Agent programming in Dribble: from beliefs to goals using plans

[Extended Abstract]

Birna van Riemsdijk *
Computer Science
Utrecht University
The Netherlands
birna@cs.uu.nl

Wiebe van der Hoek
Computer Science
University of Liverpool
United Kingdom
wiebe@csc.liv.ac.uk

John-Jules Ch. Meyer
Computer Science
Utrecht University
The Netherlands
jj@cs.uu.nl

ABSTRACT

To support the practical development of intelligent agents, several programming languages have been introduced that incorporate concepts from agent logics: on the one hand, we have languages that incorporate beliefs and plans (i.e., procedural goals), and on the other hand, languages that implement the concepts of beliefs and (declarative) goals. We propose the agent programming language Dribble, in which these features of procedural and declarative goals are combined. The language Dribble thus incorporates beliefs and goals as well as planning features. The idea is, that a Dribble agent should be able to select a plan to reach a goal from where it is at a certain point in time. In order to do that, the agent has beliefs, goals and rules to select plans and to create and modify plans. Dribble comes with a formally defined operational semantics and, on top of this semantics, a dynamic logic is constructed that can be used to specify and verify properties of Dribble agents. The correspondence between the logic and the operational semantics is established.

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*Semantics*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*Logics of programs*; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software

General Terms

Languages, Theory

*PhD student in computer science

AAMAS'03, July 14–18, 2003, Melbourne, Australia.

Keywords

Intelligent agent, agent-oriented programming, declarative goals, plans, practical reasoning rule

1. INTRODUCTION

In research on agents, besides architectures, the areas of agent theories and agent programming languages are distinguished.

Theories concern descriptions of (the behaviour of) agents. Agents are often described using logic ([10, 12]). Concepts that are commonly incorporated in such logics are for instance knowledge, beliefs, desires, intentions, commitments, goals and plans.

To support the practical development of intelligent agents, several programming languages have been introduced that incorporate concepts from agent logics. Firstly, there is a family of languages using goals as procedural (*goal to do*). Whether they are called commitments (Agent-0, [11]), intentions (AgentSpeak(L), [8]), or goals (3APL, [3]) makes little difference: all these notions are structures built from *actions* and therefore similar in nature to *plans*. In [2] and [5] it was argued that the languages AgentSpeak(L) and a restricted version of ConGolog can be embedded in 3APL. At the same time, to reason about 3APL, one could try using a BDI ([9]) or KARO ([13]) like logic, incorporating beliefs and plans. Such logics however also incorporate a declarative notion of goals, which was typically missing in 3APL (and, by virtue of the embedding, also in the other two languages). To bridge this gap, the programming language GOAL was proposed ([4]). This language takes the declarative point of view on goals (*goal to be*).

The language that we propose in this paper constitutes a synthesis between the declarative and the procedural approaches, combining both notions in one and the same programming language. The idea is, that in such a language a more precise mapping with logical specification languages like BDI or KARO can be obtained. To be more specific, we define the language Dribble, in which the declarative and procedural features of the languages GOAL and 3APL are combined. The language 3APL offers the means for *creating and modifying plans* during the execution of the agent. GOAL agents on the other hand do not have planning features, but they do offer the possibility to *use declarative goals* to select actions. The language Dribble thus incorpo-

rates beliefs and goals as well as planning features. The idea is, that a Dribble agent should be able to select a plan to reach a goal from where it is at a certain point in time. In order to do that, the agent has beliefs, goals and rules to select plans and to create and modify plans. When we state an agent has *planning* features, we thus mean that this agent is capable of constructing and modifying a plan (a sequence of actions meant to achieve some goal). Planning is thus not understood as deliberation and the agent has no way of doing lookahead. Programming agent deliberation can however be done in a meta-language, using 3APL or Dribble as the object language ([1]).

2. THE PROGRAMMING LANGUAGE DRIBBLE

2.1 Beliefs and goals

Our agent keeps two databases: a *belief base* and a *goal base* which are both sets of formulas from the propositional language $\mathcal{L}$. The difference between the two databases originates from the different meaning assigned to sentences from the belief base and goal base. Belief and goal formulas are used to refer to sentences from the belief base and/or goal base.

Definition 1. (belief and goal formulas) The language $\mathcal{L}$ is a propositional language with typical formula ϕ and the connectives $\wedge$ and $\neg$ with the usual meaning. The set of belief and goal formulas $\mathcal{L}_{BG}$ with typical formula φ is defined by:

- if $\phi \in \mathcal{L}$, then $\mathbf{B}\phi, \mathbf{G}\phi \in \mathcal{L}_{BG}$
- if $\varphi_1, \varphi_2 \in \mathcal{L}_{BG}$, then $\neg\varphi_1, \varphi_1 \wedge \varphi_2 \in \mathcal{L}_{BG}$.

Formulas with only **B**-operators are also called belief formulas, comprising the language $\mathcal{L}_B$ with typical element β, and similar for $\mathcal{L}_G$, the language with only **G**-operators.

2.2 Plans

In order to reach its goals, a Dribble agent adopts plans. A plan is a sequence of *basic elements* of a plan. There are three types of basic elements: basic actions, abstract plans and if-then-else constructs. As in the languages GOAL and 3APL, basic actions specify the capabilities with which an agent should achieve a certain state of affairs. The effect of execution of a basic action is not a change in the world, but a change in the belief base of the agent. The state of affairs that can be realized through execution of basic actions, is therefore a state of affairs of the belief base. The second basic element of a plan is the *abstract plan*. An abstract plan cannot be executed directly in the sense that it updates the belief base of an agent. Abstract plans serve as an abstraction mechanism like procedures in imperative programming. If a plan consists of an abstract plan, this abstract plan could be transformed into basic actions through reasoning rules. The third basic element is the if-then-else construct (see definition 2).
Plans are sequences of basic elements, which is defined in definition 2 below. In this definition the set of executable actions consisting of basic actions and if-then-else constructs is defined. In section 5.1 it will be convenient to have this set.

Definition 2. (plans) Assume that a set BasicAction is given, together with a set AbstractPlan. Let ExecutableAction be BasicAction $\cup$ {`if` β `then` π_1 `else` π_2 `fi`} ($\beta \in \mathcal{L}_B$ and $\pi_1, \pi_2 \in$ Plan). The symbol E denotes the empty plan.

- $E \in$ Plan,
- BasicAction $\cup$ AbstractPlan $\subseteq$ Plan,
- if $\pi_1, \pi_2 \in$ Plan and $\beta \in \mathcal{L}_B$ then `if` β `then` π_1 `else` π_2 `fi` $\in$ Plan,
- if $c \in$ ExecutableAction $\cup$ AbstractPlan $\cup \{E\}$ and $\pi \in$ Plan then $c\,;\pi \in$ Plan.

2.3 Mental state

Above, the beliefs, goals and plans of a Dribble agent were defined. These are the elements that change during the execution of the agent and together constitute the mental state of an agent. The beliefs of the agent describe the state (of the world) the agent is in (or believes to be in) and the goals describe the state (of the world) the agent wants to realize (or wants to believe to have realized). Basic actions change the beliefs of the agent. We will now discuss how the goals of the agent are changed.
The question is: when should agents adopt or drop goals? The way in which an agent does this, is called a *commitment strategy* ([4]). The strategy that was chosen for Dribble agents is as follows: an agent drops a goal if and only if it believes that that goal has been achieved. The goals of the agent can thus only be updated through belief updates. In this way, a *meaningful relation between beliefs and goals* is established.
If an agent behaves according to this commitment strategy, the mental state of the agent will have to meet the requirement that an agent does not believe that ϕ is the case if it has a goal to achieve ϕ. If it believes ϕ, it cannot have the goal to achieve ϕ, because the state of affairs ϕ is already reached. Another requirement is that a particular goal of an agent is consistent. It cannot desire a state of affairs which cannot be reached by definition. The last requirement on mental states of Dribble agents is that their belief bases are consistent. An agent with an inconsistent belief base would believe everything which is not a desirable situation. It could hardly function, because all requirements on beliefs in rules would be met, but none of the requirements on goals would be: the agent would believe everything and therefore would not have any goals.
The goals of an agent do not have to be mutually consistent, because two inconsistent goals could both be realized although not at the same time. A consequence of this is, that the agent cannot adopt the conjunction of two separate goals in the goal base as a new goal, for this could lead to inconsistent goals.

Definition 3. (mental state of a Dribble agent) A mental state of a Dribble agent is a triple $\langle \sigma, \gamma, \pi \rangle$ where $\sigma \subseteq \mathcal{L}$ are the agent's beliefs and $\gamma \subseteq \mathcal{L}$ are the agent's goals and $\pi \in$ Plan is the agent's plan. σ and γ are such that for any $\psi \in \gamma$ we have:

- ψ is not entailed by the agent's beliefs ($\sigma \not\models \psi$), and
- ψ is consistent ($\psi \not\models \bot$), and
- σ is consistent ($\sigma \not\models \bot$).

Σ with typical element s is the set of all possible mental states.

2.4 Rules

The plan of an agent can be changed through the application of rules or execution of executable actions (basic actions or if-then-else constructs). There are two kinds of rules: *goal rules* and *Practical Reasoning (PR) rules*.

Definition 4. (rules) A *goal rule* g is a pair $\varphi \rightarrow \pi$ such that $\varphi \in \mathcal{L}_{BG}$ and $\pi \in \mathsf{Plan}$. A *PR rule* ρ is a triple $\pi_h \mid \beta \rightarrow \pi_b$ such that $\beta \in \mathcal{L}_B$ and $\pi_h, \pi_b \in \mathsf{Plan}$.

A goal rule means that the plan π can be adopted, if the mental condition φ holds. Goal rules are used to *select* plans to reach a goal from a certain state. The conditions in the goal rule on the beliefs of an agent are used to describe the situation in which it could be a good idea to execute the plan. The conditions on goals in φ specify what the plan is good for.
The informal reading of a PR rule $\pi_h \mid \beta \rightarrow \pi_b$ is the following: if the agent has adopted the plan π_h and if it believes the belief formula β to be the case, the plan π_h can be replaced by the plan π_b. PR rules can be used to *create* plans (often from abstract plans), to *modify* plans and to model *reactive behaviour* (using PR rules with empty head).

2.5 A Dribble agent

To program a Dribble agent means to specify its initial beliefs and goals and to write sets of goal rules and PR rules. This is formalized in the specification of a Dribble agent.

Definition 5. (Dribble agent) A Dribble agent is a quadruple $\langle \sigma_0, \gamma_0, \Gamma, \Delta \rangle$ where $\langle \sigma_0, \gamma_0, E \rangle$ is the initial mental state, Γ is a set of goal rules and Δ is a set of PR rules.

The plan of the agent should be empty in the initial mental state. This choice reflects the idea that an agent should start to act because it has certain goals. A Dribble agent should adopt plans to reach these goals with the goal rules. Giving the agent a plan at start up is in contradiction with this idea.
Like agents searching for a plan, a Dribble agent has a goal state, a starting state and action specifications. A Dribble agent however does not search, but it uses its goal rules to select a plan. In these rules, the programmer can specify the goal state. The execution of the plan of the rule is supposed to help in bringing about this goal state. It is thus the programmer that should envision the result of a plan and the programmer should specify this in the rules. The agent cannot reason about the results of its actions, thus selecting actions that would result in realisation of the goal state. This is the job of the programmer.
The advantage of letting the programmer do this job is, that the agent does not have to search through a huge search space. It could then end up doing nothing, because all of its time is consumed by searching for an appropriate plan. A Dribble agent is moreover more flexible than a searching agent, because the agent can easily adapt its plan with its PR rules during execution. It does not have to start searching all over if the world changes in some way during the execution of the plan. We do not intend to say that searching or deliberation is never a good idea. A balance should be achieved between deliberation and plan execution.

3. OPERATIONAL SEMANTICS

The mental state of a Dribble agent consists of beliefs, goals and a plan. The mental state changes as a consequence of the execution of actions and the application of rules. We now give an operational semantics of the possible mental state changes using transition systems ([7]). First we will however define the meaning of a statement about the beliefs or goals of the agent.

3.1 Semantics of belief and goal formulas

Definition 6. (semantics of belief and goal formulas) Let $\langle \sigma, \gamma, \pi \rangle$ be a mental state, $\phi, \psi \in \mathcal{L}$ and $\varphi, \varphi_1, \varphi_2 \in \mathcal{L}_{BG}$.

- $\langle \sigma, \gamma, \pi \rangle \models \mathbf{B}\phi$ iff $\sigma \models \phi$
- $\langle \sigma, \gamma, \pi \rangle \models \mathbf{G}\psi$ iff for some $\psi' \in \gamma : \psi' \models \psi$ and $\sigma \not\models \psi$
- $\langle \sigma, \gamma, \pi \rangle \models \neg\varphi$ iff $\langle \sigma, \gamma, \pi \rangle \not\models \varphi$
- $\langle \sigma, \gamma, \pi \rangle \models \varphi_1 \wedge \varphi_2$ iff $\langle \sigma, \gamma, \pi \rangle \models \varphi_1$ and $\langle \sigma, \gamma, \pi \rangle \models \varphi_2$

A formula $\mathbf{B}\phi$ is true in a mental state, if and only if the belief base σ models ϕ. In this way, the agent believes all logical consequences of its beliefs. The semantics of $\mathbf{G}\psi$ is defined in terms of separate goals, as opposed to defining it in terms of the entire goal base. All logical consequences of a particular goal are also goals, but only if they are not believed ([4]). Two separate goals may be inconsistent and should therefore not be combined into one goal.

3.2 Transition system

Transition systems are a means to define the operational semantics of a programming language. A transition system consists of a set of derivation rules for deriving transitions for an agent. A transition is a transformation of one mental state into another and it corresponds to a single computation step. Here, we leave out the transition rule for execution of an if-then-else construct.
A goal rule is applicable in a mental state if the antecedent of the goal rule is true in that mental state. It can only be applied if the plan of the agent is empty. The idea is, that an agent can only select a new plan if it has finished executing its old plan. In the resulting mental state, the plan becomes equal to the consequent of the goal rule.

Definition 7. (application of a goal rule) Let $g : \varphi \rightarrow \pi \in \Gamma$ a goal rule.

$$\frac{\langle \sigma, \gamma, E \rangle \models \varphi}{\langle \sigma, \gamma, E \rangle \rightarrow_{\texttt{applyRule}(g)} \langle \sigma, \gamma, \pi \rangle}$$

A PR rule is applicable in a mental state, if the plan of the agent is equal to the head of the rule and if the guard of the rule is true in that mental state. The result of the application of a PR rule is that the plan in the body of the rule is adopted, replacing the plan which was equal to the head of the rule.

Definition 8. (application of a PR rule) Let $\rho : \pi_h \mid \beta \rightarrow \pi_b \in \Delta$ a PR rule.

$$\frac{\langle \sigma, \gamma, \pi_h \rangle \models \beta}{\langle \sigma, \gamma, \pi_h \rangle \rightarrow_{\texttt{applyRule}(\rho)} \langle \sigma, \gamma, \pi_b \rangle}$$

Basic actions update the belief base of an agent if they are executed. These belief updates are formally represented by a partial function $\mathcal{T}$, where $\mathcal{T}(a, \sigma)$ returns the result of updating belief base σ by performing action a. The fact that $\mathcal{T}$ is a partial function represents the fact that an action may not be executable in some belief states. Goals that are reached through execution of an action are removed from the goal base. After execution of an action, the action is removed from the plan.

Definition 9. (basic action execution) Let $\gamma' = \gamma \setminus \{\psi \in \gamma \mid \sigma' \models \psi\}$.

$$\frac{\mathcal{T}(a, \sigma) = \sigma'}{\langle \sigma, \gamma, a \rangle \rightarrow_{\texttt{execute(a)}} \langle \sigma', \gamma', E \rangle}$$

A plan can be either a basic element or a sequence of basic elements. The meaning of executing a basic element (for instance a basic action) or a certain fixed sequence of basic elements (the head of a PR rule), was defined in the previous transition rules. In the transition rule for execution of sequential composition, it is defined what it means to execute an arbitrary sequence of basic elements (a plan). The first part of the plan is executed first, any changes to the belief base and goal base are recorded and the agent continues executing the remainder of the plan.

Definition 10. (execution of sequential composition) Let $x \in \{\texttt{applyRule}(\rho), \texttt{execute}(a), \texttt{execute}(\texttt{if } \beta \texttt{ then } \pi_1 \texttt{ else } \pi_2 \texttt{ fi})\}$.

$$\frac{\langle \sigma, \gamma, \pi_1 \rangle \rightarrow_x \langle \sigma', \gamma', \pi_1' \rangle}{\langle \sigma, \gamma, \pi_1 \circ \pi_2 \rangle \rightarrow_x \langle \sigma', \gamma', \pi_1' \circ \pi_2 \rangle}$$

In this rule, we use the operator $\circ$. It is used to denote that some plan is a sequence of basic elements of which π_1 (or π_1') is the first part and π_2 is the second part: π_1 (or π_1') is the prefix of this plan. This prefix can be executed and the rest of the plan remains to be executed.
The transition rule specifies that it can only be used if the transition that is used as the premise of the rule, was not derived with the transition rule for application of a goal rule: $x \neq \texttt{applyRule}(g)$. The reason is, that goal rules should only be applied if the plan of the agent is empty. It should therefore not be possible to derive some transition $\langle \sigma, \gamma, E \circ \pi' \rangle \rightarrow_{\texttt{applyRule}(g)} \langle \sigma, \gamma, \pi \circ \pi' \rangle$ with $g : \varphi \rightarrow \pi \in \Gamma$ a goal rule. If this transition could be derived, the goal rule would be applicable in a mental state with a plan that is not empty. This transition could be derived through application of the transition rule for sequential composition, after deriving the transition $\langle \sigma, \gamma, E \rangle \rightarrow_{\texttt{applyRule(g)}} \langle \sigma, \gamma, \pi \rangle$ with the rule for application of a goal rule (see definition 7). The restriction that the transition rule for sequential composition cannot be applied to a transition $s_i \rightarrow_{\texttt{applyRule}(g)} s_{i+1}$, is thus necessary to prevent the derivation of the transition $\langle \sigma, \gamma, E \circ \pi' \rangle \rightarrow_{\texttt{applyRule}(g)} \langle \sigma, \gamma, \pi \circ \pi' \rangle$ from the transition $\langle \sigma, \gamma, E \rangle \rightarrow_{\texttt{applyRule(g)}} \langle \sigma, \gamma, \pi \rangle$.

3.3 Semantics of a Dribble agent

The semantics of a Dribble agent is derived directly from the transition relation $\rightarrow$. The meaning of a Dribble agent consists of a set of so called computation runs.

Definition 11. (computation run) A computation run $\texttt{CR}(s_0)$ for a Dribble agent with goal rules Γ and PR rules Δ is a finite or infinite sequence $s_0, \ldots, s_n$ or $s_0, \ldots$ where $s_i \in \Sigma$ are mental states, and $\forall_{i>0} : \ s_{i-1} \rightarrow_x s_i$ is a transition in the transition system for the Dribble agent.

The meaning of a Dribble agent $\langle \sigma_0, \gamma_0, \Gamma, \Delta \rangle$ is the set of computation runs $\texttt{CR}(\langle \sigma_0, \gamma_0, E \rangle)$. Note that the first state of the computation runs is the initial mental state of the Dribble agent.
The transition rules for goal rule application and PR rule application do not specify which rule to apply, in case more than one rule is applicable in a mental state. The transition rules neither specify whether to apply a rule or to execute an action, if both are possible in a mental state. Any implementation of the language Dribble has to deal with how to reduce this non-determinism in the semantics of the language. Control structures are needed to determine whether an action should be executed or whether a rule should be applied and possibly which rule should be applied.

4. EXAMPLE

Our example agent has to solve the problem of building a tower of blocks. The blocks have to be stacked in a certain order: block C has to be on the floor, B on C and block A on B. Initially, the blocks A and B are on the floor, while C is on A. The only action an agent can take, is to move a block x from some block y to another block z ($move(x, y, z)$). The action is enabled only if the block to be moved (x) and the block to which x is moved (z) are clear. The result of the action is, that x is on z and not on y, block y becomes clear and block z is not clear anymore (assuming that z is not the floor, because the floor is always clear). In the example, variables are used as a means for abbreviation. Variables should be thought of as being instantiated with the relevant arguments in such a way that predicates with variables reduce to propositions. Let

$$\begin{array}{rcl} \sigma_0 & = & \{on(C, A) \wedge on(A, Fl) \wedge on(B, Fl) \wedge clear(B) \wedge \\ & & \quad clear(C) \wedge clear(Fl),\ on(x, y) \rightarrow \neg clear(y)\}, \\ \gamma_0 & = & \{on(A, B) \wedge on(B, C) \wedge on(C, Fl)\} \end{array}$$

where $y \neq Fl$.

4.1 The rules

A Dribble agent can solve the tower building problem with the following rules.

$$\begin{array}{rcll} \Gamma & = & \{g_1 : & \mathbf{B}(on(x, y)) \wedge \mathbf{G}(on(x, z)) \rightarrow move(x, y, z)\} \\ \Delta & = & \{\rho_1 : & move(x, y, z) \mid \mathbf{B}(\neg clear(x) \wedge on(a, x)) \rightarrow \\ & & & \quad move(a, x, Fl); move(x, y, z), \\ & & \rho_2 : & move(x, y, z) \mid \mathbf{B}(\neg clear(z) \wedge on(a, z)) \rightarrow \\ & & & \quad move(a, z, Fl); move(x, y, z)\} \end{array}$$

The goal rule is used to derive the $move(x, y, z)$ action that should be executed to fulfil a goal $on(x, z)$. The preconditions of the move action are not checked in this rule, so it is possible that the derived action cannot be executed in a particular mental state. The PR rules can then be used to create a mental state in which this action *can* be executed. Note that the goal rule is used to select an action to fulfil a single proposition of the form $on(x, z)$. The initial goal base however contains a conjunction of $on(x, z)$ propositions. The goal rule is applicable to this conjunction, because a formula $\mathbf{G}\psi$ is true if ψ is a logical consequence of a goal in the goal base, but only if $on(x, z)$ is not believed by the agent (see section 3.1).

PR rule ρ_1 can be applied to an action $move(x, y, z)$ if the condition that x is clear is not satisfied which means that the action cannot be executed. Rule ρ_2 can be applied if z is not clear. The PR rules with head $move(x, y, z)$ construct a plan to create a mental state in which the move action can be executed. Rule ρ_1 for example specifies that if x is not clear because $on(a, x)$, a $move(x, y, z)$ action should be replaced by the plan $move(a, x, Fl); move(x, y, z)$: first clear x, then move x. If in turn the action $move(a, x, Fl)$ cannot be executed because there is some block b on top of x, the plan could be changed to $move(b, a, Fl); move(a, x, Fl); move(x, y, z)$: clear a, clear x and then move x. These PR rules can also be used to adapt an agent's plan if some other block moving agent is present, resulting in a very flexible agent. Suppose the first agent plans to do $move(A, Fl, B)$ and suppose A was clear when he selected this action. Suppose the second agent now moves block C onto A, executing $move(C, D, A)$. The first agent now adapts its plan to $move(C, A, Fl); move(A, Fl, B)$, instead of blindly trying to execute its initial move action. If the action by the second agent would have resulted in a part of the first agent's goal (for instance $on(C, D)$) becoming unsatisfied again, the first agent could apply its goal rule to select an action to achieve this goal once again.

4.2 Example execution

In the initial mental state of the agent $\langle \sigma_0, \gamma_0, E \rangle$, three possible instantiations of goal rule g_1 could be applied: $x = A,\ y = Fl,\ z = B$ or $x = B,\ y = Fl,\ z = C$ or $x = C,\ y = A,\ z = Fl$ (yielding $move(A, Fl, B), move(B, Fl, C)$ or $move(C, A, Fl)$). Suppose the first instantiation is chosen. After application of this goal rule, the plan of the agent becomes equal to the plan in the consequent of the rule, resulting in the following mental state:

$$\begin{array}{lcl} \sigma_1 & = & \{on(A, Fl) \wedge on(B, Fl) \wedge on(C, A) \wedge clear(B) \wedge \\ & & \quad clear(C) \wedge clear(Fl)\}, \\ \gamma_1 & = & \{on(A, B) \wedge on(B, C) \wedge on(C, Fl)\}, \\ \pi_1 & = & move(A, Fl, B). \end{array}$$

The plan cannot be executed because the preconditions of the action are not satisfied in this mental state (block A is not clear). The goal rule cannot be applied because the plan of the agent is not empty. The only applicable rule is the PR rule ρ_1:

$$\begin{array}{lcl} \sigma_2 & = & \{on(A, Fl) \wedge on(B, Fl) \wedge on(C, A) \wedge clear(B) \wedge \\ & & \quad clear(C) \wedge clear(Fl)\}, \\ \gamma_2 & = & \{on(A, B) \wedge on(B, C) \wedge on(C, Fl)\}, \\ \pi_2 & = & move(C, A, Fl); move(A, Fl, B). \end{array}$$

The only option is to execute the first action of the plan:

$$\begin{array}{lcl} \sigma_3 & \models & on(A, Fl) \wedge on(B, Fl) \wedge on(C, Fl) \wedge clear(A) \wedge \\ & & clear(B) \wedge clear(C) \wedge clear(Fl), \\ \gamma_3 & = & \{on(A, B) \wedge on(B, C) \wedge on(C, Fl)\}, \\ \pi_3 & = & move(A, Fl, B). \end{array}$$

In s_3, the action $move(A, Fl, B)$ is executed. In the resulting state, the only (logical consequence of the) goal which is not satisfied, is $on(B, C)$. A plan is constructed to move B onto C: first A is moved onto the floor, then B is moved onto C. The only goal which is not satisfied is $on(A, B)$. The action $move(A, Fl, B)$ is selected using the goal rule en then executed. This results in the following final mental state in which the goal is reached and thus removed from the goal base:

$$\begin{array}{lcl} \sigma_F & \models & on(A, B) \wedge on(B, C) \wedge on(C, Fl) \wedge clear(A) \wedge \\ & & clear(Fl), \\ \gamma_F & = & \emptyset, \\ \pi_F & = & E. \end{array}$$

This example execution shows that the Dribble agent can reach its initial goal. In [14] it is shown that the agent *will always* reach its initial goal, although it will not always take the shortest path. The length of the path depends on which choices are made if multiple instantiations of the goal rule are applicable. As was shown, three instantiations of the goal rule are for instance applicable in the initial mental state.

The example also illustrates how PR rules can be used to modify plans. The guard of the rule describes a situation in which the move action cannot be executed (or would fail). The PR rules are used to replace the move action by a plan that eventually makes it possible to execute the move action that could not be executed at first. In summary, the goal rule in the example is used to select a plan to satisfy a goal. The PR rules are used to modify this plan if the initially selected plan cannot be executed yet.

In [14], it is shown that the tower building agent can be programmed in GOAL as well. The GOAL agent however will not always reach its initial goal, because it can get caught in a cycle. This is due to the fact that GOAL agents do not have a plan component in their mental states. In [14], an example of the tower building agent programmed in 3APL is also given. A 3APL agent however does not have declarative goals. In 3APL, a PR rule with the abstract plan $on(x, z)$ as the head $(on(x, z) \mid \mathbf{B}(on(x, y)) \rightarrow move(x, y, z))$, can be used to select appropriate move actions (replacing the goal rule). Abstract plans however do not have the same properties as declarative goals do in Dribble. An abstract plan is gone once it is "executed" (transformed using a PR rule) whereas a goal is removed only when believed to have been achieved. Suppose the plan $on(C, Fl); on(B, C); on(A, B)$ is the initial plan of a 3APL agent (meant to achieve the goal from the example above). The abstract plan $on(C, Fl)$ will be transformed into $move(C, A, Fl)$ (assuming the initial belief base from the example). Once this move action is executed, it is gone. Suppose a second agent now moves C back onto A again. The 3APL agent will continue executing the remainder of its plan and the goal the programmer had in mind will not be reached. Finally, [14] gives an example of an efficient Dribble program, which always takes the shortest path to the goal. This agent however is programmed to drop and adopt goals "manually" using rules, instead of just using the commitment strategy to modify them (see section 2.3). Therefore, the relation between beliefs and goals established through the commitment strategy is lost. Furthermore, this agent is prone to the same kind of problems that can arise for the 3APL tower building agent.

5. LOGIC

On top of the language Dribble and its semantics, we construct a dynamic logic ($\mathcal{L}_D$) to prove properties of Dribble agents. Mental state formulas $\mathcal{L}_M$ are used to describe properties of the mental state of a Dribble agent. They are an extension of the belief and goal formulas (see section 3.1), with the additional clause if $\pi \in \mathsf{Plan}$ then $\mathbf{Com}(\pi) \in \mathcal{L}_M$.

This formula is thus used to refer to the plan component of a mental state. It is true in a mental state $\langle \sigma, \gamma, \pi' \rangle$ iff $\pi' = \pi$.

5.1 Meta-actions

The language $\mathcal{L}_D$ is a language of *dynamic logic* [6]. Dynamic logic can be used to reason about computer programs. These programs are explicit syntactic constructs in the logic. To be able to discuss the effect of the execution of a program p on the truth of a formula ϕ, the modal construct $\langle p \rangle \phi$ is used. This construct intuitively states that it is possible to execute p and to halt in a state satisfying ϕ. In conventional procedural programming languages, the program p is a transformation function on states: p can be executed in a state, resulting in a new state.
A Dribble agent is a quadruple $\langle \sigma_0, \gamma_0, \Gamma, \Delta \rangle$. A mental state consists of beliefs, goals and a plan. A transformation of one mental state into another can be established by rule application or execution of an action from the plan. This is in contrast with conventional procedural programs, where some fixed program is the transformation function on states. In a dynamic logic for procedural programs, it is this procedural program that can be reasoned about. The question is, what the program is that should be reasoned about in the dynamic logic for Dribble agents.
For Dribble agents, the transformation of one mental state into another depends on whether an applicable rule is applied (and which, if more than one is applicable) or whether an action from the plan is executed. It is these mental state transitions that we want to reason about in the logic, because these transitions define the execution of the agent. We refer to these transitions using so called *meta-actions*. In the dynamic logic for Dribble, a sequence of meta-actions is used as the program p in the formula $\langle p \rangle \phi$.

Definition 12. (meta-actions) The set MetaAction is defined by:

- if $\pi \in$ Plan then $\texttt{commit}(\pi) \in$ MetaAction and $\texttt{uncommit}(\pi) \in$ MetaAction,
- if $g \in \Gamma$ then $\texttt{applyRule}(g) \in$ MetaAction,
- if $\rho \in \Delta$ then $\texttt{applyRule}(\rho) \in$ MetaAction,
- if $b \in$ ExecutableAction then $\texttt{execute}(b) \in$ MetaAction.

The meta-actions $\texttt{commit}(\pi)$ and $\texttt{uncommit}(\pi)$ are used in defining the semantics of the other meta-actions. The meta-actions $\texttt{applyRule}(g)$ and $\texttt{applyRule}(\rho)$ are used to specify that a goal rule g and a PR rule ρ are applied respectively. An executable action b is a basic action or an if-then-else construct (see section 2.2) and the meta-action $\texttt{execute}(b)$ is used to specify that b is executed.

5.2 Semantics

Formulas from the language $\mathcal{L}_D$ are evaluated in a mental state. These formulas consist of mental state formulas μ and of formulas of the form $\langle \alpha \rangle \mu$. The latter formula is true in a mental state s, if it is possible to reach a mental state in which μ holds, through execution of the sequence α of meta-actions and basic actions in the state s. The sequence α is a sequence of meta-actions *and* basic actions because the meaning of the meta-action $\texttt{execute}(a)$ is defined in terms of the meaning of basic action a. The formula $[\alpha]\mu$ is shorthand for $\neg\langle\alpha\rangle\neg\mu$, meaning that all states resulting from execution of α satisfy μ. The mental state reached through execution of α should not be $\mathcal{E}$, the state to which all impossible basic actions and meta-actions lead: from there, no other actions can be performed anymore (properties of $\mathcal{E}$ are not important, because formulas are never evaluated in this undefined state).
Execution of a basic action or meta-action changes the mental state of the agent. This change is specified with the *result function* $\texttt{r}^*(\alpha)\langle \sigma, \gamma, \pi \rangle$. This function yields the mental state resulting from executing α in $\langle \sigma, \gamma, \pi \rangle$. There is only one possible resulting mental state, because basic actions and meta-actions are deterministic. The formula $\langle \sigma, \gamma, \pi \rangle \models \langle \alpha \rangle \mu$ then is true iff there exists a mental state $\langle \sigma', \gamma', \pi' \rangle \neq \mathcal{E} \in \texttt{r}^*(\alpha)\langle \sigma, \gamma, \pi \rangle$ with $\langle \sigma', \gamma', \pi' \rangle \models \mu$. Although actions are deterministic, the function $\texttt{r}^*$ is defined using "exists" to be able to make a distinction between the meaning of $[\alpha]\mu$ and $\langle \alpha \rangle \mu$. Below, the function $\texttt{r}^*$ is defined for single meta-actions and for basic actions (except for the meta-action for execution of the if-then-else construct). For a complete definition of $\texttt{r}^*$, we refer to [14].

Definition 13. (semantics of (un)commit)

$$\begin{array}{lcl} \texttt{r}^*(\texttt{commit}(\pi'))\langle \sigma, \gamma, E \rangle & = & \langle \sigma, \gamma, \pi' \rangle \\ \texttt{r}^*(\texttt{commit}(\pi'))\langle \sigma, \gamma, \pi \rangle & = & \mathcal{E} \textit{ with } \pi \neq E \\ \\ \texttt{r}^*(\texttt{uncommit}(\pi'))\langle \sigma, \gamma, \pi \rangle & = & \langle \sigma, \gamma, E \rangle \textit{ if } \pi = \pi' \\ \texttt{r}^*(\texttt{uncommit}(\pi'))\langle \sigma, \gamma, \pi \rangle & = & \mathcal{E} \textit{ otherwise} \end{array}$$

The actions `commit` and `uncommit` are defined because it makes the definition of the other meta-actions easier. A Dribble agent should not be thought of as having the capability to commit or uncommit to plans. A Dribble agent can only manipulate its plan through rule application or action execution, as was defined in section 3.2.

Definition 14. (semantics of goal rule application) Let $g : \varphi \rightarrow \pi'$ be a goal rule.

$$\begin{array}{l} \texttt{r}^*(\texttt{applyRule}(g))\langle \sigma, \gamma, \pi \rangle = \texttt{r}^*(\texttt{commit}(\pi'))\langle \sigma, \gamma, \pi \rangle \\ \quad \textit{if } \langle \sigma, \gamma, \pi \rangle \models \varphi \\ \texttt{r}^*(\texttt{applyRule}(g))\langle \sigma, \gamma, \pi \rangle = \mathcal{E} \textit{ otherwise} \end{array}$$

The semantics of the $\texttt{applyRule}(g)$ action is defined in terms of the `commit` action. If goal rule g is applied, the agent should adopt the plan in the consequence of g.

Definition 15. (semantics of PR rule application) Let $\rho : \pi_h \mid \beta \rightarrow \pi_b$ be a PR rule.

$$\begin{array}{l} \texttt{r}^*(\texttt{applyRule}(\rho))\langle \sigma, \gamma, \pi \rangle = \\ \quad \texttt{r}^*(\texttt{uncommit}(\pi_h \circ \pi'); \texttt{commit}(\pi_b \circ \pi'))\langle \sigma, \gamma, \pi \rangle \\ \quad\quad \textit{if } \pi = \pi_h \circ \pi' \textit{ and } \langle \sigma, \gamma, \pi \rangle \models \beta \\ \texttt{r}^*(\texttt{applyRule}(\rho))\langle \sigma, \gamma, \pi \rangle = \mathcal{E} \textit{ otherwise} \end{array}$$

The semantics of the $\texttt{applyRule}(\rho)$ action is defined in terms of the `commit` and `uncommit` actions. First the plan of the agent is dropped, after which the plan is empty. Then the plan that has as a prefix the body of the PR rule instead of the head, is adopted through the `commit` action.

Definition 16. (semantics of basic action execution)

$$\begin{array}{l} \texttt{r}^*(\texttt{execute}(a))\langle \sigma, \gamma, \pi \rangle = \\ \quad \texttt{r}^*(a; \texttt{uncommit}(a; \pi'); \texttt{commit}(\pi'))\langle \sigma, \gamma, \pi \rangle \\ \quad\quad \textit{if } \pi = a; \pi' \\ \texttt{r}^*(\texttt{execute}(a))\langle \sigma, \gamma, \pi \rangle = \mathcal{E} \textit{ otherwise} \end{array}$$

The meta-action $\texttt{execute}(a)$ is defined in terms of the basic action a and the `commit` and `uncommit` meta-actions. First a is executed which possibly changes the belief base and goal base of the agent (see definition 17). Then the plan of the agent is updated. There is thus a difference between executing the *meta-action* $\texttt{execute}(a)$ and executing the *basic action* a. The first also results in an update of the plan of the agent, whereas the second one only updates the beliefs and goals.

Definition 17. (semantics of basic actions)
$\mathtt{r}(a)\langle\sigma,\gamma,\pi\rangle =$
$\langle\mathcal{T}(a,\sigma),\gamma',\pi\rangle$ *with* $\gamma' = \gamma \setminus \{\psi \in \gamma \mid \mathcal{T}(a,\sigma) \models \psi\}$
if $\mathcal{T}(a,\sigma)$ is defined
$\mathtt{r}(a)\langle\sigma,\gamma,\pi\rangle = \mathcal{E}$ *otherwise*

If basic actions are executed, the beliefs and goals of the agent are updated. As in section 3.2, the belief update is formally represented by a partial transformation function $\mathcal{T}$. If this function is not defined, the basic action cannot be executed.

5.3 Characteristics of actions

We now show how characteristics of meta-actions can be specified in the logic. These are characteristics of *realizability* and *results* of meta-actions. Realizability specifies when an action can be executed and results specify the properties of the mental state resulting from executing a meta-action. In the following, ρ is a PR rule $\pi_h \mid \beta \rightarrow \pi_b$ and g is a goal rule $\varphi \rightarrow \pi$.

THEOREM 1. *(realizability of meta actions)*

1	$\models \mathbf{Com}(E)$	$\leftrightarrow$	$\langle\texttt{commit}(\pi)\rangle\top$
2	$\models \mathbf{Com}(\pi)$	$\leftrightarrow$	$\langle\texttt{uncommit}(\pi)\rangle\top$
3	$\models \mathbf{Com}(E) \wedge \varphi$	$\leftrightarrow$	$\langle\texttt{applyRule}(g)\rangle\top$
4	$\models \mathbf{Com}(\pi_h \circ \pi) \wedge \beta$	$\rightarrow$	$\langle\texttt{applyRule}(\rho)\rangle\top$
5	$\models \mathbf{Com}(a;\pi) \wedge \langle a\rangle\top$	$\rightarrow$	$\langle\texttt{execute}(a)\rangle\top$

All actions require a commitment to some plan to be executed successfully. The first three properties specify an equivalence whereas the last two specify an implication. The last two properties do not specify an equivalence because no conclusion can be drawn about the plan of the agent, assuming that one of these meta-actions is executable. Assuming that the meta-action $\texttt{execute}(a)$ is executable for example, the agent could have a commitment to $a;\pi_1$, to $a;\pi_2$ and so on and so forth. The third and fourth properties do not only have a requirement on the plan, but also on the beliefs and/or goals and the beliefs respectively. The fifth property has the requirement that basic action a should be realizable. Only then can the meta-action $\texttt{execute}(a)$ be executed successfully.

THEOREM 2. *(results of meta actions)*

1	$\models [\texttt{commit}(\pi)]\mathbf{Com}(\pi)$
2	$\models [\texttt{uncommit}(\pi)]\mathbf{Com}(E)$
3	$\models [\texttt{applyRule}(g)]\mathbf{Com}(\pi)$
4	$\models \mathbf{Com}(\pi_h \circ \pi) \rightarrow [\texttt{applyRule}(\rho)]\mathbf{Com}(\pi_b \circ \pi)$
5	$\models \mathbf{Com}(a;\pi) \wedge [a]\beta \rightarrow [\texttt{execute}(a)]\Big(\mathbf{Com}(\pi) \wedge \beta\Big)$

The first, second and third properties state that if the specified atomic actions can be executed, they have the specified commitments as a result. The fourth action needs a condition on the plan of the agent ($\mathbf{Com}(\pi_h \circ \pi)$). Without this condition, no statement can be made about the resulting plan. The same holds for the fifth meta-action. The fifth property specifies the result of the meta-action $\texttt{execute}(a)$ if the result on the belief base of executing the basic action a is known.
Actions are deterministic. A consequence of this is, that the formula $\langle\alpha\rangle\mu$ is stronger than the formula $[\alpha]\mu$. The diamond formula expresses that μ holds in the state resulting from executing α. The box formula only states that *if* α can be executed, μ will hold in the resulting state. The following equivalence holds: $\langle\alpha\rangle\mu \leftrightarrow (\langle\alpha\rangle\top \wedge [\alpha]\mu)$. The conditions on realizability $\langle\alpha\rangle\top$ and results $[\alpha]\mu$ of actions can thus be combined, deriving the formula $\langle\alpha\rangle\mu$.

5.4 Example

An example of a property which can be specified in the logic for the Dribble agent of section 4, is the following: it is possible to realize the goal from the initial beliefs (where the goal, initial beliefs and rules are as specified in that section). We will need the following instantiations of the goal rule of that section.

Definition 18. (goal rule instantiations)

$g_1 : \mathbf{B}(on(C,A))$	$\wedge$	$\mathbf{G}(on(C,Fl))$	$\rightarrow$	$move(C,A,Fl)$
$g_2 : \mathbf{B}(on(B,Fl))$	$\wedge$	$\mathbf{G}(on(B,C))$	$\rightarrow$	$move(B,Fl,C)$
$g_3 : \mathbf{B}(on(A,Fl))$	$\wedge$	$\mathbf{G}(on(A,B))$	$\rightarrow$	$move(A,Fl,B)$

The property is expressed in the following formula:

$$
\begin{aligned}
s_0 \models\ & \langle \texttt{applyRule}(g_1); \texttt{execute}(move(C,A,Fl)); \\
& \texttt{applyRule}(g_2); \texttt{execute}(move(B,Fl,C)); \\
& \texttt{applyRule}(g_3); \texttt{execute}(move(A,Fl,B))\rangle \\
& \mathbf{B}(on(A,B) \wedge on(B,C) \wedge on(C,Fl)).
\end{aligned}
$$

The formula specifies that in the initial mental state s_0, the following is true: the specified sequence of rule applications and action executions results in a state in which the agent believes that the goal tower is built. It can be shown that this property indeed holds for the example agent, using the definitions of $\mathtt{r}^*$ and of the meta-actions (see [14]).

6. CORRESPONDENCE BETWEEN LOGIC AND OPERATIONAL SEMANTICS

In section 3.3, the meaning of a Dribble agent was defined as the set of computation runs, starting in the initial mental state of the agent. A computation run is a series of mental state transitions, where each transition is a transition in the transition system for the Dribble agent. In section 5, a dynamic logic was sketched in which one can reason about actions defined in that logic. These actions transform the mental state of the agent, expressed with the function $\mathtt{r}^*$. The idea is, that mental state transitions defined by actions in the logic, correspond to the mental state transitions defined by the transition system. If this were the case, properties derived for actions in the logic are actually properties of the Dribble agent, since the meaning of a Dribble agent is defined in terms of these transitions in the transition system. The correspondence is however not defined for all actions in the logic. This is because these actions do not all correspond to a mental state transition in the transition system. The meta-actions `commit` and `uncommit` for instance were only defined because it made the definition of other meta-actions

easier (see section 5.2). They do not have a counterpart in the transition system. The actions that do have a counterpart in the transition system are $\texttt{applyRule}(g)$, $\texttt{applyRule}(\rho)$ and $\texttt{execute}(b)$, which are called *transition actions*. In the logic, one can reason about sequences of these transition actions θ $(= t_1; \ldots; t_n)$. Note that the fact that we exclude the (un)commit actions from θ, does not mean that $\texttt{r}^*(\theta)$ is undefined. The mental state resulting from execution of θ can be defined in terms of (un)commit actions without including them in the sequence θ initially.

What we want to prove is, that the mental state resulting from execution of some sequence θ, corresponds to the mental state resulting from a sequence of transitions in the transition system. This sequence of transitions is however not just any sequence. The elements of the sequence of transitions have a one to one correspondence with the elements of the sequence of transition actions θ: each transition $s_i \rightarrow_{t_i} s_{i+1}$ corresponds to the transition action t_i in the sequence θ. We express this correspondence using the concept of a *computation run of* θ $(\texttt{CR}_\theta(s_0))$. A computation run of θ is a finite sequence $s_0, t_1, s_1, \ldots, t_n, s_n$ where s_i are mental states and $\forall_{i>0}: \; s_{i-1} \rightarrow_{t_i} s_i$ is a transition in the transition system and $\theta = t_1; \ldots; t_n$.

The correspondence between logic and operational semantics is expressed in the following theorem, which is proven in [14]. The function last yields the last state of the finite computation run.

THEOREM 3. *(correspondence between operational semantics and actions in the logic) For* $\theta \in$ TransitionActionSeq *and* $s_0, s_n \in \Sigma$*, the following holds for a Dribble agent with goal rules* Γ *and PR rules* Δ *:*

$$\texttt{r}^*(\theta) \; s_0 = s_n \Leftrightarrow \texttt{last}(\texttt{CR}_\theta(s_0)) = s_n$$

7. CONCLUSION AND FUTURE WORK

The language Dribble combines features from the languages GOAL and 3APL. It incorporates beliefs and (declarative) goals, as well as planning features (or procedural goals). It thus implements key concepts of logics like BDI and KARO. It is an enhancement of the language 3APL which only uses procedural goals and of the language GOAL, using only declarative goals.

An obvious limitation of Dribble is that it is a propositional language. Extending the language with first order features is therefore an important issue for future research. Furthermore, Dribble uses goals for plan selection only. Another extension could therefore concern for instance the addition of rules to reason with goals. A third possibility to extend Dribble concerns the commitment strategy. In the current version, the agent drops a goal only if it believes it has been achieved. An interesting issue could be to investigate whether the strategy can be extended in such a way that the agent will drop its goal also if it believes it is unachievable. Finally, the goal for which an agent's plan was selected, should be recorded. The agent could then be programmed to drop its plan if the goal for which it was selected, is achieved. Extending Dribble with these features is important to make it suitable for practical use. We nevertheless believe that the language presented in this paper is an important step towards bridging the gap between theory and practice.

8. REFERENCES

[1] M. Dastani, F. S. de Boer, F. Dignum, and J.-J. Ch. Meyer. Programming agent deliberation. In *Proceedings of AAMAS*, Melbourne, 2003.

[2] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. A formal embedding of AgentSpeak(L) in 3APL. In G. Antoniou and J. Slaney, editors, *Advanced Topics in Artificial Intelligence*, pages 155–166. Springer, LNAI 1502, 1998.

[3] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in 3APL. *Int. J. of Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[4] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming with declarative goals. In *Intelligent Agents VI - Proceedings of the 7th International Workshop on Agent Theories, Architectures, and Languages (ATAL'2000)*, Lecture Notes in AI. Springer, Berlin, 2001.

[5] K. V. Hindriks, Y. Lespirance, and H. Levesque. A formal embedding of ConGolog in 3APL. In *Proceedings of the 14th European Conference on Artificial Intelligence*, pages 558–562, 2002.

[6] D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 789–840. Elsevier, Amsterdam, 1990.

[7] G. Plotkin. A structural approach to operational semantics. Technical report, Aarhus University, Computer Science Department, 1981.

[8] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. van der Velde and J. Perram, editors, *Agents Breaking Away (LNAI 1038)*, pages 42–55. Springer-Verlag, 1996.

[9] A. S. Rao and M. Georgeff. BDI Agents: from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319, San Francisco, CA, June 1995.

[10] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann, 1991.

[11] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[12] W. van der Hoek, B. van Linder, and J.-J. Ch. Meyer. An integrated modal approach to rational agents. In M. Wooldridge and A. S. Rao, editors, *Foundations of Rational Agency*, Applied Logic Series 14, pages 133–168. Kluwer, Dordrecht, 1998.

[13] B. van Linder, W. van der Hoek, and J.-J. Ch. Meyer. Formalizing abilities and opportunities of agents. *Fundamenta Informaticae*, 34(1,2):53–101, 1998.

[14] M. B. van Riemsdijk. Agent programming in Dribble: from beliefs to goals with plans. Master's thesis, Utrecht University, 2002.