Auton Agent Multi-Agent Syst (2009) 19:4–29
DOI 10.1007/s10458-008-9065-6

Toward a programming theory for rational agents

K. V. Hindriks · J.-J. Ch. Meyer

Published online: 10 October 2008

Abstract An important problem in agent verification is a lack of proper understanding of the relation between agent programs on the one hand and agent logics on the other. Understanding this relation would help to establish that an agent programming language is both conceptually well-founded and well-behaved, as well as yield a way to reason about agent programs by means of agent logics. As a step toward bridging this gap, we study several issues that need to be resolved in order to establish a precise mathematical relation between a *modal* agent logic and an agent programming language specified by means of an *operational semantics*. In this paper, we present an *agent programming theory* that provides both an agent programming language as well as a corresponding agent verification logic to verify agent programs. The theory is developed in stages to show, first, how a modal semantics can be *grounded* in a state-based semantics, and, second, how *denotational semantics* can be used to define the mathematical relation connecting the logic and agent programming language. Additionally, it is shown how to integrate declarative goals and add precompiled plans to the programming theory. In particular, we discuss the use of the concept of higher-order goals in our theory. Other issues such as a complete axiomatization and the complexity of decision procedures for the verification logic are not the focus of this paper and remain for future investigation.

Keywords Rational agents · Programming theory · Verification logic · Denotational semantics · State-based semantics

Part of this research was carried out while the first author was affiliated with the Nijmegen Institute for Cognition and Information, Radboud University Nijmegen.

K. V. Hindriks (✉)
Delft University of Technology, Delft, The Netherlands
e-mail: k.v.hindriks@tudelft.nl

J.-J. Ch. Meyer
Computing and Information Science Department, Utrecht University, Utrecht, The Netherlands
e-mail: jj@cs.uu.nl

1 Introduction

As has been argued in [42], the proper tools for verifying agents are still lacking due to the fact that there is no clear relationship between agent logics on the one hand and concrete computational models used to implement agents on the other hand. One particularly appealing approach to implementing agents is to use an agent programming language and the question becomes how to relate agent programs and agent logics. Though progress has been made [4, 5, 8, 16, 19, 24], the relation between rational agent programs and agent logics including declarative goals is still insufficiently understood. Establishing a proper relation between the semantics of such agent programs and agent logics will provide several benefits, due to the fact that such a connection:

- Indicates that the agent programming language is well-behaved. In particular, this is where *denotational semantics*, a type of mathematical semantics that abstracts from procedural aspects and concentrates on the *effects* of a program, is useful. The hallmark of denotiational semantics is that semantic definitions are defined *compositionally* [15], which provides the basis for a verification logic to verify agent programs and compositional verification tools.
- Indicates that the agent programming language is conceptually well-founded, i.e. based on a set of clearly defined concepts which facilitate understanding rather than complicate the understanding of agent programs. In particular, by providing an elegant denotiational semantics one ensures that a programming framework does not incorporate programming constructs that are hard to understand due to e.g. detailed procedural or operational aspects—from which such a semantics abstracts—and which might be consequently hard to apply in the design and development of agent programs.

In line with these motivations, we see the work reported in this paper as contributing to the design of a well-behaved agent programming language, as well as a contribution to the study of the conceptual foundations of agent programming. To this end, we discuss and present an *agent programming theory*, that consists of two components: an agent programming language and a corresponding verification logic that can be used to axiomatically verify programs written in the programming language (cf. [15]).

The aim of this paper is to define an agent programming theory for *rational agents*. Intuitively, rational agents derive their choice of *action* from their *knowledge* and *goals* and these core concepts need to be incorporated into an agent programming theory.[1] These concepts are key concepts in any account of agency, as is also reflected in some of the best known agent logics [12, 37, 41]. Typically, these logics also include additional concepts such as *commitment* and *intention*, derived from the work of e.g. Bratman [10]. In agent programming languages, additionally, constructs for representing the *plans* of agents are incorporated. Summarizing, our programming theory should include an *informational* component, a *motivational* component, and a *planning or means-ends* component.

For our purposes, it is essential to incorporate the core concepts defining a rational agent into the programming theory but there is also a pragmatic motivation to start with a minimal theory. We take a somewhat pragmatic stance on what is to be included and start with a theory based on the concepts of *action*, *knowledge*, *goals*, and *plans* covering each of the components of a rational agent programming theory and the fundamental notion of action.

For the purpose of developing an agent programming theory, we illustrate how these concepts can be operationalized in an agent programming language and how the operational

[1] In this paper no distinction between knowledge and belief is made. At those places where the distinction is important this will be noted below.

semantics can be formally related to the semantics of an associated agent logic. In our approach to agent programming, knowledge and goals are explicit computational resources (i.e. databases with associated operations) and the operational semantics defines how agents compute with these resources. The corresponding agent logic provides for a declarative formalism to specify and reason about agents (cf. [15]).

The programming theory discussed in this paper is presented in stages, starting with the most basic concepts of action and knowledge and the consecutive incorporation of the other core concepts. Our primary objective is to demonstrate how a programming theory can be constructed for rational agents and to show how traditional techniques from programming language semantics can be useful here. The approach in stages corresponds with our aim, which is only achieved to a limited extent here, to construct various parts of the programming theory in a more or less modular way, in line with the idea that each of these parts (e.g. a theory of action) could be supplanted with another one (e.g. the reader's preferred theory of action). As a first step, a simple programming language based on action and knowledge is introduced and a verification logic for this language based on a modal dynamic logic extended with an epistemic operator is presented. The main issue dealt with here is demonstrating that the state-based semantics for the programming language is equivalent with the modal semantics using techniques from denotational semantics [15]. More precisely, we will be concerned with showing that a state-based semantics that provides the ingredients for defining an operational semantics is equivalent to a denotational semantics that provides the semantics for a modal logic of agent programs. Second, the programming language is extended with a motivational component, i.e. a database of goals. Rational agents perform actions to pursue their goals. Goals are *achievement goals* in this paper, but also include higher-order goals which usefully can be interpreted as *dispositions* to adopt goals (see below). Accordingly, the verification logic is extended with a motivational operator. Some constraints are discussed that need to be imposed on the modal semantics to be able to demonstrate that the thus extended logic is a verification logic for the programming language with goals. Third, the programming language is extended with a precompiled plan library. The semantics for this programming language as well as its verification logic is defined using techniques from denotational semantics, which demonstrates the use of such techniques for agent programming. Fourth, and finally, so-called goal adoption rules are introduced as a means to program the adoption of new goals. The goal adoption mechanism presented is based on the concept of second-order goals or dispositions to adopt a goal, and defines an interesting alternative to some other mechanisms present in the literature for goal adoption.

2 Action and knowledge

The basic ingredients that any theory of rational agency presumably should include are the concepts of action and knowledge. This topic has been a central area of research in Artificial Intelligence from its early beginnings [29] and has also been extensively addressed within computing science more generally [17, 30]. Here, we do not contribute to this research, but instead introduce a simple action and knowledge component of the agent programming language discussed in this paper. The action and knowledge component presented here is common to a number of existing programming languages in the literature [7]. In this section, we also introduce a verification logic for this language based on propositional dynamic logic (PDL) extended with an epistemic operator [17, 22, 30].

We assume that agents use a classical propositional language $\mathcal{L}_0$ to represent their environment, including minimally logical operators for conjunction $\wedge$ and negation $\neg$. The language

$\mathcal{L}_0$ is based on an *infinite* set At of atomic propositions and has an associated entailment relation $\models$. (The requirement that At is infinite is needed in the Proof of Theorem 1 below.) Propositions from $\mathcal{L}_0$ are called *objective propositions*. The extension of $\mathcal{L}_0$ with an epistemic operator K is denoted by $\mathcal{L}_k$ and $\varphi \in \mathcal{L}_k$ are called *knowledge propositions*. Agents maintain a database of sentences from $\mathcal{L}_0$ representing their current *knowledge*. Since in this paper we only deal with single agents, a *state* of the system comprising both the agent and its environment can be represented as a pair: $\langle v, k\rangle$ where $v \subseteq At$, called *world state*, represents the environment, and $k \subseteq \mathcal{L}_0$ an arbitrary set of sentences from $\mathcal{L}_0$, called *knowledge base*, represents the knowledge of the agent. States are typically denoted by $s, s_1, s_2, \ldots$ In this paper, we assume that knowledge is veridical, i.e. $v \models k$ (a constraint that can be dropped without much consequence if k is to model the weaker notion of belief). Note that this requirement also implies that knowledge is consistent.

Agents can perform actions to change the state of the system. Actions thus may change both the environment as well as the agent's knowledge, or either of those, as long as the constraint of veridicality of knowledge is maintained. Actions may be either *basic* or *composed*. We assume a finite set Act of basic actions typically denoted by a; the symbol π is used to denote arbitrary, possibly composed actions. Composed actions are either sequentially composed actions $\pi_1; \pi_2$ or $\texttt{if}\ \varphi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2$ where $\varphi \in \mathcal{L}_k$ is a knowledge proposition. There are many formalisms for specifying actions, e.g. [2, 27], but here we abstract from such issues and represent the preconditions and effects of actions abstractly by means of a *transition function* $\mathcal{T}$. A transition function $\mathcal{T}$ maps a basic action a and a state s to a new state s'. In case $\mathcal{T}(a, s)$ is undefined, action a cannot be performed indicating that some of the action preconditions of a do not hold in state s. It is assumed that preconditions only depend on the environment and not on the knowledge of an agent, meaning that if $\mathcal{T}(a, \langle v, k\rangle)$ is defined $\mathcal{T}(a, \langle v, k'\rangle)$ is also defined for all other knowledge bases k'. Whether an action can be executed thus depends on the environment only (which needs to be in the right configuration, provide a communication medium, etc.) whereas it is the responsibility of the programmer to associate and program conditions on the mental state of an agent as conditions for actual action execution. It is not within the scope of this paper to discuss detailed issues associated with e.g. communicative or sensing actions.

All basic ingredients for defining the core of the agent programming language now have been introduced and the *transition semantics* of the programming language can be defined by means of a transition system [34]. A transition system is a set of transition or derivation rules that inductively define the possible transitions of actions, either basic or composed. A transition of a configuration $\langle \pi, s\rangle$ to a configuration $\langle \pi', s'\rangle$, denoted by $\langle \pi, s\rangle \longrightarrow \langle \pi', s'\rangle$, means that program π can perform one computation step with as effect the transformation of the initial configuration into the latter.

Definition 1 (*Transition semantics*) Let $s = \langle v, k\rangle$ and s' be states, $\mathcal{T}$ a transition function, φ an objective proposition, and let E denote termination of a program. The transition semantics for the basic agent programming language is inductively defined by:

$$\frac{\mathcal{T}(a, s) = s'}{\langle a, s\rangle \longrightarrow \langle E, s'\rangle} \qquad \frac{\langle \pi_1, s\rangle \longrightarrow \langle E, s'\rangle}{\langle \pi_1; \pi_2, s\rangle \longrightarrow \langle \pi_2, s'\rangle} \qquad \frac{\langle \pi_1, s\rangle \longrightarrow \langle \pi_1', s'\rangle}{\langle \pi_1; \pi_2, s\rangle \longrightarrow \langle \pi_1'; \pi_2, s'\rangle}$$

$$\frac{s \models_s \varphi \text{ and } \langle \pi_1, s\rangle \longrightarrow \langle \pi_1', s'\rangle}{\langle \texttt{if}\ \varphi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2, s\rangle \longrightarrow \langle \pi_1', s'\rangle} \qquad \frac{s \not\models_s \varphi \text{ and } \langle \pi_2, s\rangle \longrightarrow \langle \pi_2', s'\rangle}{\langle \texttt{if}\ \varphi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2, s\rangle \longrightarrow \langle \pi_2', s'\rangle}$$

The definition of the semantics for the programming language so far is fairly standard, except maybe for the use of the entailment relation $\models_s$ to evaluate conditions φ on a state

$s = \langle v, k \rangle$. A definition of $\models_s$ is given in Definition 4 and the notation $s \models_s \varphi$ is justified in Proposition 1 in Sect. 3.2. Note that a test φ in an `if` φ `then ... else ...` statement may be an objective formula to evaluate a state in the environment as well as a knowledge proposition to test the state of an agent. Whether direct tests on the environment can be implemented depends on the particular application but our framework allows the modelling of both types of tests.

Our main objective in this paper is to investigate and define a programming theory for rational agents. As a first step toward such a theory, we briefly look at what kind of verification logic is suitable for the simple programming language introduced so far. It is natural to consider standard PDL [22] extended with an epistemic operator K [17, 30], a logic we label DKL here.[2] PDL can be used to reason about the action component of the programming language whereas epistemic logic can be used to reason about the knowledge of an agent. The logic DKL is formally introduced next to allow for a formal proof that the transition semantics and the logical semantics are adequately related.

Definition 2 (*Syntax of dynamic logic extended knowledge*) The same set of propositional atoms At and set of basic actions Act used to define the programming language are assumed. The syntax of propositional dynamic and epistemic formulas $\varphi \in \mathcal{L}_{DKL}$ and programs $\pi \in \Pi_{DKL}$ is given by:

$$\varphi ::= \text{ any element of } At \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\pi]\varphi \mid \mathsf{K}\varphi$$
$$\pi ::= \text{ any element of } Act \mid \pi;\pi \mid \texttt{if}\ \varphi\ \texttt{then}\ \pi\ \texttt{else}\ \pi$$

Propositions $\varphi \in \mathcal{L}_{DKL}$ without occurrences of the dynamic operator $[\pi]$ are called *knowledge propositions* and the set of such propositions is denoted by $\mathcal{L}_k$. Using the fact that Act is finite, we can also define the next operator $\bigcirc\varphi$ as an abbreviation for $[a_1]\varphi \wedge \ldots [a_n]\varphi$ where $a_1, \ldots, a_n$ exhaust the members of Act. This definition illustrates the assumption built into dynamic logic that these actions are the only way to change the agent's environment.

The semantics for the language $\mathcal{L}_{DKL}$ is defined as usual. A *DKL model* is a quadruple $\langle W, R, K, V \rangle$ with W a set of *worlds*, denoted by w, w_1, w_2 and to be distinguished from states, $R : Act \to W \times W$ a mapping of basic actions onto accessibility relations on W, $K \subseteq W \times W$ an *equivalence relation* on W, and $V : At \times W \to \{1, 0\}$ a *valuation function* mapping atoms to truth values. The equivalence relation K defines an S5 operator and is used to model the knowledge of an agent. For our purposes, the epistemic operator could be defined as an S5 or KD45 operator [11], but in line with the veridicality constraint introduced above we will use an S5 model as is usual for knowledge.

Definition 3 (*Semantics of dynamic logic with knowledge*) Let $M = \langle W, R, K, V \rangle$ be a model. The semantics for $\mathcal{L}_{DKL}$ is defined by simultaneous induction:

– The truth conditions for $\varphi \in \mathcal{L}_{DKL}$ are defined by:
  - $M, w \models p$ iff $V(p, w) = 1$,
  - $M, w \models \neg\varphi$ iff $M, w \not\models \varphi$,
  - $M, w \models \varphi \wedge \psi$ iff $M, w \models \varphi$ and $M, w \models \psi$,
  - $M, w \models [\pi]\varphi$ iff $\forall w'(w R_\pi w' \Rightarrow M, w' \models \varphi)$,
  - $M, w \models \mathsf{K}\varphi$ iff $\forall w'(w K w' \Rightarrow M, w' \models \varphi)$.

[2] We do not use the label DEL here which is standardly used nowadays for the epistemic logic with announcement operators, see e.g. [43].

– The semantics of $\pi \in \Pi_{DKL}$ is defined by:[3]
  - $R_a = R(a)$,
  - $R_{\texttt{if}\ \varphi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2} = (R_{\pi_1} \cap (V(\varphi) \times W)) \cup (R_{\pi_2} \cap (V(\neg\varphi) \times W))$,
  - $R_{\pi_1;\pi_2} = R_{\pi_1} \circ R_{\pi_2}$.

Note that the modal program semantics in Definition 3 implies that performing an action or executing a program or plan in an environment may result in changes to that environment as well as to the knowledge of the agent. That is, both objective propositions as well as knowledge propositions may have changed truth value as a consequence of performing an action. Also note that *nested* occurrences of the epistemic operator K in a knowledge proposition φ can be eliminated to obtain a logically equivalent knowledge proposition ψ that does not contain knowledge operators in the scope of another (cf. [11]), a fact useful in the proof of Theorem 1 below.

2.1 DKL is a verification logic

In order to show that the DKL logic can be used to prove properties of programs, we need to show that the operational semantics corresponds "in the right way" with the modal semantics for the logic. The standard way to proceed here is to derive a *denotational semantics* from the operational semantics [15] which defines the input–output relation as a *compositional* function $[\![\pi]\!](s)$ mapping a program and a given input state s to (a set of) output state(s).[4] It is not difficult to define such a function given the operational semantics introduced so far, but the result does not quite match yet with the modal semantics because of the difference between states in the former and worlds in the latter.

It is clear that with any world w in the modal semantics a unique state $s = \langle v, k\rangle$ can be associated: Define $v = \{p \in At \mid M, w \models p\}$ and $k = \{\varphi \in \mathcal{L}_0 \mid M, w \models \mathsf{K}\varphi\}$. It is not immediately clear, however, that by such a reduction the expressivity of the logic is not also reduced. The point is that there may be multiple *copies* of a state present in a model, i.e. worlds w, w' that define the same state s, which may differ with respect to the actions that can be performed in those worlds or even may lead to different states as a result of performing the same action (see for an illustration of the latter Fig. 1). In modal logic, worlds are *intensional* entities, which differ from the *extensional* concept of state.

In order to show that no expressivity is lost, a *state-based* semantics for the DKL logic is defined and it is shown that this semantics is equivalent with the modal semantics. First, the state-based notion of a model M is defined as a pair $\langle W^s, R^s\rangle$ with W^s a set of states of the form $\langle v, k\rangle$ and $R^s : Act \to W^s \times W^s$ a function mapping actions onto a binary relation on states. (Note that a transition function can be viewed as a "curried" version of this function.) For technical reasons, it is also required that state-based models are closed under the following condition: if $\langle v, k\rangle \in W^s$, then $\langle v', k\rangle \in W^s$ for all v' such that $v' \models k$. These states need to be present to ensure all world states that model the agent's knowledge are present in the state-based model. A state-based entailment relation $\models_s$ for $\mathcal{L}_{DKL}$, with subscript s to distinguish it from the standard entailment relation, is defined in terms of these

[3] V is extended such as to apply to arbitrary formulas, and defined by: $V(\varphi) \stackrel{df}{=} \{w \in W \mid M, w \models \varphi\}$, and $R_1 \circ R_2 \stackrel{df}{=} \{(w, w') \mid \exists w''(wR_1w'' \text{ and } w''R_2w')\}$.

[4] The compositionality requirement is important since it allows for a recursive definition of the function that corresponds with the inductive clauses in the truth definition of the logic. As a result, the semantics of the logic can be defined using the denotation of the program and the abstract relation R used in the logical semantics can be replaced with a concrete input–output relation. See also Sect. 4.

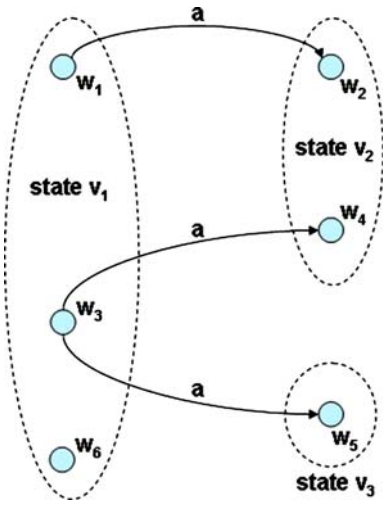

Fig. 1 Intensional worlds versus extensional states

models. We use $s[c'/c]$ to denote that component c in a state s is replaced with c', e.g. $s[v'/v]$ is $\langle v', k\rangle$ if $s = \langle v, k\rangle$.

Definition 4 (*State-based semantics for dynamic logic with knowledge*) Let $M^s = \langle W^s, R^s\rangle$ be a state-based model and $s = \langle v, k\rangle \in W^s$ be a state. The state-based semantics for $\mathcal{L}_{DKL}$ is defined by simultaneous induction as follows:

– The truth conditions for $\varphi \in \mathcal{L}_{DKL}$ are defined by:

- $M^s, s \models_s p$ iff $p \in v$,
- $M^s, s \models_s \neg\varphi$ iff $M^s, s \not\models_s \varphi$,
- $M^s, s \models_s \varphi \wedge \psi$ iff $M^s, s \models_s \varphi$ and $M^s, s \models_s \psi$,
- $M^s, s \models_s [\pi]\varphi$ iff $\forall s'(s R^s_\pi s' \Rightarrow M^s, s' \models_s \varphi)$,
- $M^s, s \models_s \mathsf{K}\varphi$ iff $\forall v'(v' \models k \Rightarrow M^s, s[v'/v] \models_s \varphi)$.

– The semantics of $\pi \in \Pi_{DKL}$ is defined by:

- $R^s_a = R^s(a)$,
- $R^s_{\texttt{if}\ \varphi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2} = (R^s_{\pi_1} \cap (V(\varphi) \times W^s)) \cup (R^s_{\pi_2} \cap (V((\neg\varphi) \times W^s)))$,
- $R^s_{\pi_1;\pi_2} = R^s_{\pi_1} \circ R^s_{\pi_2}$.

Observe that the right-hand side of the clause for $\mathsf{K}\varphi$ can be replaced with $k \models \varphi$ in case φ is an objective formula, where $\models$ denotes classical propositional entailment, the condition used in the transition semantics in Definition 1. Definition 4 is interesting since it suggests that for knowledge propositions $\varphi \in \mathcal{L}_k$ we have $s \models_s \varphi$ iff $M^s, s \models_s \varphi$, a fact that shows that the truth of such propositions only depends on the state (see Sect. 3.2 for a proof). In any case, we have to restrict the condition part in **if-then-else**-statements and disallow occurrences of dynamic operators $[\pi]$ since these cannot be evaluated in a *state-based* semantics but would require look-ahead facilities. Conditions which allow inspection of the results of program execution are called *rich tests* and tests which do not allow occurrences of $[\pi]$ are called *poor tests* (cf. [22]).

The next theorem, proving the equivalence of the standard and state-based semantics, shows that worlds can be replaced with states as usual in dynamic logic for imperative

programming, where variable assignments replace the abstract worlds of modal logic. The proof essentially uses the fact that At is infinite.

Theorem 1 (Equivalence of the modal and state-based semantics) *Let $\varphi \in \mathcal{L}_{DKL}$ be any dynamic formula with epistemic operators, possibly including rich tests. Then we have that the modal and the state-based semantics are equivalent:*

$$\text{for any } \varphi \in \mathcal{L}_{DKL} : \models \varphi \text{ iff } \models_s \varphi$$

Proof See Appendix. □

Remark 1 The results presented in this section may remind the reader of *knowledge structures* [18]. Knowledge structures provide a model-theoretic alternative for Kripke semantics. Our objectives, however, are different: We are interested in designing a *programming language for agents* that compute with databases and not in the structure of knowledge states. The "flat" syntactic approach to knowledge bases suits our purposes, whereas it "begs the question of what a model of a state of knowledge is" [18]. Since our primary goal is the design of an agent programming language, we do not feel committed to any particular view of what knowledge bases are. The "databases" may be replaced with other *state-based* structures, such as knowledge structures, which could be considered an improvement on the approach presented here.

The fact that the DKL logic can be used to prove agent programs correct is proved in Sect. 4, after the basic agent programming language has been extended with declarative goals and a "planning" capability using precompiled plans.

3 Declarative goals

We believe that the hallmark of rational agents is that they are motivated by and pursue goals. Rational agents base their choice of action on their knowledge as well as on their goals. By this token, rational agents can be differentiated from other types of software. In this section, we extend the programming theory of the previous section with declarative goals. The approach we use for incorporating declarative goals into the programming theory is, first, in this section to show how to incorporate the concept of (static) goals, and, second, in Sect. 5 to show how to incorporate goal dynamics into the semantics of agent programs.

To a certain extent, the story to be told is similar to that of the previous section. This time a goal base is added to the state instead of a knowledge base, but the combination requires careful consideration and the structure of the goal base itself raises some new issues. In this case, it can also be shown that a state-based semantics including goals is equivalent with the modal semantics, but some additional constraints on the modal semantics are needed.

3.1 Introducing declarative goals into the programming language

Extending the programming language with static (declarative) goals in a sense is quite easy: The idea is to add another database, called a *goal base*, to the system states. This reflects the approach taken in some agent programming languages [14, 23], which additionally introduce a simple commitment strategy.[5] However, it seems that the content of a goal base leaves

[5] A commitment strategy specifies the (default) strategy that an agent uses for dropping its goals; e.g. an agent with a blind commitment strategy only drops a goal when it believes it has been achieved [16, 37].

room for several choices. For example, do we allow knowledge propositions in the goal base? Does the goal base have the same "flat" structure as the knowledge base, that is, do we allow higher-order goals to represent e.g. the desire of an agent to pursue a goal?

It seems natural to answer the first question affirmatively, and, in our programming theory, we therefore include goals to obtain information, i.e. goals in the goal base of the form $\mathsf{K}\varphi$. Even though the computational costs may prohibit the unrestricted use of knowledge propositions in an agent programming language, we feel that the programming theory at least should allow for it in principle. From an engineering perspective, it will be useful to allow goals to obtain knowledge in some restricted form: Rational agents that may have goals to obtain knowledge are more flexible than those that may not.

It will be clear that higher-order goals are different from first-order goals, a fact also true in some of the best known agent logics [12, 37, 41]. In those logics, nested occurrences of motivational operators unlike the epistemic operator cannot be eliminated. We assume that an agent that has a goal φ, drawn from the set of knowledge propositions $\mathcal{L}_k$, in its goal base indicates that the agent wants to change the system such as to satisfy φ. Such goals are known as *achievement goals*. Higher-order goals can be expressed by introducing a goal operator G. An agent that has a goal $\mathsf{G}\varphi$ in its goal base wants to adopt φ as a goal. Such goals are called *second-order* goals. Interestingly, Sloman et al. in [40] have also pointed to the role of higher-order goals as to what they call *motive generators*. The importance of the distinction, moreover, is also illustrated by the fact that it has been used by philosophers to provide an account of autonomy and even to argue that it differentiates humans from animals and artificial machines (cf. [20]). Closer to agent programming, [28] also state that it is the *self-generation* of goals that makes an agent autonomous. Higher-order goals, and in particular second-order goals, naturally give rise to a mechanism for adopting goals. We will show how higher-order goals can be incorporated into the programming theory and will define one such mechanism for goal adoption in Sect. 5.

Technically, the ideas discussed can be realized by extending states $s = \langle v, k\rangle$ introduced in the previous section with a goal base g and from now on we will assume states are of the form $s = \langle v, k, g\rangle$. Here, the goal base g is defined as a mapping from the natural numbers $\mathbb{N}$ to sets of knowledge propositions, i.e. $g(i) \subseteq \mathcal{L}_k$, at various levels in a *linear* hierarchy of goals. The idea is that goals $g(i)$ at a particular level $i > 0$ in this hierarchy represent the desire of an agent to pursue a goal whereas the set $g(0)$ represent goals the agent wants to achieve in the environment. Note that we allow that an agent has epistemic goals, e.g. to obtain knowledge about its environment. There remain two issues to be dealt with. The first is whether goal formulas $\mathsf{G}\varphi$ are also allowed in the knowledge base k. The second is how to define a *computational or state-based interpretation* of conditions in agent programs that include goal expressions.

As for the first issue, goal formulas will not be allowed in the knowledge base, a choice motivated by various considerations. For one thing, it would significantly complicate the structure of the knowledge base, since it would imply incorporating a hierarchy of goals as in the goal base itself. Such a construction reminds one of the *knowledge structures* of [18]. More importantly, however, a simple and elegant way to resolve the issue is available by introducing the natural assumption that agents have full introspective capabilities with respect to their goals, just as is assumed for knowledge itself (see e.g. also [39]). This

Footnote 5 continued
One alternative could be to use priorities on goals to "guide" the execution of programs as in [38]. The questions raised next in the main text also are not exhaustive, and some other questions which naturally come to mind are not addressed, such as whether to introduce priorities on goals as in the cited paper. As far as we know, however, these questions have not been addressed yet in the agent programming literature.

assumption, moreover, also allows for an elegant definition of the computational interpretation of conditions including goal operators (see Definition 5 below). Given this assumption no changes are required to the action and knowledge part of the programming theory presented in the previous section, which can be viewed as a step toward a programming theory in which the different parts can be treated as more or less "independent modules". Finally, it also seems more natural to think of agent capabilities as only changing the environment and, indirectly, the knowledge of an agent representing these changes, but not the agent's goals. Another mechanism will be introduced to enable changes to the agent's goals in Sect. 5.

3.2 Computational interpretation of intentional propositions

The extension of the propositional language of knowledge propositions $\mathcal{L}_k$ with a goal operator G is denoted by $\mathcal{L}_g$ and formulas $\varphi \in \mathcal{L}_g$ are called *intentional propositions*. As before, in the programming language dynamic operators $[\pi]$ are not allowed in conditions in **if-then-else**-statements, but intentional propositions are allowed. In an operational semantics, such conditions are evaluated locally in the current states of the system to avoid the need for look-ahead facilities and we need a state-based or computational interpretation of intentional propositions.

Given the assumptions discussed above, the computational interpretation of intentional propositions can be defined as a simple extension of the state-based semantics for knowledge propositions. Depending on the number of nested occurrences of goal operators, different levels in the goal hierarchy are needed to evaluate an intentional proposition φ and it is useful to introduce an operator to "step through" this hierarchy. For a goal base g, we write g^{+1} for the goal base g' such that $g'(i) = g^{+1}(i) = g(i+1)$.

Definition 5 (*Computational interpretation of intentional propositions*) Let $s = \langle v, k, g\rangle$ be a state such that $v \models k$. The truth conditions for $\varphi \in \mathcal{L}_g$ an intentional proposition are defined by the clauses in Definition 4 extended with the following clause:

$$M^s, s \models_s \mathsf{G}\varphi \text{ iff } \forall v', k'(M^s, s[v'/v, k'/k] \models_s g(1) \Rightarrow M^s, s[v'/v, k'/k, g^{+1}/g] \models_s \varphi).$$

Note that the state-based entailment relation $\models_s$ can be used on the right hand side in the clause of Definition 5 since $g(1) \subseteq \mathcal{L}_k$ does not itself contain goal operators. The definition shows that first-order goals are interpreted by the set of level 1 goals $g(1)$ and higher-order goals are interpreted by higher levels in the hierarchy. A formula $\mathsf{G}\varphi$ with $\varphi \in \mathcal{L}_k$ is evaluated in all models of $g(1) \subseteq \mathcal{L}_k$, i.e. all world state and knowledge base pairs that are models of $g(1)$.

It is not hard to verify that $\mathsf{K}(\neg)\mathsf{G}\varphi \leftrightarrow (\neg)\mathsf{G}\varphi$ is valid, as a result of the fact that only the world/knowledge base components of a state are varied in the semantic clause for G above. (The same holds true if the epistemic operator K would be replaced by a doxastic operator B, but then the beliefs of an agent about its own goals must be assumed to be veridical since in this setup of the semantics this property is true by definition).

The following proposition shows that intentional propositions can be evaluated on a state and we can simply write $s \models_s \varphi$ for $\varphi \in \mathcal{L}_g$.

Proposition 1 *Let $\varphi \in \mathcal{L}_g$ be an intentional proposition and M_1^s, M_2^s be two state-based models with $s \in W_1^s, W_2^s$. Then we have:*

$$M_1^s, s \models_s \varphi \text{ iff } M_2^s, s \models_s \varphi$$

Proof Straightforward, since the truth of intentional propositions does not depend on the relation R^s in a model M^s. □

As a result of Proposition 1, the transition semantics of Definition 4 can be used for agent programs including intentional propositions without any modifications. In practice, however, the unrestricted use of intentional propositions in agent programs is prohibitive since the satisfaction problem for $\mathcal{L}_g$ is NP-hard. For restricted fragments of the logic, however, in particular the Horn clause fragment with bounded modal depth, it is reasonable to conjecture that the complexity of the satisfaction problem is in PTIME [32] (Nguyen, Private communication). Since the number of nested modalities (in our case nestings of K and G) in any agent program is bounded, this would show that the Horn clause fragment can be used in the engineering of efficient agents.

Finally, to be complete, we need to discuss the transition function $\mathcal{T}$ that now needs to be applied to states including goal bases in the extension of the agent programming language with goals. In line with the discussion above, actions are not supposed to change the agent's goals while, as before, action execution must preserve the veridicality of knowledge. Formally, these requirements are captured by the following constraint: If $\mathcal{T}(a, v, k, g) = \langle v', k', g' \rangle$, then $v' \models k'$ and $g' = g$. Note that this is a constraint on the *operational* semantics of our programming language and not of the logical semantics of the Dynamic Agent Logic discussed next.

3.3 A dynamic agent logic

To conclude this section, an extension of the dynamic logic with knowledge of the previous section with a goal operator is defined. The extended logic $\mathcal{L}_{DAL}$ is called *dynamic agent logic* (DAL) and follows standard practice in the literature by adding a KD operator to the language (cf. [12, 37, 41]). DAL models $\langle W, R, K, G, V \rangle$ extend DKL models with a *serial* relation G to model the goals of an agent. One additional semantic clause is needed to specify the truth conditions for formulas in $\mathcal{L}_{DAL}$ of the form $\mathsf{G}\varphi$.

Definition 6 (*Semantics of dynamic agent logic*) Let $M = \langle W, R, K, G, V \rangle$ be a DAL model and $w \in W$. The truth conditions for propositions $\varphi \in \mathcal{L}_{DAL}$ are defined by the clauses in Definition 3 extended with the following clause:

- $M, w \models \mathsf{G}\varphi$ iff $\forall w'(wGw' \Rightarrow M, w' \models \varphi)$.

Additionally, some constraints need to be introduced on DAL models to ensure that they incorporate the features discussed above as well as that they match with the structural features imposed on goal bases, i.e. the linear hierarchy of goals. The (positive and negative) introspective capabilities with respect to goals and the fact that goals are not changed by performing actions correspond to the following restrictions.

Constraint 1 (Introspection and goal persistence) *Let $\langle W, R, K, G, V \rangle$ be a DAL model and let $G(w)$ denote the set $\{w' \in W \mid wGw'\}$. The following restrictions are imposed:*

- $\forall w, w', w''((wKw' \text{ and } w'Gw'') \Rightarrow wGw'')$,
- $\forall w, w', w''((wGw'' \text{ and } wKw') \Rightarrow w'Gw'')$,
- $\forall w, w'(wR_aw' \Rightarrow G(w) = G(w'))$.

It is easy to verify that the first two constraints imply respectively positive and negative introspection of goals and validate the axiom schemes $\mathsf{G}\varphi \rightarrow \mathsf{KG}\varphi$ and $\neg\mathsf{G}\varphi \rightarrow \mathsf{K}\neg\mathsf{G}\varphi$.

Since for arbitrary φ we also have $\mathsf{K}\varphi \rightarrow \varphi$ it follows that $\mathsf{K}(\neg)\mathsf{G}\varphi \leftrightarrow (\neg)\mathsf{G}\varphi$ is valid, a fact that allowed us to simplify the state-based semantics as discussed. The third constraint implies that goals do not change. Such persistence, of course, is not what we ultimately aim for, and will be relaxed in later sections.

Another constraint is introduced to "tame" the branching structure which is naturally induced by the tree-like structure of the modal relation modeling goals. For example, we may have $\mathsf{G}(\mathsf{G}\varphi \vee \mathsf{G}\psi)$ but not have that either $\mathsf{GG}\varphi$ or $\mathsf{GG}\psi$ is also true. The modal semantics thus does not give rise naturally to the linear hierarchy of goals introduced in the state-based semantics. It seems to us, however, that it is both cognitively more plausible to have a linear hierarchy of goals instead of a branching one as well as that such a structure is more useful from an agent programming perspective since a branching structure would unnecessarily complicate an agent program. The following constraint restricts the branching of the modal relation G modeling goals and can be viewed as a "linearization" of that structure, or as defining a confluence property of that relation, as shown in Lemma 1.

Constraint 2 (Definite stance toward goal adoption) *Let $\langle W, R, K, G, V\rangle$ be a DAL model. The following restriction is imposed:*

– $\forall w_1, w_2, w_3, w_4((w_1 G w_2 \text{ and } w_1 G w_3 \text{ and } w_2 G w_4) \Rightarrow w_3 G w_4)$.

Lemma 1 *The class of models that satisfy Constraint 2 validate the schema $\mathsf{G}^i\mathsf{G}\varphi \vee \mathsf{G}^i\neg\mathsf{G}\varphi$ for $i \geq 0$, where $\mathsf{G}^0\varphi = \varphi$ and $\mathsf{G}^{(i+1)}\varphi = \mathsf{G}(\mathsf{G}^i\varphi)$.*

Proof By induction on i, prove that we have $G(w') = G(w'')$ for arbitrary w', w'' such that wG^iw' and wG^iw''. The result then follows since in any set of worlds either (1) all worlds satisfy φ, (2) or at least one does not. □

Besides the technical advantage that this constraint yields, we think it also expresses a clear attitude of the agent regarding its higher-order goals. The second-order version of the axiom schema $\mathsf{GG}\varphi \vee \mathsf{G}\neg\mathsf{G}\varphi$ illustrates this attitude. It expresses that an agent has a definite stance toward goal adoption and is either inclined to adopt the goal or inclined *not* to adopt it. (Which, it should be noted, is not the same as $\mathsf{GG}\varphi \vee \mathsf{GG}\neg\varphi$! Also note that disjunctive first-order goals $\mathsf{G}(\varphi \vee \psi)$ are allowed.) An agent thus cannot be "indifferent" to having a goal φ, in the sense that it does not care whether to have φ as a goal or not.

Finally, as for the logic DKL, we have for the agent logic DAL that the standard modal semantics and the state-based semantics defined in Definition 5 are equivalent.

Theorem 2 (Equivalence of $\models$ and $\models_s$ for DAL)
The modal and the state-based semantics for DAL are equivalent:

$$\text{for any } \varphi \in \mathcal{L}_{DAL} :\models \varphi \text{ iff } \models_s \varphi$$

Proof The proof is analogous to Theorem 1, but now requires the use of Constraints 1 and 2 as well. □

4 Adding precompiled plans

Rational agents derive their choice of action from their beliefs and goals. The action selection mechanism of rational agents thus is modeled on the way that humans arrive at a choice of action by means of practical reasoning: in order to achieve a goal an action is selected that is

believed to lead to that goal [1]. One of the means that agents typically are equipped with to realize their goals in agent programming is a repository of plans, or a *plan library*. Plans in this approach are not derived from first-principles but are retrieved from a predefined set of plans by means of so-called plan rules as e.g. in [21, 36].

In languages such as 3APL with agents that have declarative goals, such rules typically are of the form $\mathsf{G}\varphi \leftarrow \mathsf{K}\psi \mid \pi$ [7]. Accordingly, in this section, the basic agent programming language of the previous sections is extended with such plan rules and a denotational semantics equivalent to the operational semantics is provided to show that the agent logic extended with plan rules is a verification logic for this language.

4.1 Introducing precompiled plans into the programming language

In order to do so, the programming language is provided with a slightly different mechanism than that of 3APL which allows the adoption of a new plan by the agent purely on the basis of its current knowledge and its goals. Such a mechanism, however, is difficult to control as witnessed by the fact that various so-called deliberation cycles have been proposed to constrain the (non-deterministic) application of such rules [13, 35]. Another option to control the application of rules is to extend the programming language and introduce an explicit construct $\mathsf{G}\varphi!$ which instructs the agent to retrieve a plan to achieve the goal φ (similar to the operator introduced in [36]). Note that the program statement $\mathsf{G}\varphi!$ is differentiated by the marker "!" from the logical formula $\mathsf{G}\varphi$. The idea is that whenever φ is the case nothing has to be done, and, otherwise, the agent has to retrieve a plan to achieve the goal from a *finite* plan library $\mathsf{PlanLib}$. A plan rule has a *head* $\mathsf{G}\varphi$, a *guard* $\mathsf{K}\psi$ and a *body* π. The body π can be any program or plan, possibly including again a program statement of the form $\mathsf{G}\varphi!$. Plan rules thus may have a recursive structure. In line with the discussion about conditions in **if-then-else**-statements above, the syntax of plan rules will be restricted: Occurrences of the dynamic operator $[\pi]$ are not allowed in either the head, guard or body of a plan rule. Moreover, the guard of a rule must be a knowledge proposition.

The transition semantics introduced in Sect. 2 requires two new rules to specify the semantics of the statement $\mathsf{G}\varphi!$ and the application of plan rules. The transition rule for plan rules slightly differs from the typical body replacement rules in the literature. The main difference is that the rule does not replace "simple" procedure names (or propositional symbols, which are often overloaded and also used for naming procedures in the agent programming literature), but replaces more complex achievement goal statements of the form $\mathsf{G}\varphi!$. Two cases are distinguished: (i) the case that the agent does not believe φ has been achieved, and (ii) the case that the agent does believe φ has been achieved. In the former case, in order to execute the program statement $\mathsf{G}\varphi!$ the agent has to retrieve a plan from its plan library and replace $\mathsf{G}\varphi!$ with an appropriate plan. In the latter case, the agent does not have to do anything to achieve φ and the semantics of the program statement $\mathsf{G}\varphi!$ is defined as a so-called *skip* action, i.e. the action that does not change the agent's state and does not change the flow of control. (Note that even though plan rules are part of the agent's program most of the time they are left implicit throughout the rest of this paper.)

Definition 7 (*Transition semantics for goals and plan rules*) Let s, s' be states, $\mathcal{T}$ a transition function, $\mathsf{G}\varphi$ an intentional proposition, and $\mathsf{K}\psi$ a knowledge proposition. Then the operational semantics for the agent programming language is defined by the set of transition rules of Definition 1 and the following two rules:

$$\frac{s \models_s \neg \mathsf{K}\varphi \wedge \mathsf{K}\psi,\ \mathsf{G}\varphi \leftarrow \mathsf{K}\psi \mid \pi \in \mathsf{PlanLib}}{\langle \mathsf{G}\varphi!, s\rangle \longrightarrow \langle \pi, s\rangle} \qquad \frac{s \models_s \mathsf{K}\varphi}{\langle \mathsf{G}\varphi!, s\rangle \longrightarrow \langle E, s\rangle}$$

Observe that the semantics of $\mathsf{G}\varphi!$ can be used in a program to "force" an agent to pursue a particular goal φ *even if* φ *is not part of the agent's goal base*. The transition rules for $\mathsf{G}\varphi!$ do *not* require that the agent actually has a goal φ in order to be able to execute the program statement. The program statement thus can be used by a programmer to ensure that an agent will pursue a particular goal by adopting plans to achieve it even if the agent does not have an explicit goal φ in its goal base.

An operational semantics that provides an input–output relation for extended agent programs with plan rules can now be defined in the standard way by means of taking the transitive closure of the transition relation $\longrightarrow$ (cf. [34]).

Definition 8 (*Operational semantics for agent programs*) The operational semantics for agent programs π is defined by:

$$\mathcal{O}(\pi)(s) = \begin{cases} s', & \text{if } \langle \pi, s\rangle \longrightarrow^* \langle E, s'\rangle \\ \bot, & \text{otherwise} \end{cases}$$

This definition does not differentiate between *non-terminating programs* and *programs that "get stuck"* in some state, i.e. no further steps can be performed. In both cases, the operational semantics $\mathcal{O}$ maps such non-terminating programs on $\bot$ representing that the output is undefined. When a program successfully terminates, the function $\mathcal{O}$ defines the input–output relation as a mapping from initial states s to the final state reached upon termination. Termination properties are not dealt with any further in this paper.

4.2 Agent logic with precompiled plans

In this section we provide the denotational semantics for the agent programming language including plan rules. This semantics can be used to define a state-based semantics for an agent logic including knowledge and goals discussed in the previous sections and plan rules. The results of the previous sections show that a state-based semantics is equivalent to a standard modal semantics and apply to the extended agent logic as well. The contribution of this section is to, first, define the extended agent language and a denotational semantics, and, second, to prove the equivalence of the denotational and operational semantics. The latter result shows that the agent logic is a *verification logic* for the agent programming language.

Definition 9 (*Syntax of dynamic agent logic with plan rules*) The syntax of propositional dynamic agent formulas $\varphi \in \mathcal{L}_{DAL}$, programs $\pi \in \Pi_{DAL}$ and plan rules ρ is given by the following clauses:

$$\begin{aligned} \varphi &::= \text{any element of } At \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\pi]\varphi \mid \mathsf{K}\varphi \mid \mathsf{G}\varphi \\ \pi &::= \text{any element of } Act \mid \texttt{if}\ \psi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2 \mid \mathsf{G}\psi! \mid \pi;\pi,\ \text{with } \varphi \in \mathcal{L}_g \\ \rho &::= \mathsf{G}\varphi \leftarrow \psi \mid \pi,\ \text{with } \psi \in \mathcal{L}_k \end{aligned}$$

In line with previous remarks, the condition φ in **if-then-else**- and $\mathsf{G}\varphi!$-statements must be an intentional proposition, which allows us to use the entailment relation $\models_s$ from Definition 5 in the definition of a denotational semantics.

The main issue in defining a denotational semantics for the agent programming language with plan rules is defining the semantics for the goal statements $\mathsf{G}\varphi!$ which may involve

recursion due to the application of plan rules. The techniques from denotational semantics can be applied to resolve this issue and to define the semantics of such statements as the *least fixed point* of an interpretation function that defines the meaning of programs. Due to space limitations only a sketch of this idea will be provided and the reader is referred to [15] for a more detailed account. The key notions that we need are that of a *complete partial order* (cpo) on states and a *continuous* interpretation function $[\![\cdot]\!]$ for agent programs. To this end, the set of all states S of the form $\langle v, k, g\rangle$ is extended to a set $S_\perp$ which additionally includes the special symbol $\perp$ denoting the undefined state. The set $S_\perp$ with associated order $s \sqsubseteq s'$ whenever $s = s'$ or when $s = \perp$ defines a so-called flat cpo. Using this cpo, the interpretation function $[\![\pi]\!]$ can be defined as a mapping from $S_\perp$ to $S_\perp$, such that $[\![\pi]\!](\perp) = \perp$ (functions mapping $\perp$ in a cpo to $\perp$ are called *strict*). A strict interpretation function is clearly continuous on a flat cpo (cf. [15]), guaranteeing the existence of a least fixed point. Using the notions introduced, the interpretation function $[\![\pi]\!]$ can now be defined for arbitrary programs including plan rules as a least fixed point. In order to be able to apply the least fixed point operator μX we write $\pi[X/\mathsf{G}\varphi!]$ to denote the program π with occurrences of $\mathsf{G}\varphi!$ with X. For simplicity, here we assume that plans are not mutually recursive, i.e. we assume there are no plans invoking another plan that invokes the first plan again (for this more general case see [15]).

Definition 10 (*Denotational semantics for agent programs*) Let S be the set of all states as above, and $\mathcal{T}$ be a transition function on actions and states. We use ρ to denote plan rules of the form $\mathsf{G}\varphi! \leftarrow \mathsf{K}\psi \mid \pi$ and $body(\rho)$ to denote the body π of such a rule. The denotational semantics for agent programs is defined by:

$$
\begin{aligned}
[\![a]\!](s) &= \begin{cases} \mathcal{T}(a,s), & \text{if } \mathcal{T}(a,s) \text{ is defined} \\ \perp, & \text{otherwise} \end{cases} \\
[\![\pi_1; \pi_2]\!](s) &= [\![\pi_2]\!]([\![\pi_1]\!](s)) \\
[\![\texttt{if } \varphi \texttt{ then } \pi_1 \texttt{ else } \pi_2]\!](s) &= \begin{cases} [\![\pi_1]\!](s), & \text{if } s \models_s \varphi \\ [\![\pi_2]\!](s), & \text{otherwise} \end{cases} \\
[\![\mathsf{G}\varphi!]\!](s) &= \begin{cases} s, & \text{if } s \models_s \mathsf{K}\varphi \\ (\mu X.body(\rho)[X/\mathsf{G}\varphi!])(s), & \text{if } s \models_s \neg\mathsf{K}\varphi \wedge \mathsf{K}\psi \\ \perp, & \text{otherwise} \end{cases}
\end{aligned}
$$

It is not difficult to show that $[\![\cdot]\!]$ is well-defined. The next theorem states that the denotational semantics and the operational semantics are equivalent.

Theorem 3 (Equivalence of denotational and operational semantics) *The denotational and operational semantics for DAL programs are equivalent:*

$$[\![\pi]\!](s) = \mathcal{O}(\pi)(s)$$

Proof Use induction on the length of a computation to prove that $\mathcal{O}(\pi)(s) \sqsubseteq [\![\pi]\!](s)$. Use induction on the size of the program π and depth of recursion k to show that $[\![\pi]\!](s) \sqsubseteq \mathcal{O}(\pi)(s)$. □

The equivalence of the denotational and operational semantics shows that the agent logic can be used to verify (partial) correctness properties of agent programs. This fact is expressed mathematically in the following corollary. The left hand side of the corollary statement concerns the step-by-step behavior of an agent program whereas the equivalence with the right hand side shows that the agent logic can be used to (compositionally) reason about such programs, without the need to inspect the operational behavior of an agent program.

Corollary 1 (Proving partial correctness properties of agent programs) *Let π be a poor test DAL program. Then we have:*

$$\forall s, s' : \text{ if } \models_s \varphi \text{ and } \langle \pi, s \rangle \longrightarrow^* \langle E, s' \rangle, \text{ then } \models_{s'} \psi$$
$$\text{iff}$$
$$\models \varphi \rightarrow [\pi]\psi$$

Proof Immediate from Theorems 2 and 3. □

5 Adopting declarative goals

In the previous sections it was shown how declarative goals combined with plan rules can be incorporated into an agent programming theory. The main objective was to ensure that the agent logic can be used to reason about agent programs that have a computational semantics, which was achieved by introducing some reasonable constraints on the modal semantics. One of those constraints, however, required goals to be static. Here, we relax this constraint by introducing a mechanism for goal adoption using the concept of second-order goals.

5.1 Dispositions to adopt goals

The mechanism for goal adoption is based on a dispositional view of second-order goals: An agent with a second-order goal φ is *disposed* to adopt φ as a first-order goal.

In the literature, a goal has been interpreted primarily as a first-order goal denoting a state that can be achieved through action (cf. [33]). Second-order and higher-order goals have not been paid equal attention, though some related notions have been discussed. As mentioned, Sloman et al. in [40] discuss second-order motives as first-order goal generators. The notion of a *motive* in [33] and that of *concern* in [31] is also related to our dispositional view of second-order goals. A motive or concern is more stable than a first-order goal, just as, in our view, second-order goals are. In [31, 33] motives and concerns respectively provide reasons for goal adoption. In [33] it is claimed that (pro-active) goal adoption based on motives provides for a more efficient goal adoption mechanism, which is based on *attention triggering* in [33].

One intuitive interpretation of second-order goals is that they are *dispositions* to change ones mental state and, given the right circumstances, result in the adoption of a (first-order) goal to achieve φ. Second-order goals thus have a "monitoring" function. As an example, consider a cleaning robot. A cleaning robot does not have (to have) a goal to clean garbage when there is no garbage, but it should be disposed to clean any garbage if there is any.

5.2 Goal adoption in the agent programming language

An agent program is extended with *goal adoption rules* of the form $\varphi \Rightarrow_G \psi$ to implement the adoption of first-order goals ψ. The formula φ is called the *triggering condition* of the rule and ψ the *goal* of the rule. The triggering condition φ and goal ψ must be knowledge propositions, i.e. $\varphi, \psi \in \mathcal{L}_k$. Informally, a goal adoption rule $\varphi \Rightarrow_G \psi$ fires when the agent is disposed to adopt the goal ψ, i.e. $\mathsf{GG}\psi$ holds and the circumstances are believed to be right by the agent, i.e. $\mathsf{K}\varphi$ holds. Additionally, a goal is only adopted if it is consistent with the current set of goals of an agent. The rationale for this is the assumption that an agent that is disposed to adopt ψ, i.e. $\mathsf{GG}\psi$ but also has a goal $\neg\psi$, will not have self-generated this goal,

but e.g. have adopted such a goal upon request of another agent. A goal adoption rule in our programming language is similar to the notion of a goal template in [33], whereas, the notion of a so-called "repository of known goals" in [28] can be related to our operationalization of second-order goals.

Formally, an (extended) agent program is defined as a pair $A = \langle \pi, \Gamma \rangle$ with Γ a set of goal adoption rules. We introduce some additional notation to facilitate the introduction of the transition semantics: Given an agent $A = \langle \pi, \Gamma \rangle$ we use $A[\pi']$ to denote the agent $\langle \pi', \Gamma \rangle$; given a state $s = \langle v, k, g \rangle$ we use $s[d/g(0)]$ to denote the state $\langle v, k, g' \rangle$ with $g'(0) = d$ and $g'(i+1) = g(i+1)$. The transition semantics for agent programs is extended with the following transition rule.

Definition 11 (*Transition rule for goal adoption*) Let $s = \langle v, k, g \rangle, s' = \langle v', k', g \rangle$ be DAL states and define the adoption set e by:

$$e = \bigcup_{\varphi \Rightarrow_G \psi \in \Gamma} \{\psi \mid s \models_s \mathsf{K}\varphi \wedge \mathsf{GG}\psi \wedge \neg\mathsf{G}\neg\psi\}$$

Then the transition semantics of an agent program $A = \langle \pi, \Gamma \rangle$ is defined by:

$$\frac{\langle \pi, s \rangle \longrightarrow \langle \pi', s' \rangle}{\langle A, s \rangle \longrightarrow \langle A[\pi'], s'[g(0) \cup e/g(0)] \rangle}$$

As a simple example to illustrate the semantics, assume that a cleaning robot is disposed to adopt the goal $GarbageInBin$, i.e. $\mathsf{GG}GarbageInBin$, initially has no goals, i.e. $g(0) = \emptyset$, and executes the program:

$$(\mathsf{G}GarbageInBin; clean + \neg\mathsf{G}GarbageInBin; observe; wander)^*; \mathsf{K}(time = 6pm)?$$

with $clean$, $observe$ and $wander$ atomic actions. The robot is supposed to clean until 6 pm every day. Additionally, a single goal adoption rule $garbage \Rightarrow_G GarbageInBin$ is part of the robot's program. Since, initially, the robot does not have any goals it will execute the right branch of the choice program and perform the actions $observe$ and $wander$ (repeatedly). We assume that when the robot observes garbage it updates its knowledge base accordingly. In that case, all the conditions for firing the rule are satisfied: the robot knows there is garbage, $\mathsf{K}garbage$, is disposed to clean it, $\mathsf{GG}GarbageInBin$ and adopting this goal is not inconsistent with its current goals, $\neg\mathsf{G}\neg GarbageInBin$. Accordingly, the rule fires and the goal base of the agent is expanded with the first-order goal $GarbageInBin$. Consecutively, the robot will execute the left branch in the choice program and start cleaning the garbage by performing the $clean$ action.

5.3 Goal adoption in the agent logic

Finally, in the agent verification logic, analogously logical rules can be introduced to reason about the effects of goal adoption rules $\varphi \Rightarrow_G \psi$. These effects can be formalized by goal adoption axioms of the form:

$$(\mathsf{K}\varphi \wedge \mathsf{GG}\psi \wedge \neg\mathsf{G}\neg\psi) \rightarrow \bigcirc\mathsf{G}\psi$$

For any agent program, a set of goal adoption axioms corresponding to the goal adoption rules can be added to the agent verification logic to reason about the goal adoption mechanism of the agent and to verify its correctness. Semantically, the constraint imposed on the semantics of goals which required the goals of an agent to be static needs to be relaxed.

In particular, the third constraint in Constraint 1 needs to be adjusted accordingly for all models $M = \langle W, R, K, G, V \rangle$ and atomic actions a. To facilitate the introduction of the new semantic constraint that corresponds with the goal adoption mechanism, as a preliminary step, we first define the set e as follows:

$$e(w) = \bigcap_{\varphi \Rightarrow_G \psi \in \Gamma} \{w' \in W \mid M, w \models \mathsf{K}\varphi \wedge \mathsf{GG}\psi \wedge \neg\mathsf{G}\neg\psi, M, w R_a w', w' \models \psi\}$$

Then the third constraint in Constraint 1 is to be replaced with:

$$\forall w, w'(w R_a w' \Rightarrow G(w') = G(w) \cap e(w))$$

This new constraint illustrates the fact that the goals of an agent are expanded by firing goal adoption rules since all worlds that do not satisfy the adopted goals are removed from the set of alternative goal worlds $G(w)$.

Remark 2 It will be clear that care should be taken when goal adoption axioms are added to the agent logic to avoid inconsistency. A consistency check with the previously adopted goals of an agent is built into the semantics, but the consistency of the combined set of adopted goals is not incorporated. In the transition rule above, the consistency of the adoption set a itself is not checked. This remains a task for the designer or programmer, who has several methods available to ensure consistency. The most simple technique is to ensure that the triggering conditions of the rules are mutually exclusive in the sense that only one triggering condition can be true at any time. This may be too restrictive, however, and a more advanced approach would be to prove that the set of goals of arbitrary rule sets that may fire in the same state (i.e. the triggering conditions of these rules are not mutually exclusive) are consistent.

Goal adoption rules provide a mechanism for goal adoption based on the (second-order) dispositions of an agent, but also provide a means to implement adjustable autonomy. Note that from the axiom $\mathsf{GG}\varphi \vee \mathsf{G}\neg\mathsf{G}\varphi$ it follows that an agent will be either disposed to adopt φ as a (first-order) goal or will *not* be disposed to do that. Using this principle, the autonomy of agents can be explicitly adjusted by restricting or extending the set of second-order goals of an agent, i.e. to vary the set of first-order goals that an agent is disposed to generate itself and which not. The goals φ that the agent is not disposed to adopt, i.e. $\mathsf{G}\neg\mathsf{G}\varphi$ holds, still may be adopted by the agent, but not through self-generation. Such goals might be adopted e.g. through a mechanism for handling requests from other (human) agents. Second-order goals thus provide a concrete mechanism for implementing adjustable autonomy (cf. [3, 9]).

6 Conclusion

In this paper, a programming theory for rational agents has been presented. Such a theory consists of an agent programming language and a corresponding verification logic to prove the correctness of agent programs. The verification logic presented is a modal logic for rational agents that incorporates the core agent concepts of action, knowledge, goals and precompiled plans. A precise correspondence of the modal semantics of the logic and the operational semantics of the programming language has been established by providing a state-based semantics that grounds the modal agent logic. Additionally, a mechanism for adopting goals has been introduced based on a dispositional view of second-order goals. It has been argued that second-order goals viewed as dispositions to adopt goals can be used by the agent to self-generate goals and provide a means to the programmer to adjust the autonomy of the agent.

As discussed, the agent programming language presented is in many ways similar to other languages in the literature. Our main objective in this paper has been to establish a precise correspondence of the programming language with a modal verification logic for rational agents. The programming language presented is most similar to AgentSpeak(L) [36] and 3APL [7], but there are some differences. It has been our aim to present a programming theory for rational agents, which we believe should include the concept of a declarative goal, a difference with AgentSpeak(L) which does not include such goals explicitly. One of the differences with 3APL is the way goals are used to invoke plans. In 3APL plans can be invoked by conditions on the current state of the agent only, whereas in the programming language presented in this paper plans are invoked by statements referring to goals in a plan of the agent itself. Another difference our programming language provides goal adoption rules which are absent from 3APL.

The work reported in this paper builds on and extends earlier work of the authors [24, 25] in various ways. Whereas [24] provided an agent logic for a restricted version of 3APL, we here significantly extend the scope of applicability of the logic to include now also precompiled plans and declarative goals. In [25], we aimed at establishing a connection between the KARO logic [41] and a programming language. The results obtained show partial success in establishing a corresponding programming language, but, as reported, require quite a number of restrictions on the KARO logic due to its comprehensiveness and associated complexity. In contrast, here we have taken a different approach starting with a programming language and showing how a verification logic fitting for this language can be obtained.

Other related research that shares our objective of providing a verification framework for rational agents have addressed the issue mainly by investigating model checking techniques or techniques for directly executing agent logics. The latter aims at finding useful fragments of agent logics that can be executed *efficiently*. Relevant work in this area is, for example, that of [19, 26]. A drawback, however, is that these techniques are applicable only to relatively small fragments of agent logics. Instead, the work on verification of AgentSpeak(L) agents is based on model checking techniques [8]. The main problem faced in applying such techniques consists in computing finite models as input for the model checker as well as finding an appropriate agent logic to specify the properties to be verified. The verification logic for rational agents of this paper goes beyond the one presented in [8] in various ways, in particular with regards to the logical properties of the knowledge and goals of an agent. Moreover, in contrast with the rather syntactic definition of the semantics presented in [8] our semantics is firmly based on standard modal semantics. This fact is important since it allows us to reuse many of the standard results of modal logic and apply them to the theory presented, for example, in the construction of an axiomatization of the verification logic which remains for future work.

Although the benefits of model checking as used in [8] are that it allows for automated verification, our main concern has not been to provide an agent programming theory that can be used in combination with such techniques. Our main goal has been to provide a verification logic that, possibly supported by theorem proving tools, can be used to prove properties of agent programs, while taking into account that it should be possible to execute agent programs efficiently. Our approach toward the construction of such a theory, however, does not preclude the use of model checking techniques per se, although this would require a more detailed investigation of the complexity of the verification logic.

One important issue that remains for future research concerns the revision of goals. In the agent programming theory, a mechanism for goal expansion has been incorporated but not yet one for goal revision. An approach based on the introduction of an axiom like $\mathsf{K}\varphi \rightarrow \neg\mathsf{G}\varphi$ combined with an axiom to capture goal persistence $\mathsf{G}\varphi \rightarrow \bigcirc(\mathsf{G}\varphi \vee \mathsf{K}\varphi)$ does not work:

There is a simple counterexample to the goal persistence axiom with $\varphi = (p \wedge q)$ a conjunctive goal and the assumption that at the next time *only* p is achieved. In [39] an interesting approach for goal change has been proposed and we would like to study such proposals for goal revision in our programming theory as well.

Appendix: Proofs

Proof for Theorem 1

The left to right implication requires some work but is straightforward. Suppose $M^s = \langle W^s, R^s \rangle$ is a state-based DKL model such that $M^s, s \not\models_s \varphi$. We need to find a modal DKL model $M = \langle W, R, K, V \rangle$ and a world $w \in W$ such that $M, w \not\models \varphi$. The components of the modal model M can be constructed as follows:

- $W = W^s$,
- for states $s, s' \in W$, define $s R_a s'$ iff $s R^s_a s'$,
- for states $s = \langle v, k \rangle, s' = \langle v', k' \rangle \in W$, define $s K s'$ iff $v' \models k$ and $k = k'$, and
- for a state $s = \langle v, k \rangle \in W$, define $V(p, s) = 1$ iff $p \in v$.

Clearly, M is a modal DKL model. We show by induction on φ and π that:

$$\text{for all } s \in W\text{: } M, s \models \varphi \text{ iff } M^s, s \models_s \varphi, \text{ and}$$
$$\text{for all } s, s' \in W\text{: } s R_\pi s' \text{ iff } s R^s_\pi s'$$

So, assume $s = \langle v, k \rangle, s' = \langle v', k' \rangle$ are two arbitrary states.

- For formulas φ, we have the following cases:

$\varphi = p$:	By the construction of M we have $V(p, s) = 1$ iff $p \in v$.
$\varphi = \varphi_1 \wedge \varphi_2, \neg\psi$:	Immediate from the induction hypothesis,
$\varphi = [\pi]\psi$:	By the induction hypothesis, we may assume we have $s R_\pi s'$ iff $s R^s_\pi s'$ and $M, s' \models \psi$ iff $M^s, s' \models_s \psi$. Application of the semantic clause for $[\pi]$ then gives the required result.
$\varphi = \mathsf{K}\psi$:	By the induction hypothesis we have that $M, s' \models \varphi$ iff $M^s, s' \models_s \psi$ for all s'. By the construction of M, it then follows by application of the semantic clause for K that $M, s \models \mathsf{K}\psi$ iff $M^s, s \models_s \mathsf{K}\psi$.

- For programs π, all cases follow immediately from the construction of M and the induction hypothesis. For the **if-then-else**-statement, use the fact that for all states s, $M, s \models \varphi$ iff $M^s, s \models_s \varphi$ may be assumed to hold by the induction hypothesis.

For the right to left implication, we use the finite model property for the logic DKL [6] and the fact that the truth value of a formula φ depends only on the truth values of the propositional atoms that occur in φ. So, suppose that $M, w \not\models \varphi$. By the finite model property, we may assume that $M = \langle W, R, K, V \rangle$ is finite, i.e. W is finite.

The basic idea to construct a state-based DKL model from a finite, standard modal DKL model is to give *names* to each of the worlds in the finite standard DKL model. These names enable the identification of the world in the standard modal model in which a formula is being evaluated. Since the set of propositional atoms At is infinite and the truth of a formula φ depends only on the truth values of a finite number of atoms, we have an infinite stock of atoms available to name each of the finite number of worlds in W. To simplify the construction of a state-based model below, we may as well add a finite number of atoms $n_1, \ldots, n_m$

to the language to name the m worlds in M and assume a one-to-one mapping s such that $l(w) = n_i$ for some i. In the construction of a state-based model the name $l(w)$ of a world w in the modal model then will be included in the world state v of the state $s = \langle v, k\rangle$ corresponding with w. The construction of $M^s = \langle W^s, R^s\rangle$ from a given (finite) modal model $M = \langle W, R, K, V\rangle$ now proceeds as follows:

- $W^s = \{(v \cup \{l(w)\}, k) \mid v = \{p \mid V(p, w) = 1\}, k = \{\varphi \mid M, w \models \mathsf{K}\varphi, \varphi \in \mathcal{L}_0\}\}$,
- for $s = \langle v, k\rangle, s' = \langle v', k'\rangle \in W^s$, define: $sR^s_a s'$ iff $wR_a w'$ and $l(w) \in v$ and $l(w') \in v'$.

First, we will show that M^s is a state-based DKL model. That is, we prove that:

(i) for all $(v, k) \in W^s$: $v \models k$, and
(ii) if $(v, k) \in W^s$ and $v' \models k$, then $(v' \cup \{l(w'), k) \in W^s$ for some world $w' \in W$.

Before we prove that (i) and (ii) showing that M^s is a state-based model, we prove a fact about states in the state-based model that correspond to worlds w, w' that are K-related, i.e. wKw':

(iii) For all wKw' and states $(v, k), (v', k') \in W^s$ such that $l(w) \in v, l(w') \in v'$ we have: $k = k'$.

By the construction of the state-based model W^s, it is sufficient to show for $\varphi \in \mathcal{L}_0$ that $M, w \models \mathsf{K}\varphi$ iff $M, w' \models \mathsf{K}\varphi$. This follows immediately from the fact that K is an equivalence relation.

In order to prove (i) and (ii), let $(v, k) \in W^s$ be a state and $w \in W$ be the unique world such that $l(w) \in v$. By the construction of W^s and the assumption that s is one-to-one, there must be such a world.

To prove (i), we use the fact that the relation K in the modal model M is reflexive. Suppose that $\varphi \in k$. It follows from the construction of k that we have: $M, w \models \mathsf{K}\varphi$. Since K is reflexive, we then also have: $M, w \models \varphi$. Since φ is a state proposition, the truth of φ only depends on the assignment of truth values to propositional variables by V. By construction, $\{p \mid V(p, w) = 1\} \subseteq v$ and because φ was arbitrarily choosen, we have that $v \models k$.

To prove (ii), we show that there is a world $w' \in W$ such that wKw' and a corresponding state $(v', k') \in W^s$ such that $l(w') \in v'$. From fact (iii) it then follows that we also must have that $k' = k$. So, assume that for some v' we have $v' \models k$, and suppose, to arrive at a contradiction, that there is no world w' such that $(v' \cup \{l(w')\}, k') \in W^s$ and wKw'. Because W is finite, there then must be an objective proposition φ such that $v' \models \varphi$ and for all w' such that wKw' we have $M, w' \not\models \varphi$. As a result, we obtain $M, w \models \mathsf{K}\neg\varphi$ and $\neg\varphi \in k$ by the construction of W^s. But in that case we would have $v' \not\models k$, contradicting our initial assumption. This finishes the proof that M^s is a state-based model.

Finally, to complete the proof, we show by simultaneous induction on φ and π that:

$$\text{for all } w \in W \text{ and } s = (v, k) \in W^s \text{ such that } l(w) \in v\text{: } M, w \models \varphi \text{ iff } M^s, s \models_s \varphi, \text{ and}$$
$$\text{for all } w, w' \in W \text{ and } s = (v, k), s' = (v', k') \in W^s \text{ such that } l(w) \in v, l(w') \in v'\text{:}$$
$$wR_\pi w' \text{ iff } sR^s_\pi s'$$

- For propositions φ, the following cases apply:

$\varphi = p$: Immediate from the construction of M^s,
$\varphi = \varphi_1 \wedge \varphi_2$: Immediate from the induction hypothesis,
$\varphi = \neg\psi$: Immediate from the induction hypothesis,

$\varphi = [\pi]\psi$: By the induction hypothesis we have for arbitrary w, w' that $wR_\pi w'$ iff $sR^s_\pi s'$ for $s = \langle v, k\rangle, s' = \langle v', k'\rangle \in W^s$ with $l(w) \in v, l(w') \in v'$ and $M, w' \models \psi$ iff $M^s, s' \models_s \psi$ for $s' = (v', k') \in W^s$ with $l(w') \in v'$. Application of the semantic clause for $[\pi]$ then gives the required result.

$\varphi = \mathsf{K}\psi$: Suppose that $M, w \models \mathsf{K}\psi$ and $s = (v, k) \in W^s$ with $l(w) \in v$. For all $w' \in W$ such that wKw' we then have that $M, w' \models \varphi$. By the construction of M^s, the proof of fact (ii), fact (iii), and the induction hypothesis, this is equivalent to the fact that for all v' such that $v' \models k$ we have $M^s, (v', k) \models_s \varphi$. By the truth condition for K, this is equivalent to $M, v, k \models_s \mathsf{K}\varphi$.

– For programs π, all cases follow immediately from the construction of M^s and the induction hypothesis.

This concludes the proof of the theorem.

Proof for Theorem 2

Again, we first show the left to right implication. Suppose that $M^s = \langle W^s, R^s\rangle$ is a state-based DAL model and $s = \langle v, k, g\rangle$ a state including goals, such that $M^s, s \not\models_s \varphi$. Then we need to construct a modal DAL model $M = \langle W, R, K, G, V\rangle$ and a world $w \in W$ such that $M, w \not\models \varphi$. The construction proceeds as follows:

– $W = W^s$,
– for states $s, s' \in W$, define sR_as' iff sR^s_as',
– for states $s = \langle v, k, g\rangle, s' = \langle v', k', g'\rangle$, define sKs' iff $v' \models k', k = k', g = g'$,
– for states $s = \langle v, k, g\rangle, s' = \langle v', k', g'\rangle$, define sGs' iff $v', k' \models_s g(1)$, and $g' = g^{+1}$,
– for a state $s = \langle v, k, g\rangle$, define $V(p, (v, k, g)) = 1$ iff $p \in v$.

Clearly, M is a standard DAL model. We show by simultaneous induction on φ and π that:

$$\text{for all } s \in W\text{: } M, s \models \varphi \text{ iff } M^s, s \models_s \varphi, \text{ and}$$
$$\text{for all } s, s' \in W\text{: } sR_\pi s' \text{ iff } sR^s_\pi s'$$

– For propositions φ, the following cases apply:

$\varphi = p$: By the construction of M we have $V(p, (v, k, g)) = 1$ iff $p \in v$.

$\varphi = \varphi_1 \wedge \varphi_2, \neg\psi$: Immediate from the induction hypothesis,

$\varphi = [\pi]\psi$: By the induction hypothesis we have $sR^s_\pi s'$ iff $sR_\pi s'$ and $M^s, s' \models_s \psi$ iff $M, s' \models \psi$.

$\varphi = \mathsf{K}\psi$: By the induction hypothesis we have that $M^s, s' \models_s \psi$ iff $M, s' \models \psi$ for all s'. By the construction of M and the closure conditions on W^s we also have that $(v, k, g)K(v', k', g')$ iff $v' \models k, k = k'$ and $g = g'$. It follows that for $s = \langle v, k, g\rangle$ we have $M^s, s \models_s \mathsf{K}\psi$ iff $M^s, v', k, g \models_s \psi$ for all v' such that $v' \models k$ iff $M, (v', k, g) \models \psi$ for all v' such that $v' \models k$ iff $M, s' \models \psi$ for all $s' = (v', k', g')$ such that $(v, k, g)K(v', k', g')$ iff $M, s \models \mathsf{K}\psi$.

$\varphi = \mathsf{G}\psi$: By the induction hypothesis we have that $M, (v', k', g^{+1}) \models_s \psi$ iff $M, (v', k', g^{+1}) \models \psi$ for all v', k', g. By the construction of M and the closure conditions on W^s we also have that $(v, k, g)G(v', k', g')$ iff $(v', k', g) \models_s g(1)$ and $g' = g^{+1}$. It follows that $M, (v, k, g) \models_s \mathsf{G}\psi$ iff $M, (v', k', g^{+1}) \models_s \psi$ for all v', k' such that $(v', k', g) \models_s g(1)$ iff $M, (v', k', g^{+1}) \models \psi$ for all v', k' such that $(v', k', g) \models_s g(1)$ iff $M, (v', k', g') \models \psi$ for all (v', k', g') such that $(v, k, g)G(v', k', g')$ iff $M, (v, k, g) \models \mathsf{G}\psi$.

– For programs π, all cases immediately follow from the construction of M and the induction hypothesis.

For the right to left implication, we use the finite model property for DAL [6] and the fact that the truth value of a formula φ depends only on the truth values of the propositional atoms that occur in φ. Suppose that $M, w \not\models \varphi$. Then we may assume that $M = \langle W, R, K, G, V\rangle$ is finite, i.e. W is finite.

The basic idea to construct a state-based DAL model from a standard modal DAL model is similar to that for constructing a state-based DKL model. Since the set of atoms $At \subseteq \mathcal{L}_{DAL}$ is infinite, we then know that the truth of φ does not depend on the truth value of an infinite number of atoms and we can use these atoms as names for the finite number of worlds in W. To simplify the construction of the state-based model below, we may as well add a finite number of atoms $n_1, \ldots, n_m$ to the language to name the m worlds in M and assume a one-to-one mapping s such that $l(w) = n_i$ for some i. In the state-based model the name $l(w)$ of a world w in the standard model is included in the world state component of a state. The construction of $M^s = \langle W^s, R^s\rangle$ from the standard model $M = \langle W, R, K, G, V\rangle$ then proceeds as follows:

– $W^s = \{(v \cup \{l(w)\}, k, g) \mid v = \{p \mid V(p, w) = 1\}, k = \{\varphi \mid M, w \models \mathsf{K}\varphi, \varphi \in \mathcal{L}_0\}, g(i) = \{\varphi \mid M, w \models \mathsf{G}^{i+1}\varphi, \varphi \in \mathcal{L}_k\}\}$,
– $(v, k, g)R^s_a(v', k', g')$ iff wR_aw' and $l(w) \in v$ and $l(w') \in v'$.

We have to show that M^s is a state-based DAL model. That is, we need to prove that:

(i) for all $(v, k, g) \in W^s$: $v \models k$,
(ii) if $(v, k, g) \in W^s$ and $v' \models k$, then $(v' \cup \{l(w')\}, k, g) \in W^s$ for some world $w' \in W$, and
(iii) if $(v, k, g) \in W^s$ and $(v', k') \models_s g(1)$, then $(v' \cup \{l(w')\}, k', g^{+1}) \in W^s$ for some world $w' \in W$.

Before we prove that M^s is a state-based model, we prove two additional facts about the relations K and G between worlds in W and states in W^s:

(iv) For all wKw' and states $(v, k, g), (v', k', g') \in W^s$ such that $l(w) \in v, l(w') \in v'$ we have: $k = k'$ and $g' = g$.
(v) For all wGw' and states $(v, k, g), (v', k', g') \in W^s$ such that $l(w) \in v, l(w') \in v'$ we have that $(v', k') \models_s g(1)$ and $g' = g^{+1}$.

To prove (iv), we first prove that $k = k'$: By the construction of the state-based model M^s, it is clear that it is sufficient to show for a state proposition φ that $M, w \models \mathsf{K}\varphi$ iff $M, w' \models \mathsf{K}\varphi$. This follows from the fact that K is an equivalence relation. To show that $g' = g$, we use the fact that $M, w \models \mathsf{KG}\varphi \leftrightarrow \mathsf{G}\varphi$. It follows that $M, w \models \mathsf{KG}$ iff $M, w \models \mathsf{G}\varphi$. As a consequence, for any w' such that wKw' we have that $M, w' \models \mathsf{G}\varphi$ iff $M, w \models \mathsf{G}\varphi$ and the fact that $g = g'$ follows from the construction of M^s.

To prove (v), first, we have to show that $(v', k') \models_s g(1)$. Suppose, to arrive at a contradiction, that $\varphi \in g(1) \subseteq \mathcal{L}_k$ and $(v', k') \not\models_s \varphi$. By the construction of $g(1)$ we have that $M, w \models \mathsf{G}\varphi$ and as a result that $M, w' \models \varphi$ since wGw'. Since $v' = \{p \mid V(p, w') = 1\}$ and $k' = \{\varphi \mid M, w' \models \mathsf{K}\varphi\}$, and $\varphi \in \mathcal{L}_k$, we have that $M, w' \models \varphi$ iff $(v', k') \models_s \varphi$, contradicting our assumption that $(v', k') \not\models_s \varphi$.[6]

[6] Proof by induction on φ that: $M, w \models \varphi$ iff $(v, k) \models_s \varphi$ for all $\varphi \in \mathcal{L}_k$ and $v = \{p \mid V(p, w) = 1\}$ and $k = \{\varphi \mid M, w \models \mathsf{K}\varphi\}$. The cases for atoms p and boolean combinations of propositions are easy. So, suppose φ has the form $\mathsf{K}\psi$. Then we have that $M, w \models \mathsf{K}\psi$ iff $M, w' \models \psi$ for all wKw' iff $(v', k) \models_s \psi$ for all $v' \models k$ by the induction hypothesis and fact (iv) iff $(v, k) \models_s \mathsf{K}\psi$.

Second, we have to show that $\{\varphi \mid M, w \models \mathsf{G}^{i+2}\varphi, \varphi \in \mathcal{L}_k\} = g(i+1) = g^{+1}(i) = g'(i) = \{\varphi \mid M, w' \models \mathsf{G}^{i+1}\varphi, \varphi \in \mathcal{L}_k\}$. It is sufficient to show that $M, w \models \mathsf{G}^{i+2}\varphi$ iff $M, w' \models \mathsf{G}^{i+1}\varphi$ for wGw'. The left to right implication is trivial and follows from the truth conditions for G. For the right to left implication, use the axiom schema of Lemma 1. So, suppose that $M, w' \models \mathsf{G}^{i+1}\varphi$ for a contingent $\varphi \in \mathcal{L}_g$ (for tautologies the proof is trivial). By the finiteness of W, we have that there is a contingent $\psi \in \mathcal{L}_g$ such that $M, w \models \mathsf{G}^{i+2}\psi$. It follows that we have $M, w \models \mathsf{G}^{i+1}(\mathsf{G}\psi \vee \mathsf{G}\varphi)$. By Lemma 1 we then also have $M, w \models \mathsf{G}^{i+2}\varphi$ and we are done.

The proof of (i) and (ii) is similar to the analogous cases in Theorem 1. Use fact (iv) to prove (ii).

To prove (iii), we show that there is a world $w' \in W$ such that wGw' and, by the construction of M^s, a state $(v', k', g') \in W^s$ such that $l(w') \in v'$. Note that from (v) it follows that in that case we must have $g' = g^{+1}$. Now, let v', k' be an arbitrary world state and knowledge base such that $(v', k') \models_s g(1)$. Suppose, to arrive at a contradiction, that there is no world $w' \in W$ such that $l(w') \in v'$. In that case, using the finiteness of M, there is a φ such that $(v', k') \models_s \varphi$ and for all w' such that wGw' we have that $M, w' \not\models \varphi$. It follows that $M, w \models \mathsf{G}\neg\varphi$ and $\neg\varphi \in g(1)$, contradicting the fact that $(v', k') \models_s g(1)$.

We show by simultaneous induction on φ and π that:

for all $w \in W$ and $s = (v, k, g) \in W^s$ such that $l(w) \in v$:

$M, w \models \varphi$ iff $M^s, s \models_s \varphi$, and

for all $w, w' \in W$ and $s = (v, k, g), s' = (v', k', g') \in W^s$ such that $l(w) \in v, l(w') \in v'$:

$w R_\pi w'$ iff $s R^s_\pi s'$

– For propositions φ, all proofs except for the case that $\varphi = \mathsf{G}\psi$ are analogous to those of Theorem 1. For the proof of $\varphi = \mathsf{K}\psi$ use fact (i), (ii) and (iv) above. We give the proof for the case that $\varphi = \mathsf{G}\psi$:

  $\varphi = \mathsf{G}\psi$: Suppose that $M, w \models \mathsf{G}\psi$ and $s = (v, k, g) \in W^s$ such that $l(w) \in v$. Then for all w' such that wGw' we have $M, w' \models \psi$. By the induction hypothesis, and fact (iii) and (v), this is equivalent to the fact that for all v', k' such that $(v', k', g) \models_s g(1)$ we have that $M^s, (v', k', g^{+1}) \models_s \psi$. By the truth condition for G, this is equivalent to $M, s \models_s \mathsf{G}\psi$.

– For programs π, the proof is completely analogous to that for Theorem 1.

This concludes the proof of the theorem.

References

1. Baral, C., & Son, T. C. (1998). Relating theories of action and reactive control. *Electronic Transactions on Artificial Intelligence, 2*, 211–271.
2. Baral, C., Son, T. C., & Tuan, L. C. (2002). A transition function based characterization of actions with delayed and continuous effects. In *Proceedings of the 8th International Conference on Principles and Knowledge Representation and Reasoning (KR'02)*, pp. 291–301.
3. Barber, K. S., & Martin, C. E. (1999). Agent autonomy: Specification, measurement, and dynamic adjustment. In *Proceedings of the Autonomy Control Software Workshop*, pp. 8–15.

4. Birna van Riemsdijk, M., de Boer, F. S., Dastani, M., & Meyer, J.-J. Ch. (2006). Prototyping 3APL in the Maude term rewriting language. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06)*, pp. 1279–1281.
5. Birna van Riemsdijk, M., de Boer, F.S., & Meyer, J.-J. Ch. (2006). Dynamic logic for plan revision in agent programming. *Journal of Logic and Computation, 16*(3), 375–402.
6. Blackburn, P., de Rijke, M., & Venema, Y. (2001). *Modal logic*. Cambridge University Press.
7. Bordini, R. H., Dastani, M., Dix, J., & El Fallah Seghrouchni, A. (Eds.) (2005). *Multi-agent programming: Languages, platforms and applications*. Springer.
8. Bordini, R. H., Fisher, M., Visser, W., & Wooldridge, M. (2006). Verifying multi-agent programs by model checking. *Journal of Autonomous Agents and Multi-Agent Systems, 12*(2), 239–256.
9. Bradshaw, J. M., Jung, H., Kulkarni, S., Johnson, M., Feltovich, P., Allen, J., Bunch, L., Chambers, N., Galescu, L., Jeffers, R., Suri, N., Taysom, W., & Uszok, A. (2005). Toward trustworthy adjustable autonomy in KAOS. In *Trusting Agents for Trusting Electronic Societies*, pp. 18–42.
10. Bratman, M. E. (1987). *Intention, plans, and practical reasoning*. Harvard University Press.
11. Chellas, B. F. (1980). *Modal logic: An introduction*. Cambridge University Press.
12. Cohen, P. R., & Levesque, H. J. (1990). Intention is choice with commitment. *Artificial Intelligence, 42*, 213–261.
13. Dastani, M., de Boer, F. S., Dignum, F. P. M., & Meyer, J.-J. Ch. (2003). Programming agent deliberation: An approach illustrated using the 3APL language. In *Proceedings of the 2nd Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03)*, pp. 97–104.
14. Dastani, M. M., van Riemsdijk, M. B., Dignum, F. P. M., & Meyer, J.-J. Ch. (2004). A programming language for cognitive agents: Goal-directed 3APL. In M. Dastani, J. Dix, & A. El Fallah-Seghrouchni (Eds.), *Proceedings of the 1st International Workshop on Programming Multi-Agent Systems (ProMAS'03)* (pp. 111–130).
15. de Bakker, J. (1980). *Mathematical theory of program correctness*. Prentice-Hall.
16. de Boer, F. S., Hindriks, K. V., van der Hoek, W., & Meyer, J.-J. Ch. (2007). A verification framework for agent programming with declarative goals. *Journal of Applied Logic, 5*, 277–302.
17. Fagin, R., Halpern, J. Y., Moses, Y., & Vardi, M. Y. (1995). *Reasoning about knowledge*. MIT Press.
18. Fagin, R., Halpern, J. Y., & Vardi, M. Y. (1991). A model-theoretic analysis of knowledge. *Journal of the ACM, 38*(2), 382–428.
19. Fisher, M. (2006). METATEM: The story so far. In *Proceedings of the 3rd International Workshop on Programming Multiagent Systems (ProMAS'05), Revised and Invited Papers*. Lecture Notes in Computer Science, 3862, Springer, Berlin, pp. 3–22.
20. Frankfurt, H. (1971). Freedom of the will and the concept of a person. *Journal of Philosophy, 68*(1), 5–20.
21. Georgeff, M. P., & Lansky, A. L. (1987). Reactive reasoning and planning. In *Proceedings of the 6th National Conference on Artificial Intelligence*, pp. 677–682.
22. Harel, D., Kozen, D., & Tiuryn, J. (2000). *Dynamic logic*. MIT Press.
23. Hindriks, K. V., de Boer, F. S., van der Hoek, W., & Meyer, J.-J. Ch. (2000). Agent programming with declarative goals. In *Proceedings of the 7th International Workshop on Intelligent Agents. Agent Theories Architectures and Languages (ATAL'00)*, pp. 228–243.
24. Hindriks, K. V., de Boer, F. S., van der Hoek, W., & Meyer, J.-J. Ch. (2001). A programming logic for part of the agent language 3APL. In *Proceedings of the 1st Goddard Workshop on Formal Approaches to Agent-Based Systems*, pp. 78–89.
25. Hindriks, K. V., & Meyer, J.-J. Ch. (2007). Agent logics as program logics: Grounding KARO. In *Proceedings of the 29th Annual German Conference on AI (KI'06)*.
26. Hustadt, U., Dixon, C., Schmidt, R. A., Fisher, M., Meyer, J.-J. Ch. & van der Hoek, W. (2001). Reasoning about agents in the KARO framework. In *Proceedings of the 8th International Symposium on Temporal Representation and Reasoning (TIME'01)*, pp. 206–213.
27. Lifschitz, V. (1986). On the semantics of STRIPS. In *Reasoning about Actions and Plans*, pp. 1–9.
28. Luck, M., & d'Inverno, M. (1998). Motivated behaviour for goal adoption. In *Selected Papers from the 4th Australian Workshop on Distributed Artificial Intelligence, Multi-Agent Systems: Theories, Languages, and Applications*, pp. 58–73.
29. McCarthy, J., & Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, pp. 463–502.
30. Meyer, J.-J. Ch. & van der Hoek, W. (1995). *Epistemic logic for AI and computer science*. Cambridge: Cambridge University Press.
31. Moffat, D., & Frijda, N. H. (1995). Where there's a will there's an agent. In *Intelligent Agents, ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, pp. 245–260.
32. Nguyen, L. A. (2005). On the complexity of fragments of modal logics. In *Advances in Modal Logic, 5*, 249–268.

33. Norman, T. J., & Long, D. (1995). Goal creation in motivated agents. In *Proceedings of the Workshop on Agent Theories, Architectures, and Languages*, pp. 277–290.
34. Plotkin, G. D. (1981). A structural approach to operational semantics. Technical Report DAIMI FN-19, University of Aarhus.
35. Pokahr, A., Braubach, L., & Lamersdorf, W. (2005). A goal deliberation strategy for BDI agent systems. In *Third German Conference on Multi-Agent Technologies and Systems (MATES'05)*, pp. 82–93.
36. Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In *Agents Breaking Away*, pp. 42–55.
37. Rao, A. S., & Georgeff, M. P. (1993). Intentions and rational commitment. Technical Report 8, Australian Artificial Intelligence Institute.
38. Sardina, S., & Shapiro, S. (2003). Rational action in agent programs with prioritized goals. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03)*, pp. 417–424.
39. Shapiro, S., Lesperance, Y., & Levesque, H. (2005). Goal change. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pp. 582–588.
40. Sloman, A., & Croucher, M. (1981). Why robots will have emotions. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI'81)*, pp. 197–202.
41. van der Hoek, W., van Linder, B., & Meyer, J.-J. Ch. (1999). An integrated modal approach to rational agents. In *Foundations of Rational Agency* (pp. 133–168). Dordrecht: Kluwer.
42. van der Hoek, W., & Wooldridge, M. (2003). Towards a logic of rational agency. *Logic Journal of the IGPL, 11*(2), 133–157.
43. van Ditmarsch, H., van der Hoek, W., & Kooi, B. (2007). *Dynamic epistemic logic*. Springer.