A Formal Analysis Method for Cognitive Agent Behavior

Alexei Sharpanskykh and Jan Treur
Department of Artificial Intelligence, Vrije Universiteit Amsterdam,
De Boelelaan 1081a, NL-1081 HV Amsterdam, The Netherlands
{sharp, treur}@few.vu.nl

Abstract

This paper contributes a formal method to analyze consequences of agent behavior. The method is based on an automated procedure to translate a given external behavioral specification of an agent into an executable specification of internal dynamics. Having an executable internal dynamics specification allows automated analysis of the consequences of the agent's behavior, based on model checking techniques. By a paradigmatic example it is shown how the developed analysis method is applied.

1. Introduction

For cognitive agents, behavior can be described 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 input and output states over time of the agent, without any reference to internal or mental states of the agent. Such a view is considered within the philosophical perspective of behaviorism [6]. A specification of externally observable behavior has the advantage that it can be directly related to empirical information, but may have a high complexity. From the internal perspective the behavior of the agent can be characterized by a specification of more direct (causal) temporal relations between observations, mental states, and actions of the agent, based on which an externally observable behavioral pattern is generated. Such a perspective is taken within the functionalist tradition [6]. The direct causal temporal relations between observation states, mental states and action states are executable so that they can more easily be used for analysis of the agent's behavior, but to a certain extent may have a hypothetical status, as they are difficult to observe.

For a given specification of externally observable behavior in different environmental circumstances, an interesting but not easily solvable problem is how to determine the (logical) consequences of this behavior. For example, if an animal has a certain behavioral repertoire with respect to different food-related circumstances in the environment, in how far will this repertoire entail its well-being, i.e., in how far the animal will become satisfied and healthy due to this behavioral repertoire. Within Computer Science, quite useful and efficient model checking techniques have been developed to determine consequences of a given behavioral specification; e.g., [2]. By performing model checking it is possible to determine automatically if a model, usually specified as a finite state transition system, entails some dynamic property, specified by temporal formulae. Given an appropriate model of the agent behavior, by such techniques and the supporting software, for different (environmental) scenarios, the consequences of the behavior of the agent in its environment can be analyzed, for example to find out whether its behavioral repertoire is adequate for its well-being.

However, to be able to use such techniques, a limitation is that this behavioral specification has to be given in a simple, executable format (as a transition system). Usually a specification of externally observable agent behavior consists of more complex temporal relations expressed as dynamic properties in non-causal (and also non-executable) logical format, which does not allow direct application of automated analysis. To achieve automated analysis based on model checking, an external behavioral description has to be replaced by one in a simpler logical format, closer to a finite state transition system format required by model checking techniques. The transformation of the external behavioral specification for the agent into an internal behavioral specification in executable format, as described in this paper, solves this problem. More specifically, it is shown how a translation of a specification of externally observable behavior into the model specification format for the SMV model checker [10] can be applied to enable SMV to be used for automated analysis. SMV uses efficient algorithms to analyze such a model and the temporal logic CTL is used for properties (e.g., properties concerning well-being) to check.

In the next section the concepts for formal specification of dynamics of an agent are introduced. After that in Section 3 the transformation procedure from a specification of externally observable agent behavior, first into executable format and then into a finite state transition system description is described and illustrated by means of an example. The application of the proposed procedure is demonstrated by a paradigmatic example in Section 4. The paper ends with a discussion.

2. Modeling 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

An agent state at a certain point in time as used here is an indication of which of the state properties of the agent and its environment (e.g., observations and actions) are true (hold) at that time point. Externally observable state properties of the agent are formalized using the interaction state 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 functions and 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.

2.2. Expressing Dynamic Properties

To characterize the dynamics of the agent, dynamic properties relate properties of states at certain points in time. Consider this 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, at which agent A goes to p."

To express such dynamic properties, and other, more sophisticated ones, the Temporal Trace Language (TTL) is used; for formal description of TTL syntax and semantics see [14]. TTL is a variant of order-sorted predicate logic [9] and has some similarities with situation calculus [12] and event calculus [7].

The language TTL includes special sorts, such as: TIME (a set of linearly ordered time points), STATE (a set of all state names of an agent system), TRACE (a set of all trace names; a trace or a trajectory can be thought of as a timeline with for each time point a state), and STATPROP (a set of all state property names). Further in the paper we shall use t with subscripts and superscripts for variables of the sort TIME; and γ with subscripts and superscripts for variables of the sort TRACE.

A state of an agent is related to a state property via the satisfaction relation |= formally defined as a binary infix predicate (or by holds as a binary prefix predicate). 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. Sometimes, when the indication of an agent aspect is not essential, this relation will be used without the third argument: state(γ, t) |= p.

Both state(γ , t, output(A)) and p are terms of the TTL language. In general, TTL terms are constructed by induction in a standard sorted predicate logic way from variables, constants and functional symbols typed with TTL sorts. Dynamic properties are expressed by TTL-formulae defined by:

(1) If v_1 is a term 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 τ_1, τ_2 are terms of any TTL sort, then $\tau_1 = \tau_2$ is an atomic TTL formula.
(3) If t_1, t_2 are terms of sort TIME, then $t_1 < t_2$ is an atomic TTL formula.
(4) The set of well-formed TTL-formulae is defined inductively in a standard way based on atomic TTL-formulae using boolean connectives and quantifiers.

2.3. External Behavior Specification

Typically a behavior specification from an external perspective consists of a number of dynamic properties that express how the agent copes with different situations it encounters in its environment. Here situations are meant as sequences of events extending over time. A simple example is the following dynamic property

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

which is expressed in formalized form as:

$\forall$t1 [[state(γ, t1, input(A)) |= observed(dark_in_room) &
$\exists$t0<t1 [state(γ, t0, input(A)) |= observed(light_on)]
$\Rightarrow$ $\exists$t2 $\geq$ t1 state(γ, t2, output(A)) |=
performing_action(switch_on_light)]

A external behavioral specification φ consists of a number of dynamic properties which have a format (see Section 3) based on temporal patterns described by a past statement, interval statement and a future statement. For simplicity, the future part has a format that prevents non-determinism in behavior.

a) A *past statement* for a trace γ and a time point t over state ontology Ont is a temporal statement $\varphi_p(\gamma,t)$, such that each time variable different from t is restricted to the time before t: 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) A *future statement* for a trace γ and a time point t over state ontology Ont is a temporal statement $\varphi_f(\gamma,t)$, 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 made.

c) 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, that is a past statement for t_2 and a future statement for t_1.

2.4. Internal Dynamics Specification

An executable specification of the internal dynamics of an agent, consists of a set of executable dynamic properties, representing temporal relations between a number of postulated internal (or mental) states. Internal states of an agent A are described using a postulated internal state ontology InternalOnt(A). It is often assumed that an agent maintains a memory in the form of some internal model of the history; some even go that far to speculate that intelligence is mainly based on that; cf. [4]. It is assumed that internal memory states are formed based on sensing (or observation). The state ontology InternalOnt includes sorts and functions for defining memories about input states. Notice that within such states representing memory, statements about time are made. To relate time within such a state property to time external to states, the function symbol present_time is used. Here the properties of correctness and uniqueness are assumed:

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:

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

As an example, memory(t, observed(a)) expresses that the agent has memory that it observed at time point t a state property a.

Furthermore, it is postulated that before performing an action an agent creates an internal preparation state. For example, preparation_for(b) specifies preparation of an agent to perform 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 the internal representations.

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

Table 1: Executable Format

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)* $\forall t \; state(\gamma, t) \models X \wedge C$	**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 $\Rightarrow \; state(\gamma, t) \models Y$

3. Transformation into Executable Format

This section describes the transformation of a specification for externally observable agent behavior into executable format and subsequently into the representation of a finite state transition system.

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 and $\varphi_f(\gamma, t)$ is a future statement over the interaction ontology. 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 $state(\gamma, t_1+c) \models$ 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 (conditional) 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, expressed using ontology InteractionOnt(A), for which an executable representation should be found, then the transformation procedure is specified as follows.

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 the specification $\pi(\gamma, t)$, which describes the internal dynamics of agent A, corresponding to the property $\varphi(\gamma, t)$. Also add the dynamic properties, which were initially specified in executable form using an ontology, different than InteractOnt(A).
(5) Translate the executable properties collected in step 4 into the transition system representation.

The following theorem shows how this transformation procedure can be used to determine logical consequences

of an external behavior specification: the logical consequences of a certain external behaviour specification are the logical consequences of the corresponding internal specification, and the latter consequences are easier to determine because of the simpler format of the internal dynamics specification.

Theorem[1]

If the internal dynamics specification $\pi(\gamma, t)$ corresponds (by the transformation above) to the external behavioral specification $\varphi(\gamma, t)$, and $\psi(\gamma, t)$ is a dynamic property of the agent in its environment, then $\psi(\gamma, t)$ is entailed by $\varphi(\gamma, t)$ if and only if $\psi(\gamma, t)$ is entailed by $\pi(\gamma, t)$:

$\forall\gamma\ [\pi(\gamma, t) \Rightarrow \psi(\gamma, t)] \quad \Leftrightarrow \quad \forall\gamma\ [\varphi(\gamma, t) \Rightarrow \psi(\gamma, t)]$

The details of the procedure are described by means of an example, in which a delayed-response behavior of an animal (e.g., a laboratory mouse) is investigated; e.g., [1], [5], and [15].

The initial situation for the described 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 experiment consists of environmental properties and externally observable behavioral properties of the mouse. To illustrate the proposed transformation procedure a dynamic property that describes delayed-response behavior of the mouse has been chosen. According to the definition of an external behavioral specification the considered property $\varphi(\gamma, t)$ has the form $[\varphi_p(\gamma, t) \Rightarrow \varphi_f(\gamma, t)]$; here for the example $\varphi_p(\gamma, t)$ is a formula expressing

'at some point t2 in the past the mouse observed food and for all time points since then it did not observe the absence of food ',

or, formally:

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

Moreover, $\varphi_f(\gamma, t) \Leftrightarrow \forall t_1 > t\ [\varphi_{cond}(\gamma, t, t_1) \Rightarrow \varphi_{act}(\gamma, t_1)]$, with $\varphi_{cond}(\gamma, t, t1)$ is a formula expressing

'at t1 the mouse observes that there is no screen and since t the mouse did not observe the absence of food',

or, formally

$\text{state}(\gamma, t1, \text{input(mouse)}) \models \text{observed(not(screen))} \wedge$
$\forall t5,\ t1 \geq t5 > t\ \text{state}(\gamma, t5, \text{input(mouse)}) \models$
$\text{not(observed(not(food)))},$

and $\varphi_{act}(\gamma, t1)$ is a formula expressing that

'at t1+c the mouse goes to the position with food',

or, formally

$\text{state}(\gamma, t1+c, \text{output(mouse)}) \models \text{performing\_action(goto\_food)},$

where t is the present time point with respect to which the formulae are evaluated and c is some number.

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 $\text{state}(\gamma, t') \models p$ by $\text{state}(\gamma, t) \models \text{memory}(t', p)$ is called the *memory formula for* $\varphi_p(\gamma, t)$. Thus, a memory formula defines a sequence of past events (e.g., observations, actions) for the present time point t. The time duration for generation of an internal memory state from the event is assumed to be much smaller than time intervals between external events (i.e., stimuli). Therefore, both an observation state and a corresponding memory state are created in the model at the same time point. For example,

'at some point t2 in the past the agent observed food and for all time points since then it did not observe the absence of food '

is transformed into

'the agent has a memory state that at some point t2 in the past it observed food and for all time points since then it has memory states that it did not observe the absence of food'

According to Lemma 1 (details are given in [14]) $\varphi_{mem}(\gamma, t)$ is equivalent to some formula $\delta^*(\gamma, t)$ of the form $\text{state}(\gamma, t) \models q_{mem}(t)$, where $q_{mem}(t)$ is called *the normalized memory state formula for* $\varphi_{mem}(\gamma, t)$, which describes the memory state at time point t. Moreover, q_{mem} is the state formula $\forall t'\ [\text{present\_time}(t') \Rightarrow q_{mem}(t')]$.

In continuation of the example considered above $q_{mem}(t)$ for $\varphi_{mem}(\gamma, t)$ is defined as:

'the agent has a (complex) mental state describing that it has memory that at some point t2 in the past it observed food and for all time points since then it has memory that it did not observe the absence of food'

For this example $q_{mem}(t)$ for $\varphi_{mem}(\gamma, t)$ is formally specified as:

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

Additionally, memory state persistency properties are included for all memory atoms:

'if at time t1 the agent has a memory,
then also at time t1+1 it will have this memory.'

For the atom memory(t2, observed(food)) the persistency property is formally defined as:

$\forall t''\ \text{state}(\gamma, t'', \text{internal(mouse)}) \models \text{memory}(t', \text{observed(food)}) \Rightarrow$
$\text{state}(\gamma, t''+1, \text{internal(mouse)}) \models \text{memory}(t', \text{observed(food)})$

[1] The proof for this theorem and other formal details for this paper are given in [14]

Rules that describe creation and persistence of memory atoms are included in *the executable theory from observation states to memory states* $Th_{o\to m}$ (for a detailed description of this and the following theories we refer to [14]). 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***

Similarly to the step from past formula to memory formula, the condition formula is translated into a memory formula.

Moreover, 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. For example,

'at t1+c the agent performs action a'

is replaced by

'at t1 the agent is prepared to perform the action at t1+c'

The preparation state is created at time point t_1. For the considered example q_{prep}(t1) is composed as preparation_for(action(t1+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 included in *the executable theory from memory states to preparation states* $Th_{m\to p}$. 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 recurring actions.

Step 3. ***From preparation states to action state(s)***

The preparation state preparation_for(action(t_1+c, a)) is followed by the action state, created at 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:

'if at t' the agent has the preparation state to perform action a at time t'+c,
then at time t'+c the agent performs the action a'.

Formally:

∀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. Moreover, executable properties generating observation states from the states of the external world are added. For the considered example,

'if in the world there is food and no cup,
then the agent observes the presence of food'.

Formally:

∀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 5. ***Translation of an executable specification into a description of a transition system***

For the purpose of practical verification a behavioral specification π based on executable temporal logical properties generated at Step 4 is translated into a finite

state transition system description τ. A finite state transition system is described by a tuple $\langle Q, Q_0, \Sigma, \rightarrow\rangle$, 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. For translating the executable specification of agent behavior into the transition system representation the function present_time(t) introduced earlier is used. This function maps to a truth value true in a state for the current time point t only. The executable properties from the executable specification, translated into the transition rules for the considered example are given below:

food $\wedge$ not(cup) $\rightarrow$ observed(food)

not(food) $\wedge$ not(cup) $\rightarrow$ observed(not(food))

screen $\rightarrow$ observed(screen)

not(screen) $\rightarrow$ observed(not(screen))

present_time(t) $\wedge$ observed(food) $\rightarrow$ memory(t, observed(food))

present_time(t) $\wedge$ not(observed(not(food))) $\rightarrow$
memory(t, not(observed(not(food))))

present_time(t) $\wedge$ observed(not(food)) $\rightarrow$
memory(t, observed(not(food)))

present_time(t) $\wedge$ observed(not(screen)) $\rightarrow$
memory(t, observed(not(screen))) $\wedge$
stimulus_reaction(observed(not(screen)))

memory(t, observed(food)) $\rightarrow$ memory(t, observed(food))

memory(t, not(observed(not(food)))) $\rightarrow$
memory(t, not(observed(not(food))))

memory(t, observed(not(food))) $\rightarrow$
memory(t, observed(not(food)))

memory(t, observed(not(screen))) $\rightarrow$
memory(t, observed(not(screen)))

present_time(t) $\wedge$ $\exists$t2 [memory(t2, observed(food)) $\wedge$
$\forall$t3 t $\geq$ t3 > t2 $\rightarrow$ memory(t3, not(observed(not(food))))] $\rightarrow$
conditional_preparation_for(action(goto_food))

present_time(t) $\wedge$ conditional_preparation_for(action(goto_food))
$\wedge$ memory(t, observed(not(screen))) $\wedge$
stimulus_reaction(observed(not(screen))) $\rightarrow$
preparation_for(action(t+c, goto_food))

present_time(t) $\wedge$ stimulus_reaction(observed(not(screen))) $\wedge$
not(preparation_for(action(t+c, goto_food))) $\rightarrow$
stimulus_reaction(observed(not(screen)))

preparation_for(action(t+c, goto_food)) $\wedge$
not(performing_action(goto_food)) $\rightarrow$
preparation_for(action(t+c, goto_food))

preparation_for(action(t+c, goto_food)) $\wedge$ present_time(t+c-1) $\rightarrow$
performing_action(goto_food).

For automatic verification of general properties of the agent in its environment by means of the model checking tool SMV, a representation of a finite state transition system corresponding to the agent behavioral specification is translated into the input format of the SMV model checker. For the complete description of the translation procedure we refer to [14].

The described transformation procedure was implemented in Java, with an input (an external behavioral specification) and an output (an executable specification and finite transition system descriptions) files specified in textual format.

4. The Analysis Method and its Application

In this section the proposed analysis method is described and illustrated by an example analysis from the studies of animal behavior. In this example we investigate two cases of laboratory mouse behavior (i.e., delayed-response behavior specified by φ_1, which is the property φ shown in Section 3, and motivation-based behavior, specified by φ_2) in two different environmental experimental settings (E, resp. E') with an identical initial situation, described in the previous section. By means of the introduced approach it is determined for each of the environmental settings and each type of behavior whether the combination will bring the agent well-being.

The Analysis Method

(a) The external behavioral specifications φ_1 and φ_2 are formally defined (more detailed formal specifications for the analysis are given in [14]).

(b) By means of the translation procedure described in Section 3, each external behavioral specification φ_i is automatically translated into its corresponding executable internal dynamics specification π_i and its related state transition system representation τ_i.

(c) The well-being properties ψ and ψ' to be checked are specified in CTL

(d) Using the state transition system representations τ_i, verification of each of the agent models with respect to properties ψ and ψ' is performed in the SMV model checker, resulting in confirmed or rejected entailment relations between the π_i and ψ and ψ'.

(e) Based on the theorem in Section 3 the confirmed or rejected entailment relations between the π_i and ψ and ψ' imply corresponding confirmed or rejected entailment relations between the φ_i and ψ and ψ'.

For this example, ψ is the following conditional well-being property (which is expressed conditionally for environmental condition E):

for all traces, if the screen is removed and food is hidden under the cup, then the mouse will eventually be satisfied.

This property ψ can be expressed in Computation Tree Logic (CTL) [2] required for verification in the SMV model checking tool as follows:

AG (not_screen & food & cup $\rightarrow$ ***AF*** mouse_sat)

where ***A*** is a path quantifier defined in CTL, meaning "for all computational paths", ***G*** and ***F*** are temporal quantifiers that correspond to "globally" and "eventually" respectively.

The automatic verification in the SMV model checking tool showed that the property ψ expressing well-being under environmental conditions E is entailed by the model of the agent delayed-response behavior expressed by φ_1.

Now, consider the case of φ_2, expressing the motivation-based behavior of the animal from an external perspective. Property φ_2, expresses:

if the screen is removed and the mouse is not satisfied (has hunger) and it observed a position of food before, then it

will start searching for food first at the position of last observation and then, if it can not find food there, it will continue searching at the other position.

Such specification of behavior can be attributed, for example, to an animal that feels hunger. For the complete formal specification of the motivation-based behavior from an external perspective, we refer to [14]; however, see also next paragraph for part of this specification. Using SMV, the model of agent behavior φ_2 for this case also turns out to entail the general property ψ, expressing well-being under environmental conditions E.

In the second experimental setting, described by environmental conditions E', the mouse observed food for some time at the position p1, after that one cup is put covering the food and another cup is put at the position p2, which is also behind the transparent screen. Thereafter, invisibly for the mouse, food is removed from position p1 and put under the cup at position p2. Later the screen is raised and the animal is free to go to any position. The global property ψ' to be verified in this case expresses well-being under these environmental conditions E':

for all traces if the screen is removed and food is hidden behind the cup at position p2, then the mouse will eventually be satisfied,

or, in CTL:

AG (not_screen & food_at(p2) & cup_at(p2) → ***AF*** mouse_sat)

The automated verification in SMV showed that the model of the agent behavior φ_1 for the delayed-response case does not entail property ψ' expressing well-being under environmental conditions E'. From the counter-example generated by the model checker it is visible that the animal went to the position p1, and did not find food there, and after that did not go anywhere else, which caused the failure of the property.

Unlike the external behavior specification φ_1 that describes the delayed-response behavior of the agent, the specification φ_2 for the motivation-based behavior includes behavioral repertoire to deal with invisible food, expressed in the form of properties that turn out to ensure the entailment of global property ψ'. More specifically, φ_2 expresses the behavior that if the agent could not find food at the position where it has seen it before, and the agent is still not satisfied, then the agent will search for food at another position p2. Formally this is expressed by:

$\forall$t5 state(γ, t5) |= mouse_at(p1) & not(food_at(p1)) & not(mouse_sat) $\Rightarrow$ $\exists$t6, t6>t5 state(γ, t6, output(mouse)) |= performing_action(goto(p2))

$\forall$t7 state(γ, t7) |= mouse_at(p2) & not(food_at(p2)) & not(mouse_sat) $\Rightarrow$ $\exists$t8, t8>t7 state(γ, t8, output(mouse)) |= performing_action(goto(p1))

The automated verification in SMV confirmed that the external behavioral specification φ_2 for the case of motivation-based behavior entails property ψ'.

From the results of verification of the external behavioral specifications φ_1 and φ_2 for both types of behavior in both experimental settings with respect to the entailment of properties ψ and ψ' (see Table 2) we draw the conclusion that the mouse that manifests motivation-based behavior φ_2 fits more for surviving in the world, described by the two types of experimental conditions than the mouse that has the delayed-response behavior φ_1.

Table 2: Outcomes of the Example Analysis

		well-being under different environmental conditions	
		ψ	ψ'
behavior type	delayed response φ_1	+	-
	motivation-based φ_2	+	+

5. Discussion

The possibility to perform automated analysis of a system is important in any domain. From the external perspective, behavior of an agent can be described by dynamic properties specifying correlations between input and output states over time of the agent. A specification of externally observable behavior can be related to empirical information, but may have a high complexity which makes it not suitable for automated analysis. In order to enable automated analysis, such a specification of agent external dynamics has to be transformed into more simple, executable format. This can be achieved in different ways. For example, for translating agent behavioral specifications expressed in modal temporal logics into executable format, procedures described in [3] can be used.

This paper exploits a procedure to generate an executable internal behavioral specification from a more expressive external specification than is possible in modal temporal logics. The complexity of the representation of the executable model obtained is linear in size of the external behavioral specification. More specifically, the external specification is related to the SMV specification in the following linear way:

(1) for every quantified variable from a non-executable specification a variable and an appropriate rule for its update are introduced;
(2) for every nested quantifier an additional variable and an auxiliary executable rule are introduced, which establishes a relation between the quantified variables;
(3) for every observed atom from a past and a conditional formulae from dynamic properties, a corresponding memory state creation and a memory state persistence rule are introduced using the variables described in (1) and (2), and variables that correspond to external events;
(4) for every non-executable dynamic property auxiliary variables fmem and fprep (i.e., the variables that indicate truth values of $\varphi_{mem}(\gamma, t)$ and $\varphi_{prep}(\gamma, t_1)$ respectively) and corresponding update rules are introduced;

(5) for every action specified in $\varphi_{act}(\gamma, t_1)$ a variable and an appropriate update rule are introduced;
(6) for reactivation of agent preparation states the auxiliary variables and the update rules corresponding to observed atoms from $\varphi_{prep}(\gamma, t_1)$ are introduced.

Based on the possibility to obtain an executable specification, an automated analysis method for the consequences of agent behavior is proposed. For this purpose the model checking techniques and the SMV model checking tool are used.

Notice that an SMV-specification comprises constants, variables and state transition rules with limited expressiveness (e.g., no quantifiers). Furthermore, for expressing one complex temporal relation a large quantity (including auxiliary) of transition rules is needed. Specification of agent system behavior in the more expressive predicate-logic-based language TTL is much easier. TTL proposes an intuitive way of creating a specification of system dynamics, which still can be automatically translated into a state transition system description, as shown here.

Alternative methods for temporal analysis of reactive systems are discussed in [8]; also these methods need an executable behavioral specification.

The internal behavioral specification contains properties, which describe creation of mental agent states (i.e., memory and preparation states) and relations between them. We claim that the introduced mental framework is expressive enough for representing most cases of externally observable behavior of an agent. Moreover, 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. For example, the neurological model of a sea hare *Aplysia Californica*, suggested by Roberts and Glanzman [13] allows finding some correspondences between the postulated internal states postulated in 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*. Nevertheless, other existing frameworks that include different types of mental states of an agent (e.g., BDI [11], KARO [16]) can be considered for internal representation. In future work it will be investigated, which alternative or additional mental states could be included into the internal framework in order to transparently and realistically represent complex behavior of an agent.

This paper tackles the problem of transformation from an external behavioral specification into an internal one. However, the other direction (i.e., from internal into external) is also of interest. How to relate externally observable dynamics of an agent to a corresponding specification of its internal dynamics is one of the problems discussed in Philosophy of Mind; e.g., [6]. Both ways of transformation between specifications of external and internal dynamics of the agent are considered of interest. The case of an automated external to internal transformation is addressed in this paper: a translation into a set of direct (causal) temporal relations, describing (hypothetical) internal dynamics of the agent. The converse translation is left for future work.

References

[1] C. Allen, and M. Bekoff, *Species of Mind: the philosophy and biology of cognitive ethology*, MIT Press, 1997.

[2] E.M. Clarke, O. Grumberg, and D.A. Peled, *Model Checking*, MIT Press, Cambridge Massachusetts, London England, 1999.

[3] M. Fisher, "An Introduction to Executable Temporal Logics", *Knowledge Engineering Review 11(1)*, 1996, pp. 3-36.

[4] J. Hawkins, *On Intelligence*, Henry Gholt and Co Ltd, 2004.

[5] W.S. Hunter, "The delayed reaction in animals", *Behavioral Monographs*, vol. 2, 1912, pp. 1-85.

[6] J. Kim, *Philosophy of Mind*, Westview Press, 1996.

[7] R. Kowalski, and M. Sergot, "A logic-based calculus of events", *New Generation Computing, 4*, 1986, pp. 67-95.

[8] Z. Manna, and A. Pnueli, *Temporal verification of reactive systems*, Springer-Verlag, Berlin Heidelberg New York, 1995.

[9] M. Manzano, *Extensions of First Order Logic*, Cambridge University Press, 1996.

[10] K. McMillan, *Symbolic Model Checking*, Kluwer Academic Publishers, 1993.

[11] A. S. Rao, and M. P. Georgeff, "Modeling agents within a BDI architecture", In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR '91)*. Morgan Kaufmann, Cambridge, MA, 1991, pp. 473-484.

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

[13] A.C. Roberts, and D.L. Glanzman, "Learning in Aplysia: looking at synaptic plasticity from both sides", *Trends in Neurosciences*, 26, 2003, pp. 662-670.

[14] A. Sharpanskykh, and J. Treur, "Formal Analysis of Cognitive Agent Behavior: formal theoretical basis", Technical Report VU-TR-260105, Vrije Universiteit Amsterdam, 2005. (http://hdl.handle.net/1871/9832)

[15] O.L. Tinklepaugh, "Multiple delayed reaction with chimpanzees and monkeys", *Journal of Comparative Psychology*, 13, 1932, pp. 207-243.

[16] B. van Linder, W. van der Hoek, and J.-J. Ch. Meyer, "Formalising Abilities and Opportunities of Agents", *Fundamenta Informaticae, 34(1-2)*, 1998, pp. 53-101.