A Formal Approach to Embedding First-Principles Planning in BDI Agent Systems

Mengwei Xu(✉), Kim Bauters, Kevin McAreavey, and Weiru Liu

University of Bristol, Bristol, UK
{mengwei.xu, kim.bauters, kevin.mcareavey, weiru.liu}@bristol.ac.uk

Abstract. The BDI architecture, where agents are modelled based on their beliefs, desires, and intentions, provides a practical approach to developing intelligent agent systems. However, these systems either do not include any capability for first-principles planning (FPP), or they integrate FPP in a rigid and ad-hoc manner that does not define the semantical behaviour. In this paper, we propose a novel operational semantics for incorporating FPP as an intrinsic planning capability to achieve goals in BDI agent systems. To achieve this, we introduce a declarative goal intention to keep track of declarative goals used by FPP and develop a detailed specification of the appropriate operational behaviour when FPP is pursued, succeeded or failed, suspended, or resumed in the BDI agent systems. Furthermore, we prove that BDI agent systems and FPP are theoretically compatible for principled integration in both offline and online planning manner. The practical feasibility of this integration is demonstrated, and we show that the resulting agent framework combines the strengths of both BDI agent systems and FPP, thus substantially improving the performance of BDI agent systems when facing unforeseen situations.

Keywords: BDI agent systems · First-principles planning
Decision making under uncertainty

1 Introduction

A well-studied and widely applied architecture for developing intelligent agents is the so-called Belief-Desire-Intention (BDI) paradigm. BDI builds a sound theoretical foundation to model an agent with an explicit representation of (B)eliefs, (D)esires, and (I)ntentions. This BDI paradigm has inspired a multitude of agent-oriented programming languages, such as AgentSpeak [1], Can [2], and CanPlan [3]. Notable BDI agent software platforms include, for example, Jack [4], Jason [5], and Jadex [6].

BDI agent systems are recognised for their efficiency and scalability in complex application domains, such as control systems [7] and power engineering [8]. However, they have often avoided the use of first-principles planning (FPP) in favour of a pre-defined plan library. While the use of a set of pre-defined plans simplifies the planning problem to an easier plan selection problem, obtaining a plan library that can cope with every possible eventuality requires adequate plan knowledge. This knowledge is not always available, particularly when dealing with uncertainty. Therefore, this limits the applicability and autonomy of BDI agent systems when there is no applicable plan for achieving a goal at hand. FPP can, on the other hand, synthesise a new plan to achieve a goal for which either no pre-defined plan worked or exists.

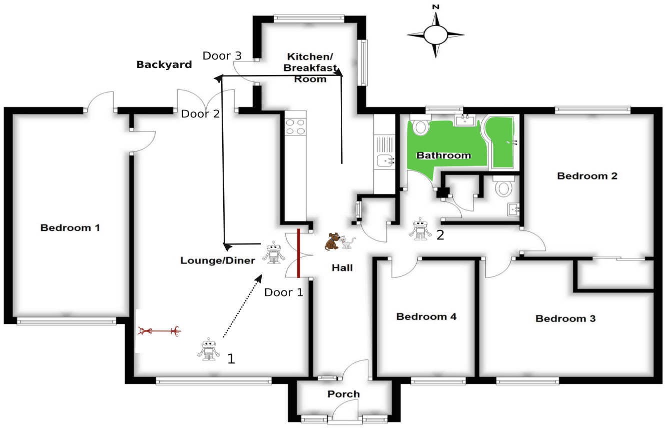

Fig. 1. Layout of a smart house with a domestic robot

To illustrate the problem, consider the following running example (see Fig. 1). In a smart home environment, there is an intelligent domestic robot whose job includes daily household chores (e.g. sweeping), security monitoring (e.g. burglary), and entertainment (e.g. playing music). The environment is dynamic and pervaded by uncertainty. When the robot does chores in the `lounge`, it may not be pre-encoded with plans to deal with an overturned clothes rack in the `lounge`, one of the doors to the `hall` being blocked unexpectedly, or urgent water overflow in a bathroom. Indeed, it is unreasonable to expect an agent designer to foresee all exogenous events and provide suitable pre-defined plans for all such eventualities. To address this weakness, a robot agent should be able to make use of FPP to generate novel plans to deal with such unforeseen events at design time in order to act intelligently.

Fortunately, to alleviate (some of) these issues, a large body of work on integrating various planning techniques with BDI have been proposed in recent years, as reviewed in [9]. For example, the work of [10] proposed an integration of AgentSpeak and a classical first-principles planner in which a new planning action in AgentSpeak is introduced to incorporate this planner. The BDI agent designer may include this new planning action at any point within a standard AgentSpeak plan to call a planner. In the work of [11], the authors provide a formal framework for FPP in BDI agent systems. This framework employs FPP to generate abstract plans, that is, plans that includes not only primitive actions, but also abstract actions summarised from the plan library. It allows for flexibility and robustness during the execution of these abstract plans. However, *none of these works have provided an operational semantics* that defines the behaviours of a BDI system with a built-in FPP. Moreover, the existing BDI systems (e.g. [10–12]) that integrate with FPP require the agent designer to define when the FPP is triggered. This limits the power and advantage of FPP to assist BDI agent systems to effectively accomplish their goals as the points of calling FPP can be unpredictable. Another important motivation of this paper is to respond to *the lack of work in strengthening the theoretical foundations of the BDI agent* pointed out by the comprehensive survey paper [9] as one of the future directions for planning in BDI agents. Therefore, the goal of this paper is to advance the state-of-art of planning in BDI agents by developing a rich and detailed specification of the appropriate operational behaviour when FPP is pursued, succeeded or failed, suspended, or resumed. In doing so, we introduce a novel operational semantics for FPP in BDI agent systems. This semantics specifies when and how FPP can be called, and articulates how a BDI agent system executes the new plan generated by FPP. To the best of our knowledge, we are not aware of any work on this problem so far.

The contributions of this paper are threefold. Firstly, we give a precise account of FPP within a typical BDI agent programming language, namely Can. Secondly, the formal relationship between FPP and the BDI agent execution is established. Finally, a scenario case study is presented to highlight the usefulness and feasibility of the integration of FPP into BDI agent systems.

The remainder of the paper is organised as follows. In Sect. 2, we provide a brief overview of BDI and FPP. Sections 3.1 and 3.2 present the full operational semantics for integrating FPP into BDI. In Sect. 3.3, we establish the formal relationship between FPP and the BDI execution. In Sect. 4, the paper offers an intricate scenario discussion, which supports the feasibility of the resulting integrated framework and motivates the merits of the proposed framework to warrant future work on a fully implemented system. Section 5 discusses related work. Finally, in Sect. 6, we draw conclusions and outline future lines of research.

2 Preliminaries

Can formalises the behaviours of a classical BDI agent, which is specified by a 5-tuple configuration $C = \langle \mathcal{B}, \Pi, \Lambda, \mathcal{A}, \Gamma \rangle$. The belief base $\mathcal{B}$ is a set of formulas encoding the current beliefs. The plan library Π is a collection of plan rules of the form $e : \varphi \leftarrow P$ with e the triggering event, φ the context condition, and P the plan-body program. The language used in the plan-body program P is defined by the following Backus-Naur grammar:

$$\begin{gathered} P ::= nil \mid act \mid ?\varphi \mid +b \mid -b \mid !e \mid P_1; P_2 \mid P_1 \triangleright P_2 \mid P_1 \parallel P_2 \mid \\ (\!| \psi_1 : P_1, \cdots, \psi_n : P_n |\!) \mid goal(\varphi_s, P, \varphi_f) \end{gathered}$$

with *nil* an empty program, *act* a primitive action, $?\varphi$ a test for φ entailment in the belief base, $+b$ and $-b$ respectively belief addition and deletion, and $!e$ a subgoal. In addition, we use $P_1; P_2$ for sequence, $P_1 \triangleright P_2$ to execute P_2 only on failure of P_1, and $P_1 \parallel P_2$ for interleaved concurrency. A set of relevant plans is encoded by $(|\psi_1 : P_1, \cdots, \psi_n : P_n|)$. A goal program $goal(\varphi_s, P, \varphi_f)$ states that the declarative goal φ_s should be achieved through the procedural program P, failing when φ_f becomes true and retrying (alternatives) as long as neither φ_s nor φ_f is true (see [13]). The action library Λ is a collection of actions *act* in the form of $a : \psi \leftarrow \phi^-; \phi^+$. We have that ψ is the precondition, while ϕ^- and ϕ^+ denote respectively a delete and add set of belief atoms, i.e. propositional atoms. The sequence of actions executed so far by an agent is denoted as $\mathcal{A}$. The intention base Γ consists of a set of (partially) executed plans P. A basic configuration $\langle \mathcal{B}, \mathcal{A}, P \rangle$[1], with the plan-body program P being executed (i.e. the current intention), is also often used in notations to define what it means to execute a single intention.

The operational semantics for a BDI agent are defined in terms of configurations $\mathcal{C}$ and transitions $\mathcal{C} \rightarrow \mathcal{C}'$. A transition $\mathcal{C} \rightarrow \mathcal{C}'$ denotes that executing a single step in configuration $\mathcal{C}$ yields $\mathcal{C}'$. We write $\mathcal{C} \rightarrow$ (resp. $\mathcal{C} \nrightarrow$) to state that there is (resp. is not) a $\mathcal{C}'$ such that $\mathcal{C} \rightarrow \mathcal{C}'$, and $\xrightarrow{*}$ to denote the transitive closure of $\rightarrow$. A derivation rule specifies in which cases an agent can transition to a new configuration. Such a rule consists of a (possibly empty) set of premises p_i and a single transition conclusion c, denoted by

$$\frac{p_1 \quad p_2 \quad \cdots \quad p_n}{c} \; l$$

where l is a label for reference. We refer the reader to [2,13] for a full overview of the semantics of Can.

A FPP problem is defined as a 3-tuple $P = \langle \varphi_s, \mathcal{B}, \Lambda \rangle$, where φ_s is a set of successful goal states to be achieved, i.e. a set of formulas over some logical language, $\mathcal{B}$ stands for a set of initial belief states, and Λ represents the action library (defined as before). A first-principles planner takes as input the models of all known actions (i.e. action library Λ), a description of the state of the world (i.e. the initial state $\mathcal{B}$), and some objective (i.e. goals φ_s). It returns a sequence of actions σ which solves P, denoted $\sigma = sol(P)$.

3 First-Principles Planning in BDI Agent Systems

We now discuss how Can agent systems and first-principles planning (FPP) can be integrated into a single framework. The resulting framework, called Can(FPP), allows us to define agents that can perform FPP to provide new behaviours at runtime in an uncertain environment. We start by introducing the concept of declarative goal intention (used by FPP) and its semantical operation

[1] The plan and action libraries Π and Λ are omitted under the assumption that they are static entities, i.e. they remain unchanged as the agent moves between configurations.

in Sect. 3.1. The semantical behaviours of FPP within BDI execution presented in Sect. 3.2 is subsequently underpinned by the formal relationship between FPP and BDI execution in Sect. 3.3.

3.1 Declarative Goal Intention for First-Principles Planning

In a CAN agent, the intention set $\mathit{\Gamma}$ is limited to just procedural goals. While valuable, procedural goals only describe *how* to achieve a given goal and do not answer the question as to which goals FPP should be trying to achieve in the BDI agent. To address this shortcoming, we modify the intention in this work to be a pair of sets, such that $\mathit{\Gamma} = \langle \mathit{\Gamma}_{pr}, \mathit{\Gamma}_{de} \rangle$ with $\mathit{\Gamma}_{pr}$ and $\mathit{\Gamma}_{de}$ a set of procedural and declarative goals, respectively. It allows us to keep track of both procedural goals (executed by the BDI engine) and declarative goals that tells us *what* we want to achieve (used by FPP). The set of declarative goals is furthermore partitioned into the subset of active goals $\mathit{\Gamma}^{+}_{de}$, and the suspended goals $\mathit{\Gamma}^{-}_{de}$. As a (slight) abuse of notation, we assume that adding an element to $\mathit{\Gamma}^{+}_{de}$ ensures the element is removed from $\mathit{\Gamma}^{-}_{de}$ and vice versa.

We start with the definition of a pure declarative goal and semantically enumerate three strategies, namely, *direct*, *belief-driven*, or *recovery-aid* strategy, to add such a pure declarative goal into the declarative goal intention to plan for by FPP.

A pure declarative goal $goal(\varphi_s, \varphi_f)$ is obtained from the ordinary declarative goal $goal(\varphi_s, P, \varphi_f)$ in CAN by dropping the procedural component P. It is read as "*achieve* φ_s; *failing if* φ_f *becomes true*" and defined to be the element of the declarative goal intention $\mathit{\Gamma}_{de}$. This new goal structure encodes the minimum information of what FPP needs to achieve (i.e. successful state φ_s) and when it is sensible to halt FPP (i.e. failure state φ_f).

The **first** *direct* strategy is to add a pure declarative goal into declarative goal intention when an ordinary declarative goal $goal(\varphi_s, nil, \varphi_f)$ is initially written as a part of the plan-body program. Here, $P = nil$ implies that there is no available procedural information on how to achieve the goal. Such a scenario occurs when either the procedural plan was not known during design time, or no efforts were made to create pre-defined plans (e.g. due to the priority of other part of plan library design tasks). Once the BDI agent selects goal $goal(\varphi_s, nil, \varphi_f)$ into procedural goal intention set $\mathit{\Gamma}_{pre}$ (first premise), a pure declarative goal $P^{\uparrow} = goal(\varphi_s, \varphi_f)$ is automatically added to $\mathit{\Gamma}_{de}$ by dropping nil if $goal(\varphi_s, \varphi_f)$ is not already in $\mathit{\Gamma}_{de}$ (second premise):

$$\frac{P = goal(\varphi_s, nil, \varphi_f) \in \mathit{\Gamma}_{pre} \quad P^{\uparrow} \notin \mathit{\Gamma}_{de}}{\langle \mathcal{B}, \mathit{\Pi}, \mathit{\Lambda}, \mathcal{A}, \mathit{\Gamma} \rangle \xrightarrow{goal} \langle \mathcal{B}, \mathit{\Pi}, \mathit{\Lambda}, \mathcal{A}, \langle \mathit{\Gamma}_{pr} \setminus \{P\}, \mathit{\Gamma}^{+}_{de} \cup \{P^{\uparrow}\} \rangle \rangle} \; A^{1}_{goal}$$

The **second** *belief-driven* strategy is to allow adding a pure declarative goal to $\mathit{\Gamma}_{de}$ in a proactive manner through the motivational library $\mathcal{M}$[2]. Inspired

[2] We only explicitly mention $\mathcal{M}$ in the agent configuration of A^{2}_{goal}; for all other rules, the library does not change and is omitted.

by conditionalised goals [14], a motivation planning library $\mathcal{M}$ is, a collection of rules of the form: $\psi \rightsquigarrow goal(\varphi_s, nil, \varphi_f)$, to add a declarative goal based on changes in beliefs. Semantically, we add a derivation rule for the motivational library $\mathcal{M}$ so that a pure declarative goal is added to Γ_{de} when the rule is triggered (second premise), the goal has an empty procedural component (third premise), and the goal $goal(\varphi_s, \varphi_f)$ is not already in Γ_{de} (fourth premise):

$$\frac{\psi \rightsquigarrow P \in \mathcal{M} \quad \mathcal{B} \models \psi\theta \quad P = goal(\varphi_s, nil, \varphi_f) \quad P^{\uparrow}\theta \notin \Gamma_{de}}{\langle \mathcal{B}, \Pi, \Lambda, \mathcal{A}, \Gamma \rangle \xrightarrow{goal} \langle \mathcal{B}, \Pi, \Lambda, \mathcal{A}, \langle \Gamma_{pr} \setminus \{P\}, \Gamma_{de}^{+} \cup \{P^{\uparrow}\theta\} \rangle \rangle} \; A_{goal}^{2}$$

The **third** *recovery-aid* strategy is to overcome the limitations of the first and second strategy by recovering the unexpected failure of procedural plan program. It adopts $goal(\varphi_s, \varphi_f)$ into Γ_{de} when an ordinary declarative goal $goal(\varphi_s, P', \varphi_f)$ has a blocked procedural component, i.e. $P' \neq nil$ and $\langle \mathcal{B}, \mathcal{A}, goal(\varphi_s, P', \varphi_f) \rangle \nrightarrow$. The failure handling mechanism in [13] is already capable of (partially) dealing with such a situation. However, goals can still be blocked if failure handling mechanism failed. Since one of the properties of declarative goals held by a rational agent is that they should be persistent [2], it is rational to try and pursue these blocked declarative goals using FPP. We have:

$$\frac{P' = (P'' \rhd P'') \vee nil \quad \langle \mathcal{B}, \mathcal{A}, goal(\varphi_s, P', \varphi_f) \rangle \nrightarrow \quad P^{\uparrow} \notin \Gamma_{de}}{\langle \mathcal{B}, \Pi, \Lambda, \mathcal{A}, \Gamma \rangle \xrightarrow{goal} \langle \mathcal{B}, \Pi, \Lambda, \mathcal{A}, \langle \Gamma_{pr} \setminus \{goal(\varphi_s, P', \varphi_f)\}, \Gamma_{de}^{+} \cup \{P^{\uparrow}\} \rangle \rangle} \; A_{goal}^{3}$$

where $P^{\uparrow} = goal(\varphi_s, \varphi_f)$.
Finally, when either φ_s or φ_f is true, the pure declarative goal $goal(\varphi_s, \varphi_f)$ has been completed and it is dropped from Γ_{de}:

$$\frac{G \in \Gamma_{de} \quad G = goal(\varphi_s, \varphi_f) \quad \mathcal{B} \models \varphi_s \vee \varphi_f}{\langle \mathcal{B}, \Pi, \Lambda, \mathcal{A}, \Gamma \rangle \xrightarrow{drop} \langle \mathcal{B}, \Pi, \Lambda, \mathcal{A}, \Gamma_{de} \setminus \{G\} \rangle} \; G_{drop}$$

3.2 First-Principles Planning for Declarative Goals

We now consider how to invoke FPP and how it integrates with the BDI system and we use $plan(goal(\varphi_s, \varphi_f))$ to symbolise calling FPP.

The following two derivation rules G_s and G_f handle the cases where either the success condition φ_s or the failure condition φ_f holds.

$$\frac{\mathcal{B} \models \varphi_s}{\langle \mathcal{B}, \mathcal{A}, plan(goal(\varphi_s, \varphi_f)) \rangle \longrightarrow \langle \mathcal{B}, \mathcal{A}, nil \rangle} \; G_s \qquad \frac{\mathcal{B} \models \varphi_f}{\langle \mathcal{B}, \mathcal{A}, plan(goal(\varphi_s, \varphi_f)) \rangle \longrightarrow \langle \mathcal{B}, \mathcal{A}, ?false \rangle} \; G_f$$

Intuitively, on success the plan is completed (replaced by nil). On failure, this is signalled to the BDI agent so that the basic CAN semantics can take over again. From now on, we will also distinguish between online planning [15] and offline planning [16] and give different derivation rules for accommodating each style of FPP due to their contrasting nature.

In *offline* planning, a complete sequence of actions σ to solve FPP problem $\langle \varphi_s, \mathcal{B}, \Lambda \rangle$ is generated first and executed afterwards.

The derivation rule for *offline* planning is defined as follows:

$$\frac{P = plan(goal(\varphi_s, \varphi_f)) \quad \langle \mathcal{B}, \mathcal{A}, P \rangle \xrightarrow{plan} \langle \mathcal{B}', \mathcal{A}', \sigma \rangle \quad \langle \mathcal{B}, \mathcal{A}, \sigma \rangle \xrightarrow{bdi_*} \langle \mathcal{B}'', \mathcal{A}'', nil \rangle}{\langle \mathcal{B}, \mathcal{A}, \Gamma \rangle \xrightarrow{bdi} \langle \mathcal{B}, \mathcal{A}, \Gamma' \rangle} \; P_{\mathcal{F}^{off}}$$

where $\Gamma = \langle \Gamma_{pr}, \Gamma^+_{de} = \{goal(\varphi_s, \varphi_f)\} \rangle$ and $\Gamma' = \langle \Gamma_{pr} \cup \{\sigma\}, \Gamma_{de} \setminus \{goal(\varphi_s, \varphi_f)\} \rangle$.

It shows that the configuration $\langle \mathcal{B}, \mathcal{A}, \langle \Gamma_{pr}, \Gamma^+_{de} = \{goal(\varphi_s, \varphi_f)\} \rangle \rangle$ will evolve to $\langle \mathcal{B}, \mathcal{A}, \langle \Gamma_{pr} \cup \{\sigma\}, \Gamma_{de} \setminus \{goal(\varphi_s, \varphi_f)\} \rangle \rangle$ if FPP can generate a sequence of actions σ (i.e. $\langle \mathcal{B}, \mathcal{A}, P \rangle \xrightarrow{plan} \langle \mathcal{B}, \mathcal{A}, \sigma \rangle$) that can achieve the successful state φ_s (i.e. $\langle \mathcal{B}, \mathcal{A}, \sigma \rangle \xrightarrow{bdi_*} \langle \mathcal{B}'', \mathcal{A}'', nil \rangle$). Once the successful state φ_s is met after the execution of the sequence of actions σ, the structure $plan(goal(\varphi_s, \varphi_f))$ will transition into nil and the goal $goal(\varphi_s, \varphi_f)$ will be dropped from Γ_{de} (i.e. $\Gamma_{de} \setminus \{goal(\varphi_s, \varphi_f)\}$) according to the above derivation rule G_s and G_{drop}.

In *online* planning, a single action is returned based on current belief states instead of generating the whole plan a priori, and executed immediately. The next action will be generated based on newly reached belief states. The loop of "*plan one action–execute one action*" will be iterated until the goal is reached.

The derivation rule for *online* planning is defined as follows:

$$\frac{P = plan(goal(\varphi_s, \varphi_f)) \quad \langle \mathcal{B}, \mathcal{A}, P \rangle \xrightarrow{plan} \langle \mathcal{B}', \mathcal{A}', act \rangle \quad act\theta = a}{\langle \mathcal{B}, \mathcal{A}, \Gamma \rangle \xrightarrow{bdi} \langle \mathcal{B}, \mathcal{A}, \Gamma' \rangle} \; P_{\mathcal{F}^{on}}$$

where $\Gamma = \langle \Gamma_{pr}, \Gamma^+_{de} = \{goal(\varphi_s, \varphi_f)\} \rangle$ and $\Gamma' = \langle \Gamma_{pr} \cup \{I\}, \Gamma^-_{de} \cup \{goal(\varphi_s, \varphi_f)\} \rangle$ with $I = act; activate(goal(\varphi_s, \varphi_f))$. The intention $act; activate(goal(\varphi_s, \varphi_f))$ pursues the action act which was returned from FPP. When the action act is executed, it ensures that FPP is called again through reactivating the goal $goal(\varphi_s, \varphi_f)$. As such, FPP can take the new belief into consideration and plan for the next action. These two interleaved planning and execution will be repeated until the successful state is achieved if all possible.

The derivation rule to reactive the suspended goal is as follows:

$$\frac{P \in \Gamma_{pr} \quad P = activate(goal(\varphi_s, \varphi_f))}{\langle \mathcal{B}, \mathcal{A}, \Gamma \rangle \xrightarrow{bdi} \langle \mathcal{B}, \mathcal{A}, \langle \Gamma_{pr} \setminus \{P\}, \Gamma^+_{de} \cup \{goal(\varphi_s, \varphi_f)\} \rangle \rangle} \; A_{re}$$

In addition, a trivial goal can be safely terminated:

$$\frac{}{\langle \mathcal{B}, \mathcal{A}, plan(nil) \rangle \xrightarrow{bdi} \langle \mathcal{B}, \mathcal{A}, nil \rangle} \; P_{\top}$$

Finally, when no solution is found to achieve goal state φ_s, the BDI agent will drop $plan(goal(\varphi_s, \varphi_f))$.

$$\frac{P = plan(goal(\varphi_s, \varphi_f)) \quad \langle \mathcal{B}, \mathcal{A}, P \rangle \overset{plan}{\not\longrightarrow} \langle \mathcal{B}', \mathcal{A}', nil \rangle}{\langle \mathcal{B}, \mathcal{A}, P \rangle \xrightarrow{bdi} \langle \mathcal{B}, \mathcal{A}, ?false \rangle} \; P_{\bot}$$

In Sect. 4, we will explore how the scenario from the introduction can be expressed using our Can(FPP) framework.

3.3 Formal Relationship Between FPP and BDI Execution

In this section, the relationship between FPP and the BDI execution is formally established. The following theorem establishes the link between $plan(goal(\varphi_s, \varphi_f))$ and FPP in both offline and online setting so that $plan(goal(\varphi_s, \varphi_f))$ can – to some extent – indeed be seen as FPP. In order to distinguish between offline and online planning, we denote an **off**line solution for a FPP problem $\langle \varphi_s, \mathcal{B}, \Lambda \rangle$ as $sol^{off}(\varphi_s, \mathcal{B}, \Lambda)$ and an **on**line solution as $sol^{on}(\varphi_s, \mathcal{B}, \Lambda)$.

Theorem 1. *For any agent,*

(i) For offline planning, we have $\langle \mathcal{B}, \mathcal{A}, plan(goal(\varphi_s, \varphi_f)) \rangle \xrightarrow{bdi_*} \langle \mathcal{B}'', \mathcal{A}'', nil \rangle \iff sol^{off}(\varphi_s, \mathcal{B}, \Lambda) = \sigma \neq \emptyset$, $\langle \mathcal{B}, \mathcal{A}, \sigma \rangle \xrightarrow{bdi_*} \langle \mathcal{B}'', \mathcal{A}'', nil \rangle$ *such that* $\mathcal{B}'' \models \varphi_s$. *The BDI agent can evolve* $plan(goal(\varphi_s, \varphi_f))$ *to an empty program as long as offline FPP returns a non-empty solution which can be successfully executed to solve FPP problem* $(\varphi_s, \mathcal{B}, \Lambda)$.

(ii) For online planning, $\langle \mathcal{B}, \mathcal{A}, plan(goal(\varphi_s, \varphi_f)) \rangle \xrightarrow{bdi} \iff sol^{on}(\varphi_s, \mathcal{B}, \Lambda) = act \neq \emptyset$ *and* $\langle \mathcal{B}, \mathcal{A}, act \rangle \xrightarrow{bdi}$. *The BDI agent can evolve* $plan(goal(\varphi_s, \varphi_f))$ *to a next step as long as online FPP returns an executable action.*

(iii) For online planning, $\langle \mathcal{B}_1, \mathcal{A}, plan(goal(\varphi_s, \varphi_f)) \rangle \xrightarrow{bdi_*} \langle \mathcal{B}_k, \mathcal{A} \cdot act_1 \cdot \ldots \cdot act_k, nil \rangle$ *with* $k \geq 1 \iff$ *there exists a solution for each online stage planning, i.e.* $sol^{on}(\varphi_s, \mathcal{B}_1, \Lambda) = act_1$, $sol^{on}(\varphi_s, \mathcal{B}_2, \Lambda) = act_2$, $\cdots$, *and* $sol^{on}(\varphi_s, \mathcal{B}_k, \Lambda) = act_k$ *such that* $\langle \mathcal{B}_j, \mathcal{A} \cdot act_1 \cdot \ldots \cdot act_{j-1}, act_j \cdot \ldots \cdot act_k \rangle \xrightarrow{bdi}$ *for* $j \in \{1, \cdots, k\}$ *and* $B_k \models \varphi_s$. *The BDI agent will successfully execute (i.e. will make the success condition* φ_s *true) if the goal can be achieved after the repetition of planning and execution.*

Proof. The proof of *(i)* relies on the derivation rule $P_{\mathcal{F}^{off}}$. In offline planning setting, the transition $\langle \mathcal{B}, \mathcal{A}, plan(goal(\varphi_s, \varphi_f)) \rangle \xrightarrow{bdi_*} \langle \mathcal{B}'', \mathcal{A}'', nil \rangle$ implies that there exists a complete sequence of actions which is generated from FPP (i.e. $sol^{off}(\varphi_s, \mathcal{B}, \Lambda) = \sigma \neq \emptyset$) such that it can then be successfully executed (i.e. $\langle \mathcal{B}, \mathcal{A}, \sigma \rangle \xrightarrow{bdi_*} \langle \mathcal{B}'', \mathcal{A}'', nil \rangle$) to achieve the goal state (i.e. $\mathcal{B}'' \models \varphi_s$). Hence, the right deduced from the left is proved. In order to prove from the right to the left, let us start from the derivation rule $P_{\mathcal{F}^{off}}$. Firstly, $sol^{off}(\varphi_s, \mathcal{B}, \Lambda) = \sigma \neq \emptyset$ means that $\langle \mathcal{B}, \mathcal{A}, plan(goal(\varphi_s, \varphi_f)) \rangle \xrightarrow{plan} \langle \mathcal{B}', \mathcal{A}', \sigma \rangle$ holds. Taking into account of that $\langle \mathcal{B}, \mathcal{A}, \sigma \rangle \xrightarrow{bdi_*} \langle \mathcal{B}'', \mathcal{A}'', nil \rangle$, the set of premises of the derivation rule $P_{\mathcal{F}^{off}}$ is satisfied. Therefore, a single transition conclusion derivable from these premises (i.e. $\langle \mathcal{B}, \mathcal{A}, \langle \Gamma_{pr}, \Gamma_{de}^{+} = \{goal(\varphi_s, \varphi_f)\} \rangle \xrightarrow{bdi} \langle \mathcal{B}, \mathcal{A}, \langle \Gamma_{pr} \cup \{\sigma\}, \Gamma_{de} \setminus \{goal(\varphi_s, \varphi_f)\} \rangle \rangle$) holds according to the derivation rule $P_{\mathcal{F}^{off}}$. The final puzzle of the proof, i.e. $\langle \mathcal{B}, \mathcal{A}, plan(goal(\varphi_s, \varphi_f)) \rangle \xrightarrow{bdi_*} \langle \mathcal{B}'', \mathcal{A}'', nil \rangle$ is solved by $\langle \mathcal{B}, \mathcal{A}, \sigma \rangle \xrightarrow{bdi_*} \langle \mathcal{B}'', \mathcal{A}'', nil \rangle$ and $\mathcal{B}'' \models \varphi_s$ again. Hence, the right implying the left is proved. Therefore, *(i)* holds.

The proof of *(ii)* can be given similarly as *(i)* but depending on the rule $P_{\mathcal{F}^{on}}$ instead. In online setting, the semantics $\langle \mathcal{B}, \mathcal{A}, plan(goal(\varphi_s, \varphi_f))\rangle \xrightarrow{bdi}$ indicates that a single action is returned (i.e. $sol^{on}(\varphi_s, \mathcal{B}, \Lambda) = act \neq \emptyset$) and can be executed (i.e. $\langle \mathcal{B}, \mathcal{A}, act\rangle \xrightarrow{bdi}$). According to the rule $P_{\mathcal{F}^o}$, if an executable action is produced by FPP, the configuration $\langle \mathcal{B}, \mathcal{A}, plan(goal(\varphi_s, \varphi_f))\rangle$ will transition to $\langle \mathcal{B}, \mathcal{A}, act\rangle$. Hence *(ii)* holds.

The proof of *(iii)* can be presented by induction on the planning step k. So if $k = 1$, then $act_1 \cdot ... \cdot act_n = \emptyset$. It means that $\langle \mathcal{B}, \Lambda, plan(goal(\varphi_s, \varphi_f))\rangle \overset{bdi}{\not\longrightarrow}$ is true if $sol^o(\langle \varphi_s, S_0, AS\rangle) = \emptyset$, which holds trivially. Therefore, *(iii)* holds. Next, suppose the claim holds for all numbers less than some $k \geq 1$. We show that *(iii)* holds for k. Since we have, by the hypothesis, that there exists a solution $act_2 \cdot act_3 \cdots act_k$ such that $\langle \mathcal{B}_j, \mathcal{A} \cdot act_1 \cdot ... \cdot act_{j-1}, act_j \cdot ... \cdot act_k\rangle \xrightarrow{bdi}$ for $j \in \{2, \cdots, k\}$ and $\mathcal{B}_k \models \varphi_s$ iff $\langle \mathcal{B}_2, \mathcal{A}, plan(goal(\varphi_s, \varphi_f))\rangle \xrightarrow{bdi_k} \langle \mathcal{B}_k, \mathcal{A} \cdot act_2 \cdot act_3 \cdots act_k, nil\rangle$. Clearly, we now only need to discuss the transition from $\langle \mathcal{B}_1, \mathcal{A}, plan(goal(\varphi_s, \varphi_f))\rangle$ to $\langle \mathcal{B}_2, \mathcal{A} \cdot act_1, plan(goal(\varphi_s, \varphi_f))\rangle$. If act_1 is a solution of FPP problem $\langle \varphi_s, \mathcal{B}_1, \Lambda\rangle$ for achieving the goal, then the problem is apparently solved already. Hence *(iii)* holds. If not, taking into consideration the hypothesis induction applying from 2 to k, *(iii)* holds still. Therefore, by induction, we have proved *(iii)*. □

This theorem underpins the theoretical foundation that a successful execution resulting from our operational rules for $plan(goal(\varphi_s, \varphi_f))$ corresponds directly to a sequence of actions from FPP. Concretely, *(i)* shows that if an offline planning step is able to start executing, then there is one solution for this FPP problem, provided there is no intervention from other concurrent intention of the BDI agent and the external environment. Both *(ii)* and *(iii)* unveil the dynamic aspect related to the online management of the plans.

4 Feasibility Study

In this section, we demonstrate the practical feasibility of integrating a BDI agent system with FPP. We show how the cleaning task scenario from the introduction can be expressed using our Can(FPP) framework. Without the loss of generality and for the simplicity of discussions, we consider the offline FPP and assume that the environment is dynamic (i.e. exogenous events can occur) and deterministic (i.e. the effects of actions can be precisely predicted). We stress though that the purpose of this discussion is not to present an actual fully developed Can(FPP) system, but rather to motivate the merits of the proposed framework to warrant future work on a fully implemented system. Therefore, we briefly discuss a prototype system which we designed to verify the feasibility of our approach as a basis for this future work.

We recall that in a cleaning task scenario in Fig. 1, a robot finished cleaning in the `lounge`, and needs to proceed to the `hall` to vacuum. There is a door labelled as `door1` connecting the `lounge` and the `hall`. The straight-forward

```
1   // Initial beliefs
2
3   dirty(hall)
4   location(lounge)
5   open(door1)
6   open(door2)
7   open(door3)
8   connect(door1, lounge, hall)
9   connect(door2, lounge, backyard)
10  connect(door3, backyard, hall)
11
12  // Initial goals
13
14  !clean(hall)
15
16  // Plan library
17
18  +!clean(X) : dirty(X) & location(X) <- vacuum(X); ? not dirty(X)
19
20  +!clean(X) : dirty(X) & location(Y) & connect(D, Y, X) & open(D)<-
goal(at(X), move(D, Y, X), nil); ? location(X); vacuum(X); ? not dirty(X)
```

Fig. 2. BDI agent in domestic cleaning scenario

route to the `hall` is to go through `door1` when it is open. There are also two doors, namely `door2` and `door3` which connect the `backyard` with the `lounge` and the `hall`, respectively. The design of this robot has been shown by its belief base, initial goal and plan library in Fig. 2. The initial beliefs of the robot are described on lines 3–10 and the initial goal to clean the `hall` is displayed on line 14 of Fig. 2. In this case, the achievement goal `!clean(hall)` is added to the event set of the robot as an external event. At this point, two plans in the plan library on lines 18–20 are stored as plans P_1 and P_2, and BDI agent reasoning cycle begins. Both of plans P_1 and P_2 are relevant plans for the event `!clean(hall)`. After validating and unifying the pre-condition given the current belief base, plan P_2 (see line 20) is identified as an applicable plan and becomes an intention in the procedural intention Γ_{pr} adopted for the execution. The execution of the body of P_2 starts from the execution of an ordinary declarative goal `goal(at(hall), move(door1,lounge,hall), nil)` which purses action `move(door1,lounge,hall)` to achieve the successful state `at(hall)` with empty failure state `nil`. However, it is realistic to expect in a real life setting that some situation will block the execution of the robot (i.e. exogenous events can occur). For example, in a scenario where the `door1` was slammed shut unexpectedly (i.e. `-open(door1)`) amidst the execution of the action `move(door1,lounge,hall)`. As a consequence, the action of `move(door1,lounge,hall)` would be undesirably halted, thus eventually causing the failure of the whole cleaning task.

To address this problem, the derivation rule A^3_{goal} in Sect. 3.1 will elevate the pure declarative goal `goal(at(hall), nil)` into the declarative intention

Γ_{de} with nil being no failure condition specified. Semantically, a FPP problem $P = \langle \texttt{at(hall)}, \mathcal{B}, \Lambda \rangle$ for $plan(\texttt{goal(at(hall), nil)})$ is to indicate that a first-principles planner will be triggered to generate a sequence of actions from action library Λ to achieve the successful state `at(hall)` in the initial belief state $\mathcal{B}$. When a sequence of actions σ is successfully generated in the offline fashion, the configuration $\langle \mathcal{B}, \mathcal{A}, plan(\texttt{goal(at(hall), nil)}) \rangle$ transitions to the configuration $\langle \mathcal{B}, \mathcal{A}, \sigma \rangle$. It follows that the BDI agent starts to execute actions in σ in turn in order to reach a goal `at(hall)`. The goal is achieved if and only if $\langle \mathcal{B}, \mathcal{A}, \sigma \rangle \xrightarrow{bdi_*} \langle \mathcal{B}'', \mathcal{A} \cdot \sigma, nil \rangle$ such that $\mathcal{B}'' \models \texttt{at(hall)}$. In practice, the BDI agent will need to pass along the successful state `at(hall)` it wants to achieve, the current belief $\mathcal{B}$, and a set of action Λ to the first-principles planner when calling the planner. We choose an offline first-principles planner called Fast-Forward planner[3] and employ the Planning Domain Definition Language (PDDL) [17] for specifying planning problems for the first-principles planner in this concrete example. Due to the syntactic knowledge difference, the transformation of knowledge (e.g. predicate, belief, and action)[4] between BDI and PDDL is required to be conducted to generate PDDL planning problem specification using our PDDL generator[5]. Afterward, the first-principles planner deliberates and generates a plan solution if all possible. Finally, a sequence of actions is returned from the planner to reach the successful state `at(hall)`, denoted as $\sigma =$ `move(door2, lounge, backyard); move(door3, backyard, hall)`. It states the robot can move to the `backyard` through the `door2` first and proceed to the `hall` through the `door3`. The route is depicted pictorially in Fig. 1.

This case study on the blocked plan-body program highlights a number of key benefits offered by the Can(FPP) systems. Compared to classical BDI agent, we are able to improve the scalability of the BDI agent systems to tackle the problems beyond their current reach (e.g. due to incomplete plans and dynamic environment). Compared to a pure FPP, our formal framework ensures maximums reactiveness for most of the subgoals (tracked in the procedural goal intention Γ_{pr}) and only plans on-demand for the pure declarative goals in the declarative goal intention Γ_{de}.

5 Related Work

There have been various planning mechanisms studied in the context of BDI agent systems.

Some researchers focus on the declarative notion of goals as a means to associate FPP. This approach is appealing because a declarative goal gives a description of the desired state for FPP to achieve. One of the first works to look at integrating FPP in a BDI agent system is the Propice-plan framework [18]. It is

[3] https://fai.cs.uni-saarland.de/hoffmann/ff.html.
[4] Due to the lack of space, interested readers are referred to [17] for the full content. We also omit the detailed discussion of the knowledge transformation between BDI and PDDL as it is implementation-dependent.
[5] https://github.com/kevinmcareavey/ppddl.

the combination of the IPP planner [19] and an extended version of the PRS BDI agent system [20]. In Propice-plan, planning occurs only (and always) when no options are available for solving an achievement goal. Another early work is [21] which combines FPP with the IndiGolog agent system [22]. The contribution of [21] is to extend IndiGolog with a classical FPP via its achieve(G) component, where G is a goal formula to achieve. Interestingly, they approached the integration from the direction of translating the planning language, namely ADL, into Golog problem, contrary to our work. However, our approach is based on the typical BDI agent systems while they explore the integration in non-BDI agent architecture (i.e. situation calculus). Notable works on FPP in a BDI setting [10,11] had been ad-hoc approaches without a formal operational semantics for the integration of FPP in BDI. Another important work [23] studies the integration of an online risk-aware planner with a BDI agent. However, it is more concerned with how to calculate risk alongside utility in online planning algorithm instead of integrating online FPP with BDI agent systems in a semantics fashion as we do in this work.

Meanwhile, some researchers tackle the problem from hierarchical task network (HTN) planning perspective. It is revealed in [24] that there are many similarities between HTN planning and BDI agent systems, hence, making them suitable candidates for a principled integration. This principled integration is the semantics of CanPlan [13]. It is an extension of Can with a built-in HTN planning structure which performs a local offline plan search in the pre-defined plan library. In some sense, our work is close to the spirit of CanPlan which provides strong theoretical underpinnings. However, this integration of the HTN planning in CanPlan functions as an advanced plan selection tool which cannot generate new plans.

Some of the works approach the integration of automated planning in BDI agent paradigms by examining the relationship between BDI agent systems and probabilistic planning techniques. For example, [25,26] explore the relationships between certain components of the BDI agent architectures and those in the Markov Decision Processes (MDPs) and the Partially Observable Markov Decision Processes (POMDPs), respectively. More pragmatic approaches to the application of probabilistic planning techniques in BDI agent systems can be found in the works of [12,27]. Although the hybrid BDI and (PO)MDP frameworks provide good insights into the potential integration of probabilistic planning into BDI agent architectures, there is still significant work to be done in modelling and reasoning uncertainty in BDI paradigms beforehand.

6 Conclusions

In this work we proposed a framework with a strong theoretical underpinning for integrating first-principles planning (FPP) within BDI agent systems based on the intrinsic relationship between the two. We introduced a formal operational semantics that incorporates FPP and that lends power to BDI agents when the situation calls for it. We do this by extending the Can language, and extending it with operational semantics to handle a tight integration with FPP. As such, a BDI agent can accomplish the goals beyond its own pre-defined capabilities. We have also established a theorem that the principled integration between FPP and the BDI execution is the one intuitively expected both in offline and online FPP style. We believe the work presented here lays a firm foundation for augmenting the range of behaviours of the agents by expanding the set of BDI plans available to the agent from FPP. More importantly, this paper is a significant step towards incorporating different types of advanced planning techniques into BDI agent systems in a principled manner. For future work, we plan to advance the state-of-art of the hybrid planning BDI agents by proposing a novel BDI plan library evolution architecture to improve the robustness of the BDI agents which operates in a fast-changing environment. To achieve this, we want to introduce the plan library expansion and contraction scheme. The plan library expansion is to adopt new plans generated from the first-principles planner for future reuse. The contraction scheme is accomplished by defining the plan library contraction operator regarding the rationality postulates to remove undesirable plans (e.g. obsolete or incorrect plans).

Acknowledgements. This work has received funding from the EU Horizon 2020 Programme through the DEVELOP project (under grant agreement No. 688127).

References

1. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: Van de Velde, W., Perram, J.W. (eds.) MAAMAW 1996. LNCS, vol. 1038, pp. 42–55. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0031845
2. Winikoff, M., Padgham, L., Harland, J., Thangarajah, J.: Declarative and procedural goals in intelligent agent systems. In: The 8th International Conference on Principles of Knowledge Representation and Reasoning. Morgan Kaufman (2002)
3. Sardina, S., Padgham, L.: A BDI agent programming language with failure handling, declarative goals, and planning. Auton. Agents Multi-Agent Syst. **23**, 18–70 (2011)
4. Winikoff, M.: Jack^{TM}: intelligent agents: an industrial strength platform. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) Multi-Agent Programming. Multiagent Systems, Artificial Societies, and Simulated Organizations (International Book Series), vol. 15, pp. 175–193. Springer, Boston (2005). https://doi.org/10.1007/0-387-26350-0_7
5. Bordini, R.H., HüJomi, J.F., Wooldridge, M.: Programming Multi-Agent Systems in Agentspeak Using Jason, vol. 8. John Wiley & Sons, Chichester (2007)
6. Pokahr, A., Braubach, L., Jander, K.: The Jadex project: programming model. In: Ganzha, M., Jain, L. (eds.) Multiagent Systems and Applications. Intelligent Systems Reference Library, vol. 45, pp. 21–53. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-33323-1_2
7. Jennings, N.R., Bussmann, S.: Agent-based control systems. IEEE Control Syst. **23**, 61–73 (2003)
8. McArthur, S.D., et al.: Multi-agent systems for power engineering applications – Part I: concepts, approaches, and technical challenges. IEEE Trans. Power Syst. **22**, 1743–1752 (2007)

9. Meneguzzi, F., De Silva, L.: Planning in BDI agents: a survey of the integration of planning algorithms and agent reasoning. Knowl. Eng. Rev. **30**, 1–44 (2015)
10. Meneguzzi, F., Luck, M.: Composing high-level plans for declarative agent programming. In: Baldoni, M., Son, T.C., van Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2007. LNCS (LNAI), vol. 4897, pp. 69–85. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77564-5_5
11. De Silva, L., Sardina, S., Padgham, L.: First principles planning in BDI systems. In: The 8th International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, vol. 2, pp. 1105–1112 (2009)
12. Bauters, K., et al.: Probabilistic Planning in agentspeak using the POMDP framework. In: Hatzilygeroudis, I., Palade, V., Prentzas, J. (eds.) Combinations of Intelligent Methods and Applications. SIST, vol. 46, pp. 19–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-26860-6_2
13. Sardina, S., Padgham, L.: Goals in the context of BDI plan failure and planning. In: The 6th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 16–23 (2007)
14. van Riemsdijk, M.B., Dastani, M., Dignum, F., Meyer, J.-J.C.: Dynamics of declarative goals in agent programming. In: Leite, J., Omicini, A., Torroni, P., Yolum, I. (eds.) DALT 2004. LNCS (LNAI), vol. 3476, pp. 1–18. Springer, Heidelberg (2005). https://doi.org/10.1007/11493402_1
15. Keller, T., Eyerich, P.: PROST: probabilistic planning based on UCT. In: The 22nd International Conference on Automated Planning and Scheduling (2012)
16. Hoffmann, J., Nebel, B.: The FF planning system: fast plan generation through heuristic search. J. Artif. Intell. Res. **14**, 253–302 (2001)
17. McDermott, D.: The AIPS-98 planning competition committee. PDDL – The Planning Domain Definition Language. (1998)
18. Despouys, O., Ingrand, F.F.: Propice-Plan: toward a unified framework for planning and execution. In: Biundo, S., Fox, M. (eds.) ECP 1999. LNCS (LNAI), vol. 1809, pp. 278–293. Springer, Heidelberg (2000). https://doi.org/10.1007/10720246_22
19. Koehler, J., Nebel, B., Hoffmann, J., Dimopoulos, Y.: Extending planning graphs to an ADL subset. In: Steel, S., Alami, R. (eds.) ECP 1997. LNCS, vol. 1348, pp. 273–285. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63912-8_92
20. Ingrand, F.F., Georgeff, M.P., Rao, A.S.: An architecture for real-time reasoning and system control. IEEE Expert **7**, 34–44 (1992)
21. Claßen, J., Eyerich, P., Lakemeyer, G., Nebel, B.: Towards an integration of Golog and planning. In: The 20th International Joint Conferences on Artificial Intelligence, pp. 1846–1851 (2007)
22. Sardina, S., Giacomo, G.D., Lespérance, Y., Levesque, H.J.: On the semantics of deliberation in indiGolog-from theory to implementation. Ann. Math. Artif. Intell. **41**, 259–299 (2004)
23. Killough, R., Bauters, K., McAreavey, K., Liu, W., Hong, J.: Risk-aware planning in BDI agents. In: The 8th International Conference on Agents and Artificial Intelligence, pp. 322–329 (2016)
24. de Silva, L., Padgham, L.: A comparison of BDI based real-time reasoning and HTN based planning. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 1167–1173. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30549-1_118
25. Simari, G.I., Parsons, S.: On the relationship between MDPs and the BDI architecture. In: the 5th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1041–1048. ACM (2006)

26. Schut, M., Wooldridge, M., Parsons, S.: On partially observable MDPs and BDI models. In: d'Inverno, M., Luck, M., Fisher, M., Preist, C. (eds.) Foundations and Applications of Multi-Agent Systems. LNCS (LNAI), vol. 2403, pp. 243–259. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45634-1_15
27. Chen, Y., Hong, J., Liu, W., Godoís, L., Sierra, C., Loughlin, M.: Incorporating PGMs into a BDI architecture. In: Boella, G., Elkind, E., Savarimuthu, B.T.R., Dignum, F., Purvis, M.K. (eds.) PRIMA 2013. LNCS (LNAI), vol. 8291, pp. 54–69. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-44927-7_5