Goal Selection Strategies for Rational Agents

Nick A.M. Tinnemeier, Mehdi Dastani, and John-Jules Ch. Meyer

Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

Abstract. In agent theory and agent programming, goals constitute the motivational attitude of rational agents and form the key concept in explaining and generating their pro-active behavior. Pursuing multiple goals simultaneously might pose problems for agents as the plans for achieving them may conflict. We argue that a BDI-based agent programming language should provide constructs to allow an agent programmer to implement agents that: 1) do not pursue goals with conflicting plans simultaneously, and 2) can choose from goals with conflicting plans. This paper presents an explicit and generic mechanism to process incompatible goals, i.e., goals with conflicting plans. The proposed mechanism can be integrated in existing BDI-based agent programming languages. We discuss different strategies to process incompatible goals based on a given conflict relation and show some properties and relations between these strategies.

1 Introduction

To facilitate the implementation of cognitive agents, BDI-based agent programming languages provide constructs to implement agent concepts such as beliefs, goals and plans. Examples of these programming languages are Jadex [1], Jack [2], Jason [3], 3APL [4], IMPACT [5], CLAIM [6] and 2APL [7]. In these agent programming languages, belief constructs can be used to implement the (incomplete) information the agent has about its world, whereas goal constructs can be used to implement the states the agent desires to achieve. In agent theory and agent programming, goals constitute the motivational attitude of rational agents and form the key concept in explaining and generating their pro-active behavior [8–10]. In pursuing its goals an agent uses (partial) plans which specify the actions that should be taken to achieve its goals. In general, most BDI-based agent programming languages allow an agent to have multiple goals at the same time. When an agent has more than one goal, different strategies are possible for adopting plans to achieve these goals. A strategy that is commonly used in many agent programming languages, for example in both 3APL[4] and 2APL [7], is to adopt a plan for each goal and to execute all generated plans at the same time (in an interleaving mode).

Pursuing multiple goals simultaneously might be beneficial for an agent, it also poses problems. Goals might be incompatible with each other in the sense

that the plan for reaching one goal possibly conflicts with the plans for other goals. Consider for example a household agent with the capability of cleaning rooms. Suppose that the agent has two goals: to have cleaned room one and five. Although it is possible for the agent to achieve one goal after the other, trying to achieve them simultaneously by first making a step in the direction of one room and then in the direction of the other would clearly be irrational. So, the goals that an agent has committed to by adopting plans might pose constraints for the adoption of plans to pursue other goals. Furthermore, confronted with different incompatible goals, an agent should still be able to choose among goals. Therefore, we argue that a BDI-based agent programming language should provide constructs to allow an agent programmer to implement agents that: 1) do not pursue incompatible goals simultaneously, and 2) can choose from possibly incompatible goals. Most agent programming languages, however, lack constructions that sufficiently deal with these issues in an explicit way. It should be noted that such constructs are different than a goal (event) selection function as proposed, for example, by Jason [3]. These selection functions are too generic and are not devised to process incompatible goals. In fact, our proposal can be considered as a specific instantiation of such a function.

One might argue that it is the responsibility of the agent programmer to implement its agents in such a way that its goal base will never contain incompatible goals. For example, the programmer should ensure that a goal is added to its goal base after the existing plans for incompatible goals are fully executed and the goals are either achieved or dropped. However, we believe that adding a goal to the goal base should not depend on the existence of incompatible goals, as the goals of an agent can in principle be incompatible or even inconsistent. Moreover, we believe that an agent programmer may not know at design time which goals it will adopt during its execution such that it becomes a cumbersome task, if not impossible, to write such an agent program. In our opinion, a generic mechanism to process incompatible goals facilitates the implementation of pro-active agents and eases the task of agent programmers. A different solution to avoid that an agent pursues incompatible goals is to use the notion of atomic plans as introduced in 2APL. Atomicity of a plan ensures that the plan is executed at once without interleaving its actions with the actions of other plans. This mechanism can be used to avoid the interleaved execution of the plans for incompatible goals, i.e., to avoid simultaneous pursuit of incompatible goals. This solution is, however, too restrictive as it does not allow the actions of an atomic plan to be interleaved with the plans of *compatible* goals.

In this paper, we propose an explicit and generic mechanism to process incompatible goals. In order to illustrate that the proposed mechanism can be integrated in arbitrary BDI-based agent programming languages, we present the proposed mechanism in the context of a simple agent programming language that can be extended to existing BDI-based agent programming languages. According to this proposal, an agent programmer should specify a conflict relation between only those *sub*-goals for which a planning rule is specified. It should be noted that in most BDI-based agent programming languages, all (sub-)goals

for which planning rules are specified, are known at design time. It should also be noted that these planning rules could be applied to adopt plans for arbitrary goals not known at design time. We discuss different strategies to process incompatible goals based on a given conflict relation and show some properties and relations between these strategies. In particular, we present in section 2 a simple generic BDI-based agent programming language by specifying its syntax and operational semantics. Then, in section 3 we extend the programming language with a goal conflict relation and discuss different strategies to process incompatible goals and show their properties and relations. We conclude the paper in section 4 with some remarks, related works and future research.

2 An agent programming language

In this section we provide the syntax and semantics of a logic-based agent programming language. The language provided here is based on 2APL [7], but does not reflect its complete syntax and semantics. Instead, we provide a simplified version that is self-contained and can be used to illustrate the different notions of a goal selection strategy. In contrast to 2APL and many logic-based agent programming languages in which the beliefs and goals of the agent are modelled in a subset of first-order logic, the programming language presented here uses a subset of propositional logic. Furthermore, 2APL provides external actions by which the agent can change its environment, communicative actions to communicate with other agents, and rules to react to external events. All of these constructs are left out in the language presented here, because they are not needed to illustrate the idea of goal selection strategies. In the next section this simplified language will be extended with some goal selection strategies.

2.1 Syntax

An agent has beliefs about its environment, and goals to denote the desirable situation it wants to realize. As mentioned earlier, these are modelled in a subset of propositional logic.

Definition 1 (belief and goal language). *Let the set $\mathcal{P}$ be the set of atomic propositions. The belief language $\mathcal{L}_B$ with typical element β, and goal language $\mathcal{L}_G$ with typical element κ are then defined as:*

- *if $\varphi \in \mathcal{P}$ then $\varphi, \neg\varphi \in \mathcal{L}_B$*
- *if $\varphi \in (\mathcal{P} \setminus \{\top, \bot\})$ then $\varphi, \neg\varphi \in \mathcal{L}_G$*
- *if $\kappa, \kappa' \in \mathcal{L}_G$ then $\kappa \wedge \kappa' \in \mathcal{L}_G$*

The symbol $\models$ will be used to denote the standard entailment relation for propositional logic.

The beliefs of the agent can thus be represented by literals, i.e. positive and negative atomic propositions. A belief that the agent is in room three, for

instance, can be represented as `in_room_3`. The goals of the agent can be represented by a conjunction of literals, for instance, `cleaned1` and `cleaned2` to denote the goals of having cleaned room one and two.

To reach its goals, an agent needs to act. A plan describes a sequence of actions an agent should perform in order to reach its goals. For the sake of simplicity and to focus on goal selection strategies we assume only a set of basic actions by which the agent can modify its beliefs, and an action by which the agent can adopt new goals.

Definition 2 (plan language). *Let $\mathcal{A}ct$ with typical element a be the set of basic actions an agent can perform. The set of plans* `Plan` *with typical element π is then defined as:*

- $\mathcal{A}ct \subseteq$ `Plan`
- *if* $\kappa \in \mathcal{L}_G$ *then* `adopt`$(\kappa) \in$ `Plan`
- *if* $\pi_1, \pi_2 \in$ `Plan` *then* $\pi_1; \pi_2 \in$ `Plan`

In the following we will use ϵ to denote the empty plan and identify $\epsilon; \pi$ and $\pi; \epsilon$ with π. Furthermore, we assume that every plan is ended by ϵ.

An agent can possibly know of more than one plan to pursue a single goal. Which plan is the best depends on the current situation. To choose and generate an appropriate plan, the agent uses so-called planning goal rules. These rules are of the form $\kappa \leftarrow \beta \mid \pi$. The informal meaning of such a rule is that the agent can use a plan π to reach a goal κ in case the agent believes β.

Definition 3 (planning goal rules). *The set of goal planning rules $\mathcal{R}_{PG}$ is defined as:*

$$\mathcal{R}_{PG} = \{(\kappa \leftarrow \beta \mid \pi) : \kappa \in \mathcal{L}_G \text{ and } \beta \in \mathcal{L}_B \text{ and } \pi \in \texttt{Plan}\}$$

2.2 Semantics

In this section we define the operational semantics of the agent programming language as defined in the previous section in terms of a transition system. A transition system is a set of derivation rules for deriving transitions for this language. A transition is a transformation of one configuration C into another configuration C', denoted by $C \longrightarrow C'$. Each transition corresponds to a single computation step for the presented (agent) programming language. A configuration represents the state of an agent at each point during computation.

Definition 4 (agent configuration). *Let $\Sigma = \{\sigma : \sigma \subseteq \mathcal{L}_B \text{ and } \sigma \not\models \bot\}$ be the set of consistent belief sets, and let $\Gamma = \{\kappa \in \mathcal{L}_G : \kappa \not\models \bot\}$ be the set of goals. An agent configuration is then defined as a tuple $\langle \sigma, \gamma, \Pi, \texttt{PG} \rangle$ where $\sigma \in \Sigma$ is the belief base, $\gamma \subseteq \Gamma$ is the goal base, $\Pi \subseteq (\mathcal{L}_G \times$ `Plan`$)$ are the plans, and* `PG` $\subseteq \mathcal{R}_{PG}$ *are the planning goal rules.*

The plan base of the agent is a set of pairs (κ, π), where κ denotes the state of affairs that is supposed to be reached by the sequence of actions denoted by π. We use κ to keep track of the goals the agent is working on.

Note that in contrast to the belief base, the individual goals in the goal base are consistent, but different goals can be inconsistent. An agent can thus have as goal `in_room_3` while it also has a goal $\neg$`in_room_3`. We say that an agent has a goal κ when κ is derivable from the goal base of that agent. As the goal base can be inconsistent we cannot use the same entailment relation as we use for the belief base. Instead, we define a goal entailment relation to be used for the goal base (cf. [11]). As it would be irrational for an agent to have goals that are already believed to be achieved, the belief base of the agent is also used for this goal entailment relation.

Definition 5 (goal entailment). *Let $\gamma \subseteq \Gamma$ be a goal base, and let $\sigma \in \Sigma$ be a belief base. The goal entailment relation $\models_g$ is then defined in the following way:*

$$(\gamma, \sigma) \models_g \kappa \Leftrightarrow (\exists \gamma_i \in \gamma : \gamma_i \models \kappa) \text{ and } \sigma \not\models \kappa$$

An agent with belief base σ and goal base γ is thus said to have a goal κ if and only if κ is derivable from some of its goals and is not entailed by the belief base.

The agent can update its belief base by performing basic actions. For this purpose we use a function $\texttt{update} : \mathcal{A}ct \times \Sigma \rightarrow \Sigma$ that takes as arguments a basic action and a belief base, and evaluates to a new belief base as a consequence of executing the basic action. The transition rule defined below defines the semantics of executing a basic action, which can be executed in case the goal for which the plan is generated is still a goal of the agent (condition $(\gamma, \sigma) \models_g \kappa$). A goal of the agent is removed from its goal base if the goal is believed to be reached after having executed the belief update operation (clause 3). Moreover, it would be irrational for an agent to execute a plan for a goal already believed to be reached. Therefore, the plans that were generated for this goal are removed from the plan base (clause 2). In defining the transition rules below an agent configuration $C = \langle \sigma, \gamma, \Pi, \texttt{PG} \rangle$ is assumed. The set of planning rules `PG` will be omitted whenever possible, since this component does not change during the agent's execution.

R1 (belief update) *Let $C = \langle \sigma, \gamma, \Pi, \texttt{PG} \rangle$ be an agent configuration, and let $a \in \mathcal{A}ct$ and $(\kappa, a; \pi) \in \Pi$.*

$$\frac{\texttt{update}(a, \sigma) = \sigma' \text{ and } (\gamma, \sigma) \models_g \kappa}{\langle \sigma, \gamma, \Pi \rangle \longrightarrow \langle \sigma', \gamma', \Pi'' \rangle}$$

where
1) $\Pi' = (\Pi \setminus \{(\kappa, a; \pi)\}) \cup \{(\kappa, \pi)\}$
2) $\Pi'' = \Pi' \setminus \{(\kappa', \pi') \in \Pi' : (\gamma, \sigma') \not\models_g \kappa'\}$
3) $\gamma' = \gamma \setminus \{\gamma_i \in \gamma : \sigma' \models \gamma_i\})$

Note that under the interpretation of the goal entailment relation a goal `in_room_1` $\wedge$ `battery_loaded` differs from having two separate goals `in_room_1` and `battery_loaded` in the goal base. The first goal is only achieved once the agent believes `in_room_1` $\wedge$ `battery_loaded`, while the single goal `in_room_1` is achieved if it believes `in_room_1` even when it does not believe `battery_loaded`.

Agents can adopt new goals by performing an `adopt` action. The goal is added to the goal base only if the goal is not already believed to be achieved. The following two transition rules capture a goal adoption. The first rule captures the case in which the goal is not already believed to be achieved, whereas the second rule captures the case in which the goal is already believed to be achieved. In the latter case the plan execution proceeds without any changes in the agent's belief and goal bases.

R2 (goal adoption 1) *Let* $C = \langle \sigma, \gamma, \Pi, \texttt{PG} \rangle$ *be an agent configuration.*

$$\frac{(\kappa, \texttt{adopt}(\kappa'); \pi) \in \Pi \text{ and } (\gamma, \sigma) \models_g \kappa \text{ and } \sigma \not\models \kappa'}{\langle \sigma, \gamma, \Pi \rangle \longrightarrow \langle \sigma, \gamma \cup \{\kappa'\}, \Pi' \rangle}$$

where $\Pi' = (\Pi \setminus \{(\kappa, \texttt{adopt}(\kappa'); \pi)\}) \cup \{(\kappa, \pi)\}$

R3 (goal adoption 2) *Let* $C = \langle \sigma, \gamma, \Pi, \texttt{PG} \rangle$ *be an agent configuration.*

$$\frac{(\kappa, \texttt{adopt}(\kappa'); \pi) \in \Pi \text{ and } (\gamma, \sigma) \models_g \kappa \text{ and } \sigma \models \kappa'}{\langle \sigma, \gamma, \Pi \rangle \longrightarrow \langle \sigma, \gamma, \Pi' \rangle}$$

where $\Pi' = (\Pi \setminus \{(\kappa, \texttt{adopt}(\kappa'); \pi)\}) \cup \{(\kappa, \pi)\}$

When an agent has executed all the actions of a plan, this plan is removed from the plan base. Removing a plan from the plan base does not affect the goal base of the agent. When the plan failed in establishing the desired state, the goal remains in the goal base of the agent. The next transition rule is for removing empty plans from the plan base.

R4 (empty plan) *Let* $C = \langle \sigma, \gamma, \Pi, \texttt{PG} \rangle$ *be an agent configuration.*

$$\frac{(\kappa, \epsilon) \in \Pi}{\langle \sigma, \gamma, \Pi \rangle \longrightarrow \langle \sigma, \gamma, \Pi \setminus \{(\kappa, \epsilon)\} \rangle}$$

As already mentioned, an agent uses planning goal rules for generating plans by which it hopes to reach its goals. An agent can apply a goal planning rule $\kappa \leftarrow \beta \mid \pi$ if κ is a goal of the agent, β is derivable from the agent's belief base, and the agent is not already working on a plan for κ. In defining the transition rule for plan generation we first define the set of applicable planning rules with respect to an agent configuration.

Definition 6 (applicable rules). *Let $C = \langle \sigma, \gamma, \Pi, \texttt{PG} \rangle$ be an agent configuration, and let $\kappa \in \mathcal{L}_G$. The set of applicable planning goal rules for goal κ w.r.t. configuration C is then defined as:*

$$appl(\kappa, C) =$$

$$\{\kappa \leftarrow \beta \mid \pi \in \texttt{PG} : (\gamma, \sigma) \models_g \kappa \text{ and } \sigma \models \beta \text{ and } \neg\exists \pi' \in \texttt{Plan} : (\kappa, \pi') \in \Pi\}$$

When a goal planning rule is applicable the plan will be added to the agent's plan base. The following rule captures this situation.

R5 (plan generation) *Let $C = \langle \sigma, \gamma, \Pi, \texttt{PG} \rangle$ be an agent configuration.*

$$\frac{(\kappa \leftarrow \beta \mid \pi) \in appl(\kappa, C)}{\langle \sigma, \gamma, \Pi \rangle \longrightarrow \langle \sigma, \gamma, \Pi' \rangle}$$

where $\Pi' = \Pi \cup \{(\kappa, \pi)\}$

In order to show some properties of the behavior of an agent, we define the notion of an agent execution. Given a transition system consisting of a set of transition rules, an execution of an agent is a sequence of configurations that can be generated by applying transition rules to the initial configuration of that agent. An agent execution thus shows a possible behavior of the agent. All possible executions for an initial configuration show the complete behavior of an agent.

Definition 7 (agent execution). *An execution of an agent in transition system $\mathcal{T}$ is a (possibly infinite) sequence of agent configurations $\langle C_0, C_1, \ldots \rangle$ such that for each for $i \in \mathbb{N}$, $C_i \longrightarrow C_{i+1}$ can be derived from $\mathcal{T}$. We use the term initial configuration to refer to C_0.*

Recall that $(\kappa, \pi) \in \Pi$ means that a plan has been generated to achieve a state denoted by κ. We assume that in the initial configuration the associated κ to each plan in the plan base is in fact a goal of the agent. Under this assumption we can show that $R1, \ldots, R5$ ensures that the associated κ to each plan in the plan base in all derived configurations is in fact a goal of the agent. In other words, the agent will never adopt a plan for which the corresponding κ is not a goal of the agent.

Proposition 1. *Let $\langle C_0, C_1, \ldots \rangle$ be an agent execution in transition system $\mathcal{T}$, where $C_i = \langle \sigma^i, \gamma^i, \Pi^i, \texttt{PG} \rangle$, and let $\mathcal{T}$ consist of the rules $R1, \ldots, R5$. Given that $\forall (\kappa, \pi) \in \Pi^0 : (\gamma^0, \sigma^0) \models_g \kappa$, then $\forall i \in \mathbb{N}. \forall (\kappa, \pi) \in \Pi^i : (\gamma^i, \sigma^i) \models_g \kappa$*

Proof. By induction on the depth of the execution. We have $\forall (\kappa, \pi) \in \Pi^0 : (\gamma^0, \sigma^0) \models_g \kappa$ by assumption. Now assume that $\forall (\kappa, \pi) \in \Pi^k : (\gamma^k, \sigma^k) \models_g \kappa$ for arbitrary $k \geq 0$. Now we have to prove that after application of one of $R1, \ldots, R5$ it holds that $\forall (\kappa, \pi) \in \Pi^{k+1} : (\gamma^{k+1}, \sigma^{k+1}) \models_g \kappa$. For $R2$, $R3$ and $R4$ this

is trivial as they do not change the belief and goal bases. Assume that a plan $(\kappa, \pi) \in \Pi^{k+1}$ *is adopted by application of rule R5. From definition 6 and that* $\gamma^k = \gamma^{k+1}$ *after application of R5 it follows that* $(\gamma^{k+1}, \sigma^{k+1}) \models_g \kappa$*. Now assume that a goal is removed from the goal base by application of rule R1. From the fact that if a goal is removed, also all the plans that are associated to this goal are removed we conclude that* $\forall (\kappa, \pi) \in \Pi^{k+1} : (\gamma^{k+1}, \sigma^{k+1}) \models_g \kappa$ *still holds.* □

3 Goal Selection Strategies

The previous section defined a simplified version of an agent programming language. In this section we consider several possible goal selection strategies for this agent programming language. Central to the notion of a goal selection strategy is that we relate those goals that cannot be pursued simultaneously. For this purpose we extend the previously defined agent configuration with a binary relation $\mathcal{R}$ on the set of goals. We call such an extended agent configuration a goal strategy agent.

Definition 8 (goal strategy agent). *Let* $\langle \sigma, \gamma, \Pi, \mathtt{PG} \rangle$ *be an agent configuration. A goal strategy agent is a tuple* $\langle \sigma, \gamma, \Pi, \mathtt{PG}, \mathcal{R} \rangle$ *where* $\mathcal{R} \subseteq (\mathcal{L}_G \times \mathcal{L}_G)$ *is a goal selection strategy.*

The main idea of $\mathcal{R}$ is thus that it specifies which goals are incompatible with each other. To work on two goals that might hinder the achievement of one another would be irrational. Consequently, we desire that if two goals are incompatible the agent should not be working on plans for these goals at the same time.

Definition 9 (non-conflicting plan base). *Let* $C = \langle \sigma, \gamma, \Pi, \mathtt{PG}, \mathcal{R} \rangle$ *be a goal strategy agent. The plan base in* C *is* $\mathcal{R}$*-non-conflicting iff:*

$$\forall (\kappa, \pi), (\kappa', \pi') \in \Pi : (\kappa, \kappa') \notin \mathcal{R} \text{ and } (\kappa', \kappa) \notin \mathcal{R}$$

In the sequel we consider several notions of a goal selection strategy by introducing the relations $\mathcal{R}^{<>}$ and $\mathcal{R}^{\prec}$ as concrete instances of $\mathcal{R}$. We study some of the possible semantics of these relations by providing alternative definitions of the plan generation rule R5.

3.1 Incompatibility of goals

Goals the agent has already committed to by having adopted a plan constrain the possibility for the pursuit of other goals. A rational agent is expected to refrain from adopting a plan that hinders the achievement of the goals the agent is currently committed to. In this subsection we define the *incompatibility* relation $\mathcal{R}^{<>}$ to relate goals that cannot be pursued at the same time. We adapt the previously defined plan generation rule R5 to ensure that the agent generates its plans in such a way that the plan base remains non-conflicting.

Definition 10 (goal incompatibility relation). *A goal incompatibility relation* $\mathcal{R}^{<>} \subseteq (\mathcal{L}_G \times \mathcal{L}_G)$ *is a set of pairs of goals such that:*

- $(\kappa, \kappa') \in \mathcal{R}^{<>} \leftrightarrow (\kappa', \kappa) \in \mathcal{R}^{<>}$
- $(\kappa, \kappa') \in \mathcal{R}^{<>} \rightarrow \kappa' \neq \kappa$

Intuitively, when $(\kappa, \kappa') \in \mathcal{R}^{<>}$ this means that the goal κ cannot be pursued in parallel with the goal κ'. Note that the incompatibility relation is symmetric and anti-reflexive, meaning that two distinct goals are always incompatible with each other and no goal can be incompatible with itself. The next transition rule redefines rule R5 for plan generation, now taking the incompatibility of goals into account.

R5.1 (incompatibility) *Let* $C = \langle \sigma, \gamma, \Pi, \texttt{PG}, \mathcal{R}^{<>} \rangle$ *be a goal strategy agent with* $\mathcal{R}^{<>}$ *being an incompatibility relation*

$$\frac{(\kappa \leftarrow \beta \mid \pi) \in appl(\kappa, C) \text{ and } \forall \kappa' \in \mathcal{L}_G : (\kappa', \pi') \in \Pi \rightarrow (\kappa, \kappa') \notin \mathcal{R}^{<>}}{\langle \sigma, \gamma, \Pi \rangle \longrightarrow \langle \sigma, \gamma, \Pi' \rangle}$$

where $\Pi' = \Pi \cup \{(\kappa, \pi)\}$

In words, a plan can be generated for κ if there is an applicable planning rule for κ, and none of the current plans the agent is working on are for a goal that is incompatible with κ. Returning to the example of section 1, in this new transition system we can prevent the household agent from trying to clean the rooms one and five at the same time by defining the goals `clean1` and `clean5` as incompatible with each other, i.e. $(\texttt{clean1}, \texttt{clean5}), (\texttt{clean5}, \texttt{clean1}) \in \mathcal{R}^{<>}$. When the household agent has for example adopted a plan for cleaning room one, it will not adopt a plan for cleaning room five as long as it is still working on cleaning room one. Note that in case the adopted plan finished, but failed to clean room one, the agent can either try again to clean room one or it can start working on cleaning room five instead.

Similar to proposition 1 we show for rules $R1, \ldots, R4, R5.1$ that the agent will never adopt a plan for which the corresponding κ is not a goal of the agent when in the initial configuration the associated κ to each plan in the plan base is in fact a goal of the agent. Although not needed for proving that the plan base of the agent remains non-conflicting during its execution, we provide this property for the sake of completeness.

Corollary 1. *Let* $\langle C_0, C_1, \ldots \rangle$ *be an agent execution in transition system* $\mathcal{T}$*, where* $C_i = \langle \sigma^i, \gamma^i, \Pi^i, \texttt{PG}, \mathcal{R}^{<>} \rangle$*, and* $\mathcal{T}$ *consists of the rules* $R1, \ldots, R4$ *and* $R5.1$*. Given that* $\forall (\kappa, \pi) \in \Pi^0 : (\gamma^0, \sigma^0) \models_g \kappa$*, then* $\forall i \in \mathbb{N}. \forall (\kappa, \pi) \in \Pi^i : (\gamma^i, \sigma^i) \models_g \kappa$

Proof. This follows from proposition 1 and the fact that $R5.1$ *is a more restrictive version of* $R5$*.* □

The bottom line of $\mathcal{R}^{<>}$ is to ensure that if the agent started with a non-conflicting plan base, the plan base of the agent remains non-conflicting. Given that the agent starts with a non-conflicting plan base, we show that the plan base stays non-conflicting for all executions in the transition system consisting of the rules $R1, \ldots, R4, R5.1$.

Proposition 2. *Let $\langle C_0, C_1, \ldots \rangle$ be an agent execution in transition system $\mathcal{T}$ consisting of the rules $R1, \ldots, R4$ and $R5.1$. Given that the plan base in C_0 is $\mathcal{R}^{<>}$-non-conflicting then the plan base in C_i is $\mathcal{R}^{<>}$-non-conflicting $\forall i \in \mathbb{N}$.*

Proof. By induction on the depth of the execution. The plan base of C_0 is non-conflicting by assumption. Now assume that the plan base of C_k is non-conflicting for arbitrary $k \geq 0$. The only way in which the plan base can become conflicting is by adoption of a plan to the plan base. Suppose that in configuration C_{k+1} a plan for goal κ has been adopted by applying $R5.1$. From the premises of $R5.1$ it directly follows that $\forall(\kappa', \pi) \in \Pi^{k+1} : (\kappa, \kappa') \notin \mathcal{R}^{<>}$. From the symmetry of $\mathcal{R}^{<>}$ we conclude that $\forall(\kappa', \pi) \in \Pi^{k+1} : (\kappa', \kappa) \notin \mathcal{R}^{<>}$, which means that the plan base of C_{k+1} is also non-conflicting. □

3.2 Precedence of goals

The above defined incompatibility relation ensures that the agent refrains from adopting plans for goals that hinder the achievement of goals the agent is currently pursuing. It does not, however, guarantee a certain order in which the agent tries to achieve its goals. Under the interpretation of the incompatibility relation $\mathcal{R}^{<>}$ the choice between two incompatible goals for which no plan is adopted yet is non-deterministic. Sometimes, however, when an agent is faced with such a choice, one goal should have precedence over the other. Suppose, for instance, that the household agent cannot clean rooms in case its battery charge is low. Therefore, one would expect the agent to first achieve its goal to have its battery loaded (denoted by `battery_loaded`) before pursuing a goal to clean a room. In this subsection we provide the precedence relation $\mathcal{R}^{\prec}$ enabling the agent not only to avoid pursuing goals that cannot be achieved simultaneously, but also to choose between such incompatible goals.

Definition 11 (goal precedence relation). *A precedence relation $\mathcal{R}^{\prec} \subseteq (\mathcal{L}_G \times \mathcal{L}_G)$ is a set of pairs of goals such that:*

- $(\kappa, \kappa') \in \mathcal{R}^{\prec}$ *and* $(\kappa', \kappa'') \in \mathcal{R}^{\prec} \rightarrow (\kappa, \kappa'') \in \mathcal{R}^{\prec}$
- $(\kappa, \kappa') \in \mathcal{R}^{\prec} \rightarrow (\kappa', \kappa) \notin \mathcal{R}^{\prec}$ *and* $\kappa' \neq \kappa$

The intuitive meaning of the precedence relation is as follows. When some goals κ and κ' are related by $\mathcal{R}^{\prec}$, i.e. $(\kappa, \kappa') \in \mathcal{R}^{\prec}$, these goals are not to be pursued simultaneously, and when both κ and κ' are goals of the agent the achievement of κ has precedence over the achievement of κ'. Precedence implies an order in which goals are pursued. The relation $\mathcal{R}^{\prec}$ is irreflexive, i.e. no goal can have precedence over itself, and transitive. When, for example, the goal to

have the battery loaded precedes the goal to have room one clean (the room of the boss), and cleaning room one on its turn precedes a goal to have cleaned room two, then it seems not unreasonable to assume that the goal of having the battery loaded also precedes the goal to have cleaned room two.

Suppose, for example, that in generating its plans an agent is faced with a choice between two goals `battery_loaded` and `clean1`. Furthermore, assume that $(\texttt{battery\_loaded}, \texttt{clean1}) \in \mathcal{R}^{\prec}$. If the agent has not adopted a plan for one of them and does not have any goals with higher precedence than `battery_loaded`, then one might expect this agent to adopt a plan to load its battery. If, however, the agent was already working on a plan for cleaning room one before the goal `battery_loaded` was adopted, then applying a planning goal rule for `battery_loaded` results in a conflicting plan base. In the following we propose two different strategies to keep the plan base non-conflicting by providing two transition rules for plan generation with precedence. The first transition rule implements a strategy in which the agent stops pursuing goals with less precedence the moment a plan can be adopted for a goal with higher precedence. Plans the agent is already executing might thus be disrupted in case a plan is adopted for a more important goal.

R5.2 (disruptive precedence) *Let $C = \langle \sigma, \gamma, \Pi, \texttt{PG}, \mathcal{R}^{\prec} \rangle$ be a goal strategy agent where $\mathcal{R}^{\prec}$ is a precedence relation.*

$$\frac{(\kappa \leftarrow \beta \mid \pi) \in \mathit{appl}(\kappa, C) \text{ and } \forall \kappa' \in \mathcal{L}_G : (\gamma, \sigma) \models_g \kappa' \rightarrow (\kappa', \kappa) \notin R^{\prec}}{\langle \sigma, \gamma, \Pi \rangle \longrightarrow \langle \sigma, \gamma, \Pi' \rangle}$$

where $\Pi' = (\Pi \cup \{(\kappa, \pi)\}) \setminus \{(\kappa', \pi') \in \Pi : (\kappa, \kappa') \in \mathcal{R}^{\prec}\}$

In words, an applicable planning goal rule for κ is applied if no other goal has precedence over κ. Note that in contrast to transition rule R5.1 in the premises only conflicting goals are taken into account instead of plans in the plan base. It might thus seem that there are plans (κ', π') in the plan base of the agent such that $(\kappa', \kappa) \in \mathcal{R}^{\prec}$, which means that the agent has plans for goals that precede κ, and therefore conflict with κ. However, this will never be the case because the premises ensures that the agent has no goal that precedes κ, and as we show by the following corollary, every κ' associated to a plan of the agent is also a goal of the agent.

Corollary 2. *Let $\langle C_0, C_1, \ldots \rangle$ be an agent execution in transition system $\mathcal{T}$, where $C_i = \langle \sigma^i, \gamma^i, \Pi^i, \texttt{PG}, \mathcal{R}^{\prec} \rangle$, and $\mathcal{T}$ consists of the rules $R1, \ldots, R4$ and $R5.2$. Given that $\forall (\kappa, \pi) \in \Pi^0 : (\gamma^0, \sigma^0) \models_g \kappa$, then $\forall i \in \mathbb{N}. \forall (\kappa, \pi) \in \Pi^i : (\gamma^i, \sigma^i) \models_g \kappa$*

Proof. This follows from proposition 1 and the fact that $R5.2$ is a more restrictive version of $R5$. □

When a planning rule for a goal κ is applied, all plans associated with goals that are to be preceded by κ are dropped. This way it is ensured that the goal κ for which a plan has been adopted does not conflict with plans for goals with

less precedence. At this point we are able to show that if the agent starts with an empty plan base, with rules $R1, \ldots, R4$ and $R5.2$ it is guaranteed that the plan base stays non-conflicting in the rest of the execution.

Proposition 3. *Let $\langle C_0, C_1, \ldots \rangle$ be an agent execution in transition system $\mathcal{T}$ consisting of the rules $R1, \ldots, R4$ and $R5.2$ and let the plan base in C_0 be empty. Then the plan base in C_i is $\mathcal{R}^{\prec}$-non-conflicting $\forall i \in \mathbb{N}$.*

Proof. By induction on the depth of the execution. In C_0 the plan base is non-conflicting, because $\Pi = \emptyset$. Assume that for arbitrary $k \geq 0$ the plan base of C_k is non-conflicting. Now we have to prove that when a plan for goal κ is adopted by rule $R5.2$ for all plans in C_{k+1} for a goal κ' it holds that $(\kappa, \kappa'), (\kappa', \kappa) \notin \mathcal{R}^{\prec}$. The premises of rule $R5.2$ ensures that $(\kappa', \kappa) \notin \mathcal{R}^{\prec}$ for any goal κ'. From $\Pi^0 = \emptyset$ and corollary 2 it follows that $\forall(\kappa', \pi) \in \Pi^k : (\gamma, \sigma) \models_g \kappa'$, we can thus conclude that $\forall(\kappa', \pi) \in \Pi^{k+1} : (\kappa', \kappa) \notin \mathcal{R}^{\prec}$. That there are no plans in the plan base for κ' such that $(\kappa, \kappa') \in \mathcal{R}^{\prec}$ follows from the fact that all these plans are dropped once a plan for κ has been adopted. □

When the goal of the household robot to have loaded its battery should precede a goal to have a room clean, and the agent is not cleaning a room already, the agent should postpone the adoption of a plan for cleaning a room until its battery is loaded. In the following we show that for rules $R1, \ldots, R4, R5.2$ such behaviour can indeed be expected. More generally, when the agent has adopted a goal κ that has precedence over some other goal κ', then if the agent has not adopted a plan for κ' it will not do so as long as κ is still a goal of the agent. Recall that κ is a goal of the agent as long as the state denoted by κ is not believed to be reached.

Proposition 4. *Let transition system $\mathcal{T}$ consist of the rules $R1, \ldots, R4$ and $R5.2$. Let $C_0 = \langle \sigma^0, \gamma^0, \Pi^0, \mathtt{PG}, \mathcal{R}^{\prec} \rangle$ be an initial configuration where $(\gamma^0, \sigma^0) \models_g \kappa$, $(\gamma^0, \sigma^0) \models_g \kappa'$, $\forall \pi \in \mathtt{Plan} : (\kappa', \pi) \notin \Pi^0$ and $(\kappa, \kappa') \in \mathcal{R}^{\prec}$. Then for every execution with initial configuration C_0 in $\mathcal{T}$ it holds that $\forall_{i \geq 0} : ((\forall_{0 \leq j \leq i} : (\gamma^j, \sigma^j) \models_g \kappa) \rightarrow \forall \pi \in \mathtt{Plan} : (\kappa', \pi) \notin \Pi^i)$*

Proof. By induction on the depth of an execution $\langle C_0, C_1, \ldots \rangle$ with all of the above assumptions about C_0. Then $\forall \pi \in \mathtt{Plan} : (\kappa', \pi) \notin \Pi^0$ by assumption. Assume that up to arbitrary $k \geq 0$ it holds that $\forall_{0 \geq i \geq k} : ((\gamma^i, \sigma^i) \models_g \kappa$ and $\forall \pi \in \mathtt{Plan} : (\kappa', \pi) \notin \Pi^i)$. Now we have to prove that it cannot be that $(\gamma^{k+1}, \sigma^{k+1}) \models_g \kappa$ and $\forall \pi \in \mathtt{Plan} : (\kappa', \pi) \in \Pi^{k+1}$ which can only happen after application of $R5.2$. As the condition $\forall \kappa \in \mathcal{L}_G : (\gamma^k, \sigma^k) \models_g \kappa \rightarrow (\kappa, \kappa') \notin R^{\prec}$ of the premises of rule $R5.2$ is not satisfied, we can conclude that $\forall \pi \in \mathtt{Plan} : (\kappa', \pi) \notin \Pi^{k+1}$. □

Note that in the above proposition no assumptions are made about whether the agent is already working on a plan for κ or not. Even if no plan is adopted for κ and there are no applicable planning rules, the agent will not adopt a plan for goals that are to be preceded by κ. As a consequence, the execution of the

agent might block when all goals of the agent are to be preceded by a goal for which no plan can be adopted. A weaker version of R5.2 can be introduced such that a rule for κ' can be applied when there are no *applicable* rules for more important goals.

By dropping plans for goals with less precedence immediately after a PG-rule for some goal with higher precedence is enabled the agent might give up too soon. Particularly in situations in which goals with higher precedence are often adopted the agent might never finish a plan for some of its goals. The agent would then never reach those goals. For example, when the boss' room needs a lot of cleaning, the agent might never finish a plan for cleaning the other rooms. Therefore, we also propose a more cautious form of precedence, in which the agent persists to plans it has already adopted. With such a strategy the agent is more cautious about dropping plans. This strategy is captured by rule R5.3 as defined below.

R5.3 (cautious precedence) *Let* $C = \langle \sigma, \gamma, \Pi, \mathtt{PG}, \mathcal{R}^{\prec} \rangle$ *be a goal strategy agent where* $\mathcal{R}^{\prec}$ *is a precedence relation.*

$$\frac{(\kappa \leftarrow \beta \mid \pi) \in \mathit{appl}(\kappa, C) \text{ and } \forall \kappa' \in \mathcal{L}_G : (\gamma, \sigma) \models_g \kappa' \rightarrow (\kappa', \kappa) \notin R^{\prec} \text{ and } \forall \kappa' \in \mathcal{L}_G : (\kappa', \pi') \in \Pi \rightarrow (\kappa, \kappa') \notin R^{\prec}}{\langle \sigma, \gamma, \Pi \rangle \longrightarrow \langle \sigma, \gamma, \Pi' \rangle}$$

where $\Pi' = \Pi \cup \{(\kappa, \pi)\}$

The first clause of the premises of R5.3 states that an agent can apply an applicable planning goal rule for a goal κ if no other goal has precedence over κ. The second clause of the premises states that the agent is not already working on a goal κ' with less precedence than κ. The agent will thus persist working on a plan for a goal even though a goal planning rule for a goal with higher precedence is applicable. Note that when the plan for such a conflicting goal κ' failed to achieve this goal, the agent will not retry with another plan as long as κ is still a goal of the agent. Under the assumption that no new goals with higher precedence than κ are adopted, the agent will adopt a plan for κ as soon as all plans for goals with lower precedence than κ are completed.

Note that just like in transition rule R5.2 only conflicting goals are taken into account instead of plans in the plan base. To avoid that a plan for a goal κ is adopted while there already is a plan (κ', π) in the plan base such that $(\kappa', \kappa) \in \mathcal{R}^{\prec}$, it is needed that every κ' associated to a plan of the agent is also a goal of the agent. This is shown by the corollary below.

Corollary 3. *Let* $\langle C_0, C_1, \ldots \rangle$ *be an agent execution in transition system* $\mathcal{T}$*, where* $C_i = \langle \sigma^i, \gamma^i, \Pi^i, \mathtt{PG}, \mathcal{R}^{\prec} \rangle$*, and* $\mathcal{T}$ *consists of the rules* $R1, \ldots, R4$ *and* $R5.3$*. Given that* $\forall (\kappa, \pi) \in \Pi^0 : (\gamma^0, \sigma^0) \models_g \kappa$*, then* $\forall i \in \mathbb{N}. \forall (\kappa, \pi) \in \Pi^i : (\gamma^i, \sigma^i) \models_g \kappa$*.*

Proof. This follows from proposition 1 and the fact that $R5.3$ *is a more restrictive version of* $R5$*.* □

Next, we show that similar to proposition 3, in a transition system consisting of the rules $R1, \ldots, R4$ and $R5.3$ the plan base of the agent stays non-conflicting during the execution of an agent that started with an empty plan base.

Proposition 5. *Let $\langle C_0, C_1, \ldots \rangle$ be an agent execution in transition system $\mathcal{T}$ consisting of the rules $R1, \ldots, R4$ and $R5.3$ and let the plan base in C_0 be empty. Then the plan base in C_i is $\mathcal{R}^{\prec}$-non-conflicting $\forall i \in \mathbb{N}$.*

Proof. Observe that rule $R5.3$ is a more restricted version of $R5.2$ and ensures that no goal κ is adopted when $\exists(\kappa', \pi) \in \Pi : (\kappa, \kappa') \in \mathcal{R}^{\prec}$. The proof is therefore similar to the proof of proposition 3. □

Just like we have shown for rules $R1, \ldots, R4, R5.2$ we show that with rules $R1, \ldots, R4, R5.3$ an agent that has not already adopted a plan for a goal κ' will not do so as long as the agent has a goal κ with higher precedence, i.e. $(\kappa, \kappa') \in \mathcal{R}^{\prec}$.

Proposition 6. *Let transition system $\mathcal{T}$ consist of the rules $R1, \ldots, R4$ and $R5.3$. Let $C_0 = \langle \sigma^0, \gamma^0, \Pi^0, \mathtt{PG}^0, \mathcal{R}^{\prec} \rangle$ be an initial configuration where $(\gamma^0, \sigma^0) \models_g \kappa$, $(\gamma^0, \sigma^0) \models_g \kappa'$, $\forall \pi \in \mathtt{Plan} : (\kappa', \pi) \notin \Pi^0$ and $(\kappa, \kappa') \in \mathcal{R}^{\prec}$. Then for every execution with initial configuration C_0 in $\mathcal{T}$ it holds that $\forall_{i \geq 0} : ((\forall_{0 \leq j \leq i} : (\gamma^j, \sigma^j) \models_g \kappa) \rightarrow \forall \pi \in \mathtt{Plan} : (\kappa', \pi) \notin \Pi^i)$*

Proof. Similar to the proof of proposition 4. □

We have already shown that all three transition systems guarantee that if a plan base is non-conflicting initially it remains non-conflicting during the agent's execution. We have not shown, however, to what extent the different transition systems differ from each other. We omit a formal proof that the behaviour of a transition system with disruptive precedence differs from one with cautious precedence, as we believe that the difference should be clear; the first will drop a plan for less important goals immediately after a planning rule is applied for a goal with higher precedence, while the latter will wait for plans associated to goals with less precedence to finish before applying a planning rule for a goal with higher precedence. We will, however, as a final property show that both forms of plan generation with the precedence relation generate different behaviour than plan generation with incompatibility. The crux is that with the rules $R1, \ldots, R4, R5.1$ no order of the pursuit of goals is assured.

Proposition 7. *Let transition system $\mathcal{T}_1$ consist of rules $R1, \ldots, R4$, $R5.1$, transition system $\mathcal{T}_2$ of $R1, \ldots, R4$, $R5.2$, and transition system $\mathcal{T}_3$ of $R1, \ldots, R4$, $R5.3$. Neither transition system $\mathcal{T}_2$ nor $\mathcal{T}_3$ can generate the same behaviour as $\mathcal{T}_1$.*

Proof. Assume that $\forall \pi \in \mathtt{Plan} : (\kappa, \pi) \notin \Pi^0$, $(\kappa' \leftarrow \beta | \pi) \in appl(\kappa', C_0)$, and that $\mathcal{R}^{<>} = \{(\kappa, \kappa'), (\kappa', \kappa)\}$. Recall that if a rule for κ is applicable then κ is a goal of the agent. Then after applying $R5.1$ for $(\kappa' \leftarrow \beta | \pi)$, in C_1 it holds that $(\kappa', \pi) \in \Pi^1$ and $(\gamma^1, \sigma^1) \models_g \kappa$. According to proposition 4 and proposition 6 such an execution is not possible in $\mathcal{T}_2$ nor in $\mathcal{T}_3$. □

4 Conclusions, related work and future research

In this paper, we have introduced three types of goal selection strategies as an explicit and generic mechanism to process incompatible goals. These mechanisms prevent an agent from simultaneously pursuing goals that are incompatible with each other, and enable the agent to choose from possibly incompatible goals in adopting plans to reach its goals. We have presented the proposed mechanism in the context of a simple agent programming language that can be extended to most BDI-based agent programming languages. The three goal selection strategies are implemented as conditions for the application of goal planning rules. These strategies are integrated in the transition rules for PG-rule applications. It should be noted that our account of precedence might look like a preference relation. However, it should be emphasized that the precedence relation is defined as a programming construct to help an agent to choose a goal from a (possibly) incompatible set of goals. It is not a concept agents reason about.

Related to our work is the work presented in [12], which describes the structure of a goal model which can be used by an agent to reason about goals during its deliberation process and means-ends reasoning. As part of this model an inconsistency operator is provided to denote that the success of one goal implies the failure of another. Also a preference operator is provided to express that in case of inconsistency between goals one goal is preferred to another. Also related to our work is the goal deliberation strategy as proposed in [13]. This strategy allows agent developers to specify the relationship between incompatible goals in order to avoid negative interference in pursuing multiple goals. This relation also implies a precedence of one goal over another. The main difference between our work and that of [12] and [13] is that our proposal is not limited to a specific platform, but can be integrated in existing BDI-based agent programming language. Furthermore, [12] and [13] do not provide a formal semantics of the proposed constructions. The lack of a formal semantics makes it hard to compare different approaches.

Another solution that involves avoiding negative interference in pursuing multiple goals is the one proposed in [14]. Possible conflicts are detected and avoided by annotating plans and goals with detailed information about their effects, pre-conditions, and in-conditions. Just as observed in [13] we believe that acquiring and assigning such information to goals and plans is a cumbersome task for an agent programmer. Also, in contrast to our approach, it is not possible to enforce that one goal precedes another. Moreover, as we have integrated the goal selection strategies in transition rules we believe that our approach can be directly used to build agent interpreters that can process incompatible goals.

In the current proposal different strategies are studied apart from each other. We are currently investigating new relations that can be used to denote the incompatibility of goals. Further, for now it is the task of the agent programmer to specify which goals are incompatible with each other. We envisage a mechanism that detects incompatibility automatically. As for now, $\mathcal{R}$ is defined on only the head of PG-rules. A possible extension would be to define $\mathcal{R}$ on the entire PG-rules, allowing for a more fine grained specification of incompatibility. Finally,

we observe that in the current approach the precedence of goals is fixed, while in some cases precedence might depend on a specific context, e.g., the current beliefs of the agent. Investigating how the precedence relation can be extended taking the context into account remains for further research.

References

1. Pokahr, A., Braubach, L., Lamersdorf, W.: Jadex: A BDI Reasoning Engine. [15] 149–174
2. Winikoff, M., Padgham, L., Harland, J., Thangarajah, J.: Jack intelligent agents: An industrial strength platform. [15] 175–193
3. Bordini, R., Hübner, J., Vieira, R.: Jason and the Golden Fleece of agent-oriented programming. [15] 3–37
4. Dastani, M., van Riemsdijk, M., Meyer, J.: Programming Multi-Agent Systems in 3APL. [15] 39–67
5. Dix, J., Zhang, Y.: IMPACT: A Multi-Agent Framework with Declarative Semantics. [15] 69–122
6. Fallah-Seghrouchni, A.E., Suna, A.: CLAIM and SyMPA: A Programming Environment for Intelligent and Mobile Agents. [15] 95–122
7. Dastani, M., Meyer, J.: A Practical Agent Programming Language. In: Proc. of the fifth Int. Workshop on Programming Multi-agent Systems. (2007)
8. Winikoff, M., L. Padgham, J.H., Thangarajah, J.: Declarative and Procedural Goals in Intelligent Agent Systems. In: Proc. of the Eighth Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2002). (2002)
9. Dastani, M., van Riemsdijk, M.B., Meyer, J.: Goal types in agent programming. In: Proc. of AAMAS. (2006) 1285–1287
10. van Riemsdijk, M.B., Dastani, M., Meyer, J., de Boer, F.S.: Goal-oriented modularity in agent programming. In: Proc. of AAMAS. (2006) 1271–1278
11. van Riemsdijk, M.B., Dastani, M., Meyer, J.J.C.: Semantics of declarative goals in agent programming. In: Proc. of AAMAS. (2005) 133–140
12. Morreale, V., Bonura, S., Francaviglia, G., Centineo, F., Cossentino, M., Gaglio, S.: Goal-Oriented Development of BDI Agents: The PRACTIONIST Approach. In: IAT '06: Proc. of the IEEE/WIC/ACM int. conf. on Intelligent Agent Technology. (2006)
13. Pokahr, A., Braubach, L., Lamersdorf, W.: A Goal Deliberation Strategy for BDI Agent Systems. In Eymann, T., Klügl, F., Lamersdorf, W., Klusch, M., Huhns, M., eds.: Third German conf. on Multi-Agent System TEchnologieS (MATES-2005), Springer-Verlag, Berlin Heidelberg New York (2005) 82–94
14. Thangarajah, J., Padgham, L., Winikoff, M.: Detecting & Avoiding Interference Between Goals in Intelligent Agents. In: Proc. of the 18th Int. Joint Conference on Artificial Intelligence. (2003)
15. Bordini, R., Dastani, M., Dix, J., Fallah-Seghrouchni, A.E., eds.: Multi-Agent Programming: Languages, Platforms and Applications. Springer (2005)