |= [memory(t', p) $\wedge$ stimulus_reaction(p)]

$\forall t'', t'$ state(γ, t'') |= memory(t', p) $\Rightarrow$ state(γ, t''+1) |= memory(t', p)

$\forall t'$ state(γ, t') |= q_{mem} $\Rightarrow$ state(γ, t') |= q_{cprep},

$\forall t', t$ state(γ, t') |= [q_{cprep} $\wedge$ q_{cond}(t) $\wedge$ $\bigcap_p$ stimulus_reaction(p)] $\Rightarrow$ state(γ, t') |= q_{prep},

$\forall t'$ state(γ, t') |= [stimulus_reaction(p) $\wedge\neg$ preparation_for(action(t'+c, a))] $\Rightarrow$ state(γ, t'+1) |= stimulus_reaction(p),

$\forall t'$ state(γ, t') |= [preparation_for(action(t'+c, a)) $\wedge\neg$performing_action(a)] $\Rightarrow$ state(γ, t'+c) |= preparation_for(action(t'+c, a)),

where a an action for which state(γ, t'+c) |= performing_action(a) occurs in $\varphi_f(\gamma, t)$.

For a given $\varphi_f(\gamma, t)$ ***the executable theory from the preparation to the action state(s)*** $Th_{p\to a}$ consists of the formula

$\forall t'$ state(γ, t') |= preparation_for(action(t'+c, a)) $\Rightarrow$ state(γ, t'+c) |= performing_action(a),

where c is a number and a an action for which state(γ, t'+c) |= performing_action(a) occurs in $\varphi_f(\gamma, t)$.

The transition rules obtained from the executable theories $Th_{o\to m}$, $Th_{m\to p}$, and $Th_{p\to a}$:

Time increment rules:

present_time(0) $\wedge\neg$p $\rightarrow$ present_time(1)

present_time(t) $\wedge\neg q_{mem}\wedge\neg$p $\rightarrow$ present_time(t+1)

present_time(t) $\wedge$ q_{cprep} $\wedge\neg q_{cond}$(t) $\wedge\neg$p $\rightarrow$ present_time(t+1)

present_time(t) $\wedge$ q_{prep} $\rightarrow$ present_time(t+1)

Memory state creation rule:

For any state atom p occurring in $\varphi_{cond}(\gamma, t, t_1)$, expressed in the InteractionOnt(A) for agent A:

present_time(t) $\wedge$ p $\rightarrow$ [memory(t, p) $\wedge$ stimulus_reaction(p)]

For all other state atoms p

present_time(t) $\wedge$ p $\rightarrow$ memory(t, p)

Memory persistence rule:

memory(t, p) $\rightarrow$ memory(t, p)

Conditional preparation generation rule:

q_{mem} $\rightarrow$ conditional_preparation_for(action(a)),

where a an action for which state(γ, t'+c) |= performing_action(a) occurs in $\varphi_f(\gamma, t)$.

[3] If a future formula does not contain a condition, then stimulus_reaction atoms are generated from a corresponding past formula

Preparation state creation rule:

$$\text{present\_time}(t') \wedge \text{conditional\_preparation\_for}(\text{action}(a)) \wedge q_{cond}(t) \wedge \bigcap_{p} \text{stimulus\_reaction}(p) \rightarrow \text{preparation\_for}(\text{action}(t'+c, a))$$

for every subformula of the form

$\text{present\_time}(t') \rightarrow \text{preparation\_for}(\text{action}(t'+c, a))$

that occurs in q_{cprep}.

Preparation state persistence rule:

$\text{preparation\_for}(\text{action}(t+c, a)) \wedge \neg\text{performing\_action}(a) \rightarrow \text{preparation\_for}(\text{action}(t+c, a))$

Stimulus reaction state persistence rule:

$\text{present\_time}(t') \wedge \text{stimulus\_reaction}(p) \wedge \neg\, \text{preparation\_for}(\text{action}(t'+c, a)) \rightarrow \text{stimulus\_reaction}(p)$

Action state creation rule:

$\text{preparation\_for}(\text{action}(t+c, a)) \wedge \text{present\_time}(t+c-1) \rightarrow \text{performing\_action}(a)$, where a is an action.

Lemmas, Propositions and Theorem

Lemma 1 (Normalisation lemma)

Let t be a given time point. If a formula δ(γ, t) only contains temporal relations such as t' < t'' and t' ≤ t'', and atoms of the form state(γ, t) |= p for some state formula p, then some state formula q(t) can be constructed such that δ(γ, t) is equivalent to the formula δ*(γ, t) of the form state(γ, t) |= q(t).

Proof sketch for Lemma 1.

First in the formula δ(γ, t) replace all temporal relations such as t' < t'' and t' ≤ t'' by state(γ, t) |= t' < t'' and state(γ, t) |= t' ≤ t'' respectively. Then proceed by induction on the composition of the formula δ(γ, t). Treat the logical connectives AND, OR, NOT, ⇒, ∀s, ∃s.

1) conjunction: δ(γ, t) is δ1(γ, t) AND δ2(γ, t)

By induction hypothesis

δ1(γ, t) ⇔ state(γ, t) |= p1 (which is δ1*(γ, t))

δ2(γ, t) ⇔ state(γ, t) |= p2 (which is δ2*(γ, t))

Then

δ(γ, t) ⇔ state(γ, t) |= p1 AND state(γ, t) |= p2 ⇔ state(γ, t) |= p1 ∧ p2 (which becomes δ*(γ, t))

2) disjunction: δ(γ, t) is δ1(γ, t) OR δ2(γ, t)

Again by induction hypothesis

δ1(γ, t) ⇔ state(γ, t) |= p1 (which is δ1*(γ, t))

δ2(γ, t) ⇔ state(γ, t) |= p2 (which is δ2*(γ, t))

Then

δ(γ, t) ⇔ state(γ, t) |= p1 OR state(γ, t) |= p2 ⇔ state(γ, t) |= p1 | p2 (which becomes δ*(γ, t))

3) negation: δ(γ, t) is ¬δ1(γ, t)

δ1(γ, t) ⇔ state(γ, t) |= p1

δ(γ, t) ⇔ ¬state(γ, t) |= p1

δ(γ, t) ⇔ state(γ, t) |= ¬p1 (which is δ*(γ, t))

4) implication: δ(γ, t) is δ1(γ, t) ⇒ δ2(γ, t)

Again by induction hypothesis

δ1(γ, t) ⇔ state(γ, t) |= p1 (which is δ1*(γ, t))

δ2(γ, t) ⇔ state(γ, t) |= p2 (which is δ2*(γ, t))

Then

δ(γ, t) ⇔ [state(γ, t) |= p1 ⇒ state(γ, t) |= p2] ⇔ state(γ, t) |= [p1 ⇒ p2] (which becomes δ*(γ, t))

5) universal quantifier:

$\delta(\gamma, t) \Leftrightarrow \forall t' \; state(\gamma, t) \models p1(t')$

$\delta(\gamma, t) \Leftrightarrow state(\gamma, t) \models \forall t' \; p1(t')$ (which is $\delta^*(\gamma, t)$)

6) existential quantifier:

$\delta(\gamma, t) \Leftrightarrow \exists t' \; state(\gamma, t) \models p1(t')$

$\delta(\gamma, t) \Leftrightarrow state(\gamma, t) \models \exists t' \; p1(t')$ (which becomes $\delta^*(\gamma, t)$)

Lemma 2

If time has properties of correctness and uniqueness, then

$$\varphi_{mem}(\gamma, t) \Leftrightarrow state(\gamma, t) \models q_{mem}(t) \Leftrightarrow state(\gamma, t) \models q_{mem} \tag{4}$$

Proof.

The proof for Lemma 2 follows directly from the Lemma 1, definitions of correctness and uniqueness of time and the definition of the formula q_{mem}. Lemmas 3, 4 and 5 can be proven in the same manner.

Proposition 1

Let $\varphi_p(\gamma, t)$ be a past statement for a given t, $\varphi_{mem}(\gamma, t)$ the memory formula for $\varphi_p(\gamma, t)$, $q_{mem}(t)$ the normalized memory state formula for $\varphi_{mem}(\gamma, t)$, and $Th_{o\to m}$ the executable theory from the interaction states for $\varphi_p(\gamma, t)$ to the memory states. Then,

$Th_{o\to m} \models [\varphi_p(\gamma, t) \Leftrightarrow \varphi_{mem}(\gamma, t)]$

and

$Th_{o\to m} \models [\varphi_p(\gamma, t) \Leftrightarrow state(\gamma, t) \models q_{mem}(t) \Leftrightarrow state(\gamma, t) \models q_{mem}]$.

Proof.

From the definitions of $q_{mem}(t)$ and of $Th_{o\to m}$ follows

$Th_{o\to m} \models [\varphi_p(\gamma, t) \Leftrightarrow \varphi_{mem}(\gamma, t)]$

Further by Lemma 2

$Th_{o\to m} \models [\varphi_p(\gamma, t) \Leftrightarrow state(\gamma, t) \models q_{mem}(t)]$ ■

Lemma 3

If time has properties of correctness and uniqueness, then

$$\varphi_{cmem}(\gamma, t, t_1) \Leftrightarrow state(\gamma, t_1) \models q_{cond}(t, t_1) \Leftrightarrow state(\gamma, t_1) \models q_{cond}(t) \tag{5}$$

Proof.

The lemma can be proven in the same manner as Lemma 2.

Lemma 4

If time has properties of correctness and uniqueness, then

$$\varphi_{prep}(\gamma, t_1) \Leftrightarrow state(\gamma, t_1) \models q_{prep}(t_1) \Leftrightarrow state(\gamma, t_1) \models q_{prep} \tag{6}$$

Proof.

The lemma can be proven in the same manner as Lemma 2.

Lemma 5

If time has properties of correctness and uniqueness, then

$$\varphi_{cprep}(\gamma, t) \Leftrightarrow state(\gamma, t) \models q_{cprep}(t) \Leftrightarrow state(\gamma, t) \models q_{cprep} \tag{7}$$

Proof.

The lemma can be proven in the same manner as Lemma 2.

Proposition 2

Let $\varphi_f(\gamma, t)$ be a future statement for t of the form $\forall t_1{>}t\ [\varphi_{cond}(\gamma, t, t_1) \Rightarrow \varphi_{act}(\gamma, t_1)]$, where $\varphi_{cond}(\gamma, t, t_1)$ is an interval statement, which describes a condition for one or more actions and $\varphi_{act}(\gamma, t_1)$ is a (conjunction of) future statement(s) for t_1, which describes action(s) that are to be performed; let $\varphi_{cprep}(\gamma, t)$ be the conditional preparation formula for $\varphi_f(\gamma, t)$, $q_{cprep}(t)$ be the normalized conditional preparation state formula for $\varphi_{cprep}(\gamma, t)$, and $Th_{m\rightarrow p}$ the executable theory for $\varphi(\gamma, t)$ from memory states to preparation states. Then,

$Th_{m\rightarrow p} \models [\varphi_f(\gamma, t) \Leftrightarrow \varphi_{cprep}(\gamma, t)]$

and

$Th_{m\rightarrow p} \models [\varphi_f(\gamma, t) \Leftrightarrow \text{state}(\gamma, t) \models q_{cprep}(t) \Leftrightarrow \text{state}(\gamma, t) \models q_{cprep}]$.

Proof.

From definition of $Th_{m\rightarrow p}$, Lemmas 3 and 4 follows that

$$Th_{m\rightarrow p} \models [\varphi_{cond}(\gamma, t, t_1) \Leftrightarrow q_{cond}(t, t_1)] \quad (8)$$

and

$$Th_{m\rightarrow p} \models [\varphi_{act}(\gamma, t_1) \Leftrightarrow q_{prep}(\gamma, t_1)] \quad (9)$$

From (8), (9), definitions of the conditional preparation formula and the normalized conditional preparation state formulae, and the conditions of the proposition it follows

$$Th_{m\rightarrow p} \models [\varphi_f(\gamma, t) \Leftrightarrow \forall t_1 > t\ [q_{cond}(t, t_1) \Rightarrow q_{prep}(\gamma, t_1)] \Leftrightarrow \forall t_1{>}t\ \varphi_{cprep}(\gamma, t, t_1) \Leftrightarrow \varphi_{cprep}(\gamma, t)]$$

And from Lemma 5 follows

$Th_{m\rightarrow p} \models [\varphi_f(\gamma, t) \Leftrightarrow \forall t_1{>}t\ \text{state}(\gamma, t) \models q_{cprep}(t, t_1) \Leftrightarrow \text{state}(\gamma, t) \models q_{cprep}(t) \Leftrightarrow \text{state}(\gamma, t) \models q_{cprep}]$ ■

Proposition 3

Let $\varphi_p(\gamma, t)$ be a past statement for t and $\varphi_f(\gamma, t)$ be a future statement for t. Let $\varphi_{mem}(\gamma, t)$ be the memory formula for $\varphi_p(\gamma, t)$ and $\varphi_{cprep}(\gamma, t)$ the conditional preparation formula for $\varphi_f(\gamma, t)$. Then

$$[\varphi_p(\gamma, t) \Rightarrow \varphi_f(\gamma, t)] \Leftrightarrow [\varphi_{mem}(\gamma, t) \Rightarrow \varphi_{cprep}(\gamma, t)]$$

Proof.

From the Proposition 1 and the Proposition 2 follows

$$\varphi_p(\gamma, t) \Leftrightarrow \varphi_{mem}(\gamma, t) \text{ and } \varphi_f(\gamma, t) \Leftrightarrow \forall t_1{>}t\ \varphi_{cprep}(\gamma, t, t_1)$$

Then,

$$[\varphi_p(\gamma, t) \Rightarrow \varphi_f(\gamma, t)] \Leftrightarrow [\varphi_{mem}(\gamma, t) \Rightarrow \forall t_1{>}t\ \varphi_{cprep}(\gamma, t, t_1)]$$

So it has been proven that $[\varphi_p(\gamma, t) \Rightarrow \varphi_f(\gamma, t)] \Leftrightarrow [\varphi_{mem}(\gamma, t) \Rightarrow \varphi_{cprep}(\gamma, t)]$ ■

Two traces γ_1, γ_2 *coincide* on ontology Ont (denoted by coincide_on(γ_1, γ_2, Ont)) iff

$$\forall t\ \forall a \in At(Ont) \quad \text{state}(\gamma_1, t) \models a \Leftrightarrow \text{state}(\gamma_2, t) \models a$$

Let $\varphi(\gamma, t)$ be an externally observable dynamic property for agent A. An executable specification $\pi(\gamma, t)$ for A *refines* $\varphi(\gamma, t)$ iff

(1) $\forall \gamma, t\ \ \pi(\gamma, t) \Rightarrow \varphi(\gamma, t)$

(2) $\forall \gamma_1, t\ [\varphi(\gamma_1, t) \Rightarrow [\exists \gamma_2\ \text{coincide\_on}(\gamma_1, \gamma_2, \text{InteractionOnt}(A)) \wedge \pi(\gamma_2, t)]]$

Lemma 6

Let $\varphi(\gamma, t)$ be a dynamic property expressed using the state ontology Ont. Then the following holds:

(1) $\text{coincide\_on}(\gamma_1, \gamma_2, \text{Ont}) \wedge \text{coincide\_on}(\gamma_2, \gamma_3, \text{Ont}) \Rightarrow \text{coincide\_on}(\gamma_1, \gamma_3, \text{Ont})$

(2) $\text{coincide\_on}(\gamma_1, \gamma_2, \text{Ont}) \Rightarrow [\varphi(\gamma_1, t) \Leftrightarrow \varphi(\gamma_2, t)]$.

Proof sketch.

The transitivity property (1) follows directly from the definition of coinciding traces for coincide_on(γ_1, γ_2, Ont) and coincide_on(γ_2, γ_3, Ont):

$\forall a \in At(Ont)\ \ \forall t'\ \ [\text{state}(\gamma_1, t') \models a \Leftrightarrow \text{state}(\gamma_3, t') \models a] \Rightarrow \text{coincide\_on}(\gamma_1, \gamma_3, \text{Ont})$

From

$\forall \gamma_1, \gamma_2\ [\text{coincide\_on}(\gamma_1, \gamma_2, \text{Ont}) \Rightarrow \forall t'\ [\varphi_p(\gamma_1, t') \Leftrightarrow \varphi_p(\gamma_2, t') \text{ AND } \varphi_f(\gamma_1, t') \Leftrightarrow \varphi_f(\gamma_2, t')]]$

follows that $\varphi(\gamma_1, t) \Leftrightarrow \varphi(\gamma_2, t)$.

Note that for any past interaction statement $\varphi_p(\gamma, t)$ and future interaction statement $\varphi_f(\gamma, t)$ the following holds:

$\forall \gamma1, \gamma2$ [coincide_on($\gamma1, \gamma2$, InteractionOnt) $\Rightarrow$ [$\varphi_p(\gamma1, t) \Leftrightarrow \varphi_p(\gamma2, t) \wedge \varphi_f(\gamma1, t) \Leftrightarrow \varphi_f(\gamma2, t)$]]

Theorem

If the executable specification $\pi_A(\gamma, t)$ refines the external behavioral specification $\varphi_A(\gamma, t)$ of agent A, and $\psi(\gamma, t)$ is a dynamic interaction property of agent A in its environment, expressed using the interaction ontology InteractionOnt(A), then

$$[\forall\gamma\ [\ \pi_A(\gamma, t) \Rightarrow \psi(\gamma, t)\ ]\ ] \Leftrightarrow [\forall\gamma\ [\ \varphi_A(\gamma, t) \Rightarrow \psi(\gamma, t)\ ]\ ]$$

Proof sketch for Theorem.

$\Leftarrow$ is direct:

from $\pi_i(\gamma, t) \Rightarrow \varphi_i(\gamma, t)$ and $\wedge\varphi_i(\gamma, t) \Rightarrow \psi(\gamma, t)$ it follows $\wedge\pi_i(\gamma, t) \Rightarrow \psi(\gamma, t)$.

$\Rightarrow$ runs as follows:

Suppose $\varphi_i(\gamma, t)$ holds for all i, then since $\pi_1(\gamma)$ refines $\varphi_1(\gamma, t)$, then according to the definition of refinement of an externally observable property exists such a γ_1 that $\pi_1(\gamma_1)$ and coincide_on(γ, γ_1, InteractionOnt (A)).

Due to Lemma 6, this γ_1 still satisfies all $\varphi_i(\gamma_1, t)$ (i.e., $\varphi_i(\gamma_1, t)$ holds for all i).

Proceed with γ_1 to obtain a γ_2 and further for all i to reach a trace γ_n, for which

$\pi_i(\gamma_n)$ holds for all i,

and

coincide_on(γ, γ_n, InteractionOnt(A)),

and

$\varphi_i(\gamma_n)$ holds for all i.

From

$$\forall\gamma\ \forall i\ [\pi_i(\gamma) \Rightarrow \varphi_i(\gamma)],$$

and

$$\forall\gamma\ [\wedge\pi_i(\gamma) \Rightarrow \psi(\gamma, t)\ ]$$

it follows that $\forall\gamma\ \wedge\varphi_i(\gamma) \Rightarrow \psi(\gamma)$.

So it has been proven that $\forall\gamma\ \wedge\varphi_i(\gamma) \Rightarrow \psi(\gamma)$. ■

Appendix C: External, executable internal and transition system specifications for the example of classical conditioning for *Aplysia Californica's* defensive withdrawal reflex

External specification

Dynamic Property 1: ∃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)] ⇒

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

Dynamic Property 2: ∀t9 ≤ t state(γ, t9, input(aplysia)) |= observed(tail_shock) ⇒ state(γ, t9+c, output(aplysia)) |= performing_action(contraction)

Executable specification

∀t' state(γ, t', input(aplysia)) |= observed(touch_siphon) ⇒ state(γ, t', internal(aplysia)) |= [memory(t', observed(touch_siphon)) ∧ stimulus_reaction(observed(touch_siphon))]

∀t' state(γ, t', input(aplysia)) |= observed(tail_shock) ⇒ state(γ, t', internal(aplysia)) |= [memory(t', observed(tail_shock)) ∧ stimulus_reaction(observed(tail_shock))]

∀t'' state(γ, t'', internal(aplysia)) |= memory(t', observed(touch_siphon)) ⇒
state(γ, t''+1, internal(aplysia)) |= memory(t', observed(touch_siphon))

∀t'' state(γ, t'', internal(aplysia)) |= memory(t', observed(tail_shock)) ⇒ state(γ, t''+1, internal(aplysia)) |= memory(t', observed(tail_shock))

∀t' state(γ, t') |= ∀t'' [present_time(t'') →
[∃t2, t3, t4, t5, t6, t7 [t2 < t3 ∧ t3 < t4 ∧ t4 < t5 ∧ t5 < t6 ∧ t6 < t7 ∧ t7 < t'' ∧ memory(t2, observed(touch_siphon)) ∧ memory(t3, observed(tail_shock)) ∧ memory(t4, observed(touch_siphon)) ∧ memory(t5, observed(tail_shock)) ∧ memory(t6, observed(touch_siphon)) ∧ memory(t7, observed(tail_shock))]] ⇒
state(γ, t') |= ∀t''' [present_time(t''') → [∀t8 > t''' [memory(t8, observed(touch_siphon)) → preparation_for(action(t8+c, contracts))]]]

∀t' state(γ, t') |= [∀t''' [present_time(t''') → [∀t8 > t''' [memory(t8, observed(touch_siphon)) →
preparation_for(action(t8+c, contracts))]]] ∧
∀t'' [present_time(t'') → memory(t'', observed(touch_siphon))] ∧ stimulus_reaction(observed(touch_siphon))] ⇒
state(γ, t', internal(aplysia)) |= ∀t8 [present_time(t8) →
preparation_for(action(t8+c, contracts))]

∀t' state(γ, t') |= [stimulus_reaction(observed(touch_siphon)) ∧ not(preparation_for(action(t'+c, contracts)))] ⇒
state(γ, t'+1) |= stimulus_reaction(observed(touch_siphon))

∀t' state(γ, t') |= [stimulus_reaction(observed(tail_shock)) ∧ not(preparation_for(action(t'+c, contracts)))] ⇒
state(γ, t'+1) |= stimulus_reaction(observed(tail_shock))

∀t' state(γ, t', internal(aplysia)) |= preparation_for(action(t'+c, contracts)) ⇒
state(γ, t'+c, output(aplysia)) |= performing_action(contracts)

∀t' state(γ, t', internal(aplysia)) |= [preparation_for(action(t'+c, contracts)) ∧ not(performing_action(contracts))] ⇒
state(γ, t'+1, internal(aplysia)) |= preparation_for(action(t'+c, contracts))

∀t' state(γ, t') |= [[∀t'' [present_time(t'') → ∀t9 ≤ t'' memory(t9, observed(tail_shock))] ∧ stimulus_reaction(observed(tail_shock))] ⇒
state(γ, t', internal(aplysia)) |= preparation_for(action(t9+c, contracts))]

∀t' state(γ, t', world) |= touch_siphon ⇒ state(γ, t', input(aplysia)) |= observed(touch_siphon)

∀t' state(γ, t', world) |= tail_shock ⇒ state(γ, t', input(aplysia)) |= observed(tail_shock)

Transition system

touch_siphon ↠ observed(touch_siphon)

tail_shock ↠ observed(tail_shock)

present_time(t) ∧ observed(touch_siphon) ↠ [memory(t, observed(touch_siphon)) ∧ stimulus_reaction(observed(touch_siphon))]

present_time(t) ∧ observed(tail_shock) ↠ [memory(t, observed(tail_shock)) ∧ stimulus_reaction(observed(tail_shock))]

memory(t, observed(touch_siphon) ↠ memory(t, observed(touch_siphon))

memory(t, observed(tail_shock) ↠ memory(t, observed(tail_shock))

present_time(t) ∧ ∃t2, t3, t4, t5, t6, t7 [t2 < t3 ∧ t3 < t4 ∧ t4 < t5 ∧ t5 < t6 ∧ t6 < t7 ∧ t7 < t ∧ memory(t2, observed(touch_siphon)) ∧ memory(t3, observed(tail_shock)) ∧ memory(t4, observed(touch_siphon)) ∧ memory(t5, observed(tail_shock)) ∧ memory(t6, observed(touch_siphon)) ∧ memory(t7, observed(tail_shock))] ↠ conditional_preparation_for(action(contracts))

present_time(t) ∧ ∀t9 ≤ t ∧ memory(t9, observed(tail_shock)) ↠ conditional_preparation_for(action(contracts))

present_time(t) ∧ conditional_preparation_for(action(contracts)) ∧ memory(t8, observed(touch_siphon)) ∧

stimulus_reaction(observed(touch_siphon)) ↠ preparation_for(action(t+c, contracts))

present_time(t) ∧ conditional_preparation_for(action(contracts)) ∧ stimulus_reaction(observed(tail_shock)) ↠

preparation_for(action(t+c, contracts))

present_time(t) ∧ stimulus_reaction(observed(touch_siphon)) ∧ not(preparation_for(action(t+c, contracts))) ↠

stimulus_reaction(observed(touch_siphon))

present_time(t) ∧ stimulus_reaction(observed(tail_shock)) ∧ not(preparation_for(action(t+c, contracts))) ↠

stimulus_reaction(observed(tail_shock))

preparation_for(action(t+c, contracts)) ∧ not(performing_action(contracts)) ↠ preparation_for(action(t+c, contracts)).

preparation_for(action(t+c, contracts)) ∧ present_time(t+c-1) ↠ performing_action(contracts).

Appendix D: Complete simulation trace and a graphical model for the example of classical conditioning for *Aplysia Californica's* defensive withdrawal reflex

Complete simulation trace for the scenario, considered in the paper

0: present_time(0) world(0,touch_siphon)	24: not(preparation_for(action(8, contracts)))
1: not(world(0,touch_siphon)) observed(0,touch_siphon)	25: present_time(9) world(9,touch_siphon)
2: memory(observed(0,touch_siphon)) not(observed(0,touch_siphon)) stimulus_reaction(observed(touch_siphon))	26: not(world(9,touch_siphon)) observed(9,touch_siphon)
3: present_time(1) world(1,tail_shock)	27: memory(observed(9,touch_siphon)) not(observed(9,touch_siphon)) stimulus_reaction(observed(touch_siphon))
4: not(world(1,tail_shock)) observed(1,tail_shock)	28: present_time(10) world(10,tail_shock)
5: memory(observed(1,tail_shock)) not(observed(1,tail_shock)) stimulus_reaction(observed(tail_shock))	29: not(world(10,tail_shock)) observed(10,tail_shock)
6: conditional_preparation_for(action(contracts))	30: memory(observed(10,tail_shock)) not(observed(10,tail_shock)) stimulus_reaction(observed(tail_shock))
7: preparation_for(action(3,contracts))	31: conditional_preparation_for(action(contracts))
8: not(stimulus_reaction(observed(touch_siphon))) not(stimulus_reaction(observed(tail_shock)))	32: preparation_for(action(12,contracts))
9: present_time(2)	33: not(stimulus_reaction(observed(touch_siphon))) not(stimulus_reaction(observed(tail_shock)))
10: present_time(3) performing_action(contracts)	34: present_time(11)
11: not(preparation_for(action(3, contracts)))	35: present_time(12) performing_action(contracts)
12: present_time(4)	36: not(preparation_for(action(12,contracts)))
13: present_time(5) world(5,touch_siphon)	37: present_time(13)
14: not(world(5,touch_siphon)) observed(5,touch_siphon)	38: present_time(14)
15: memory(observed(5,touch_siphon)) not(observed(5,touch_siphon)) stimulus_reaction(observed(touch_siphon))	39: present_time(15) world(15,touch_siphon)
16: present_time(6) world(6,tail_shock)	40: not(world(15,touch_siphon)) observed(15,touch_siphon)
17: not(world(6,tail_shock))	41: memory(observed(15,touch_siphon))

observed(6,tail_shock)	not(observed(15,touch_siphon)) stimulus_reaction(observed(touch_siphon))
18: memory(observed(6,tail_shock)) not(observed(6,tail_shock)) stimulus_reaction(observed(tail_shock))	42: preparation_for(action(17,contracts))
19: conditional_preparation_for(action(contracts))	43: not(stimulus_reaction(observed(touch_siphon))) not(stimulus_reaction(observed(tail_shock)))
20: preparation_for(action(8,contracts))	44: present_time(16)
21: not(stimulus_reaction(observed(touch_siphon))) not(stimulus_reaction(observed(tail_shock)))	45: present_time(17) performing_action(contracts)
22: present_time(7)	46: not(preparation_for(action(17,contracts)))
23: present_time(8) performing_action(contracts)	47: present_time(18)

Graphical model