Programming Mental State Abduction

Michal Sindlar [*]
Intelligent Systems Group
University of Utrecht
michal@cs.uu.nl

Mehdi Dastani
Intelligent Systems Group
University of Utrecht
mehdi@cs.uu.nl

John-Jules Meyer
Intelligent Systems Group
University of Utrecht
jj@cs.uu.nl

ABSTRACT

Many multi-agent system applications involve software agents that reason about the behavior of other agents with which they interact in cooperation or competition. In order to design and develop those systems, the employed programming languages should provide tools to facilitate the implementation of agents that can perform such reasoning. This paper focuses on BDI-based programming languages and proposes a nonmonotonic reasoning mechanism that can be incorporated into agents, allowing them to reason about observed behavior to infer others' beliefs or goals. In particular, it is suggested that the behavior-generating rules of agents are translated into a nonmonotonic logic programming framework. A formal analysis of the presented approach is provided and it is shown that it has desirable properties.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents—*mental state ascription,nonmonotonic reasoning*

General Terms

Design, Theory

Keywords

Modeling other agents and self, Logic-based approaches and methods, Reasoning (single and multiagent)

1. INTRODUCTION

A fundamental principle of multi-agent systems is that they involve multiple agents, often situated in a (virtual) environment where they interact in cooperation or competition. For implementation of individual software agents in a multi-agent system there exists a range of agent programming languages, such as 2APL, GOAL, Jadex and Jason [5, 9], most of which find their roots in the Belief-Desire-Intention (BDI) paradigm of agency [8, 19]. If such agents have the ability to perceive and interpret the behavior of others, then the principle of mental state ascription comes into focus as a possible explanatory abstraction for reasoning about that behavior. This principle has solid theoretical foundations in terms of the *intentional stance* [10], which is encountered throughout A.I. literature in various guises [6, 7, 13, 21, 4]. However, there exist few principled instantiations specific to BDI-based agent programming of explanatory reasoning about the behavior of other agents. This paper presents such an instantiation, a possible application of which is in computer games where virtual characters are to exhibit believable interaction with other virtual characters. Multiple requirements for character believability (as found by Loyall [17]) are fulfilled by BDI-based software agents (e.g. proactiveness, resource-bounded agency, adaptivity), and some of those which are not (particularly those concerning social interaction) can be met using our approach. This is desirable as it has been shown that characters' believability (and players' enjoyment) increases if characters appear to incorporate mental state ascription into their decision-making [16], and specifically for role-playing games user feedback furthermore indicates that players consider characters to be deficient in this regard [1].

Because the work which is the foundation of this paper is logic-based, and because most agent programming languages utilize logic programming, it is natural that the implementation presented in this paper is a logic program. And since explanatory reasoning is in general defeasible, answer set programming [12] is employed as it is the state-of-the-art programming paradigm for nonmonotonic reasoning. Most systems for answer set programming share their syntax with Prolog, a language which is incorporated into several current agent programming languages (e.g. [9]), making integration of answer set programming natural from a syntactic point of view. In select cases this integration already exists [18].

The work presented in this paper is based on our earlier work on explanatory ascription of mental states to software agents [21], and is presented in the following steps: first, the theoretical underpinnings are briefly explained in Section 2; second, this existing work is formalized in classical logic to lay the foundation for our implementation (Section 3); and third, this implementation is presented and analyzed (Sections 4–6). Sections 7 and 8 conclude with a reflection on related work and our own, respectively.

2. PRELIMINARIES

In this section our work on mental state abduction [21] is restated, in relation to an agent programming language that

[*]This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

contains constructs shared by most state-of-the-art agent programming languages (cf. [5, 9]).

2.1 Agent Programming

This section describes the agent programming language APL used in this paper. The behavior of agents is generated by goal achievement rules of the form $n : \gamma$ `<-` β `|` π, stating that the rule with identifier n is suitable for achieving goal γ if β is believed, in which case the plan π can be selected by the agent. The BNF grammar of rules is given below, where it should be noted that it concerns ground programs (i.e. without variables), which suffices for our purposes.

$$
\begin{array}{lll}
\langle \mathit{library} \rangle & ::= & \langle \mathit{pgrule} \rangle +; \\
\langle \mathit{pgrule} \rangle & ::= & \langle n \rangle \text{“}\texttt{:}\text{”} \langle \mathit{gq} \rangle \text{“}\texttt{<-}\text{”} \langle q \rangle \text{“}\texttt{|}\text{”} \langle \mathit{plan} \rangle; \\
\langle \mathit{gq} \rangle & ::= & \langle \mathit{atom} \rangle \,|\, \langle \mathit{gq} \rangle \text{“}\texttt{and}\text{”} \langle \mathit{gq} \rangle \,|\, \langle \mathit{gq} \rangle \text{“}\texttt{or}\text{”} \langle \mathit{gq} \rangle; \\
\langle q \rangle & ::= & \langle \mathit{literal} \rangle \,|\, \langle q \rangle \text{“}\texttt{and}\text{”} \langle q \rangle \,|\, \langle q \rangle \text{“}\texttt{or}\text{”} \langle q \rangle; \\
\langle \mathit{literal} \rangle & ::= & \langle \mathit{atom} \rangle \,|\, \text{“}\texttt{not}\text{”} \langle \mathit{atom} \rangle \\
\langle \mathit{plan} \rangle & ::= & \langle \mathit{action} \rangle \,|\, \langle \mathit{test} \rangle \,|\, \langle \mathit{seq} \rangle \,|\, \langle \mathit{cond} \rangle \,|\, \langle \mathit{iter} \rangle; \\
\langle \mathit{test} \rangle & ::= & \text{“}\texttt{B(}\text{”} \langle q \rangle \text{“}\texttt{)}\text{”} \,|\, \text{“}\texttt{G(}\text{”} \langle q \rangle \text{“}\texttt{)}\text{”} \,|\, \langle \mathit{test} \rangle \text{“}\texttt{and}\text{”} \langle \mathit{test} \rangle; \\
\langle \mathit{seq} \rangle & ::= & \langle \mathit{plan} \rangle \text{“}\texttt{;}\text{”} \langle \mathit{plan} \rangle; \\
\langle \mathit{cond} \rangle & ::= & \text{“}\texttt{if}\text{”} \langle \mathit{test} \rangle \text{“}\texttt{then}\text{”} \langle \mathit{plan} \rangle \text{“}\texttt{else}\text{”} \langle \mathit{plan} \rangle; \\
\langle \mathit{iter} \rangle & ::= & \text{“}\texttt{while}\text{”} \langle \mathit{test} \rangle \text{“}\texttt{do}\text{”} \langle \mathit{plan} \rangle
\end{array}
$$

The semantics of rule interpretation are omitted for space conservation and also because the above is a theoretical agent programming language. Nevertheless, this language is subsumed by some existing agent programming languages and is strongly related to others, such that a substantiated discussion of rule interpretation can be given.

In most agent programming languages interpretation of rules occurs as part of a *deliberation cycle* (or ‘sense-reason-act’ cycle), in which the applicability of rules is considered in relation to the configuration of the agent (i.e. its beliefs, goals, intentions, etc.). An agent has a library of rules at its disposition, and if a rule is applicable then it can be fired such that the accompanying plan is adopted by the agent and becomes the agent’s active plan. A plan describes behavior, composing primitive and test actions by means of standard programming constructs (sequence, choice, iteration). It is assumed with regard to rule interpretation that 1) the agent executes the actions of its active plan until this plan is finished or dropped (in which case it is not active anymore); 2) the agent executes actions sequentially (i.e. not concurrently); and 3) the agent has at most a single active plan (i.e. no interleaving of actions from different plans).

2.2 Mental State Abduction

Mental state abduction [21] computes a set of explanations for a software agent’s observed behavior, based on knowledge of its rules. It employs the abductive scheme $\{\phi \rightarrow \psi, \psi\} \approx\!\!\!\!\mid\; \phi$, which is further clarified in Section 3.1, in inferring beliefs and goals in relation to rules, plans and observed behavior. In order to do so a relation is established between observed actions and plans, and for this purpose a propositional language and a process language [3] are defined here that allow us to formally describe the rules of the agent program. Those languages are assumed to have a common ground with the agent program; specifically they have shared sets of atomic actions Act and propositions Atom, respectively. The propositional language $\mathcal{L}_0$ and process language $\mathcal{L}_\Pi$ are then defined through $\phi \in \mathcal{L}_0$ and the typical element $\pi \in \mathcal{L}_\Pi$, as follows, given $p \in \mathsf{Atom}$ and $\alpha \in \mathsf{Act}$.

$$
\begin{array}{lll}
\phi & ::= & p \,|\, \neg\phi \,|\, \phi_1 \vee \phi_2 \\
\pi & ::= & \alpha \,|\, \mathbf{B}\phi? \,|\, \mathbf{G}\phi? \,|\, \pi_1;\pi_2 \,|\, \pi_1 + \pi_2 \,|\, \pi^*
\end{array}
$$

The operator ? defines (unobservable) test actions which can be composed by sequential composition, non-deterministic choice, and iteration (;, +, and *, respectively), along with observable actions $\alpha \in \mathsf{Act}$.

It is here assumed that for APL programming rules of the form $n : \gamma$ `<-` β `|` π holds that $n \geq 1$, $\gamma, \beta \in \mathcal{L}_0$ and $\pi \in \mathcal{L}_\Pi$. Just like in [21], we treat such rules as implications for the purpose of abduction. The preconditions γ, β of a rule $n : \gamma$ `<-` β `|` π are considered abducible, and can be abduced if observed actions are related to π. To relate primitive actions to a plan, *observable sequences* which are generated by this plan are considered. Given $\alpha \in \mathsf{Act}$ the language of observables $\mathcal{L}_\Delta$ is defined through its typical element $\delta ::= \alpha \,|\, \delta_1\delta_2$ to consist of sequences of actions, such that the function $OS : \mathcal{L}_\Pi \longrightarrow \wp(\mathcal{L}_\Delta)$ translates complex expressions (which may involve tests and looping or branching constructs) to sets of observable sequences, as follows.

$$
\begin{array}{ll}
OS(\alpha) & = \{\alpha\} \\
OS(\mathbf{B}\phi?) & = \emptyset \\
OS(\mathbf{G}\phi?) & = \emptyset \\
OS(\pi_1;\pi_2) & = OS(\pi_1) \bullet OS(\pi_2) \\
OS(\pi_1 + \pi_2) & = OS(\pi_1) \cup OS(\pi_2) \\
OS(\pi^*) & = \bigcup_{n \in \mathbb{N}} OS(\pi^n), \text{ where } \pi^0 = \mathbf{skip} \;\&\; \pi^{n+1} = \pi;\pi^n
\end{array}
$$

The composition operator $\bullet : \wp(\mathcal{L}_\Delta) \times \wp(\mathcal{L}_\Delta) \longrightarrow \wp(\mathcal{L}_\Delta)$ takes arguments $\Delta_1, \Delta_2 \subseteq \mathcal{L}_\Delta$ and maps them to $\{\delta_1\delta_2 \,|\, \delta_1 \in \Delta_1, \delta_2 \in \Delta_2\}$ if $\Delta_1 \neq \emptyset \,\&\, \Delta_2 \neq \emptyset$, to Δ_1 if $\Delta_2 = \emptyset$, to Δ_2 if $\Delta_1 = \emptyset$, or to $\emptyset$ otherwise. Note that $OS(\mathbf{skip}) = \emptyset$.

In this paper we focus on the case of *complete observation*, in which all actions of an agent are observed. The relation characterizing this condition is the prefix relation on observable sequences $\preccurlyeq \; = \{(\delta, \delta), \; (\delta, \delta\delta') \,|\, \delta, \delta' \in \mathcal{L}_\Delta\}$, because if all actions are observed then an observed sequence must be the prefix of the observable sequence of some plan. Mental state abduction under the assumption of complete observation is then functionally implemented by $\mathsf{msa}_\mathcal{R}$, defined as follows, based on a set $\mathcal{R}$ of APL rules.

$$\mathsf{msa}_\mathcal{R}(\delta) = \{(\gamma, \beta) \,|\, \exists (n : \gamma \texttt{<-} \beta \texttt{|} \pi) \in \mathcal{R} \; \exists \delta' \in OS(\pi) : \delta \preccurlyeq \delta'\}$$

In the next section mental state abduction is cast in the mold of classical abduction, which is introduced in Section 3.1.

3. MENTAL STATE ABDUCTION VIEWED AS CLASSICAL ABDUCTION

This section presents an account of mental state abduction in terms of classical logical abduction, focusing on the relating between rule application on the one hand, and action sequences and the agent’s mental state on the other.

3.1 Classical Abduction

Abduction in classical logic is typically considered in the context of *explanation* [2]. Given a logical theory Θ and an observed fact O, under certain conditions a hypothesis H can be abduced which explains the observation. The fact that H is abduced as explanation for O with respect to Θ is denoted $\Theta, O \mathrel{|\!\approx} H$, and defined as follows in terms of classical entailment $\models$, where $\mathsf{cl}(\Phi)$ denotes the consequential closure of the set Φ; i.e. all facts logically following from Φ.

$$\Theta, O \mathrel{|\!\approx} H \textit{ iff } \Theta \cup H \models O \,\&\, \Theta \not\models O \,\&\, H \not\models O \,\&\, \Theta \cup H \not\models \bot \\ \&\, \neg\exists H' : (\mathsf{cl}(H') \subset \mathsf{cl}(H) \,\&\, \Theta \cup H' \models O)$$

The definition of $\mathrel{|\!\approx}$ varies per context, but the above is often encountered [2]. Note that the last clause expresses the *minimality* of H: there should not exist a hypothesis H' such that the consequential closure of H' is a strict subset of that of H, where H' also accounts for O. Also note that the term 'theory' is used without requiring closure under cl.

An example of abductive explanation which is regularly found in literature [2, 15] goes as follows. Let $\Theta = \{r \rightarrow w, s \rightarrow w\}$ be a logical theory stating that the grass is wet (w) both if it rains (r) and if the sprinklers are on (s). The grass being wet as such is not accounted for by the theory (i.e. $\Theta \not\models w$) but it would be accounted for if it were the case that it rains or that the sprinklers are on, i.e. $\Theta, \{w\} \mathrel{|\!\approx} r$ and $\Theta, \{w\} \mathrel{|\!\approx} s$. Note, though, that $\Theta, \{w\} \not\mathrel{|\!\approx} r \wedge s$ under the above definition of $\mathrel{|\!\approx}$, because the hypothesis $r \wedge s$ is not minimal. Also note that $\mathrel{|\!\approx}$ is nonmonotonic: if it is learned at some point that it rains, such that if $\Theta' = \Theta \cup \{r\}$, then it is evident that $\Theta, \{w\} \mathrel{|\!\approx} s$ and $\Theta \subseteq \Theta'$, but $\Theta', \{w\} \not\mathrel{|\!\approx} s$.

3.2 Translating Rules

As stated earlier, we (informally) treat APL rules of the form $n : \gamma \texttt{<-} \beta \mid \pi$ as implications '$\gamma, \beta \Rightarrow \pi$' for the sake of employing the classical abductive syllogism. However, this is formally not correct because it need not hold that the agent selects plan π if it has goal γ and belief β. The reason for this lies in the fact that rules cannot be fired if the agent has an active plan, even if they are in principle applicable. A more precise implicative treatment of rules of the above form is therefore '$n \Rightarrow \gamma, \beta, \pi$', to be interpreted as stating that the application of the rule implies that the agent has a particular mental state and has selected a particular plan. Thus, 'mental state abduction' is better viewed as 'applied rule abduction', where the abduced rule implies a particular mental state. A logical description of rules can then be provided that allows for treating mental state abduction as a case of classical abduction, which is done in the present section by formulating a logical theory that describes rule application by the agent in relation to its mental state and behavior. It is noteworthy in this respect that a single rule is accompanied by a single plan, but that a single plan can give rise to multiple observable sequences. Specifically, if a plan gives rise to multiple observable sequences then it also gives rise to multiple *computation sequences* [14] (although not necessarily vice versa). Let $CS : \mathcal{L}_\Pi \longrightarrow \wp(\mathcal{L}_\Pi)$ be the function that computes computation sequences of plans, and be defined like the function OS except for

$$CS(\mathbf{B}\phi?) = \{\mathbf{B}\phi?\} \qquad\qquad CS(\mathbf{G}\phi?) = \{\mathbf{G}\phi?\}$$

For technical simplicity, and without loss of generality, it is assumed that individual actions in computation sequences are separated by sequential composition (i.e. the symbol ';') and are therefore in the language $\mathcal{L}_\Pi$. The translation from APL program to logic uses the following predicates:

- $\mathtt{r}(n, c)$: The agent has applied rule with number n and performs the computation sequence identified by c.
- $\mathtt{b}(\psi)$: The agent's belief base entails ψ.
- $\mathtt{g}(\psi)$: The agent's goal base entails ψ.

It is assumed that the background theory Θ is a subset of ground predicate logic, and that the predicates $\mathtt{b}/1$ and $\mathtt{g}/1$ take as argument the result of a function τ defined as follows, where atoms of the set Atom are available as constants.

$$\begin{aligned} \tau(p) &= p, \textit{ if and only if } p \in \mathsf{Atom} \\ \tau(\neg\phi) &= \mathtt{neg}(\tau(\phi)) \\ \tau(\phi \vee \phi') &= \mathtt{disj}(\tau(\phi), \tau(\phi')) \\ \tau(\phi \wedge \phi') &= \mathtt{conj}(\tau(\phi), \tau(\phi')) \end{aligned}$$

The function τ thus maps (ground) APL terms from a logical form to a functional representation, which is left as-is for now. At a later point (in Section 5.1) this functional representation is utilized in order to regain the truth function of logical connectives, but up to then it can simply be regarded as a string that points to a particular belief/goal precondition (i.e. $\tau(\gamma)$ points to γ). It is more convenient, though, to already define and employ the translation now, in order to save space and effort in later sections.

The translation of a set of rules $\mathcal{R}$ of the form $n : \gamma \texttt{<-} \beta \mid \pi$ to a logical theory $\Theta_\mathcal{R}$ is then given below in Formula 1, where ι is a function that assigns a unique numerical identifier to its argument.

$$\begin{aligned} &\forall (n : \gamma \texttt{<-} \beta \mid \pi) \in \mathcal{R}\ \forall \pi' \in CS(\pi) : \\ &\quad \mathtt{r}(n, \iota(\pi')) \rightarrow (\mathtt{g}(\tau(\gamma)) \wedge \mathtt{b}(\tau(\beta))) \in \Theta_\mathcal{R} \end{aligned} \tag{1}$$

In order to abduce the applied rule on grounds of observed actions, action observability must be formalized as part of the theory. This is realized using the following predicate:

- $\mathtt{o}(\alpha, n)$: Action α is observable as the n'th action.

The theory $\Theta_\mathcal{R}$ relates rule application to action observability, and it should be noted in this respect that the 'position' at which an action is observable is reified in instances of the predicate $\mathtt{o}/2$ which are coupled to rule application by means of the relation expressed below.

$$\begin{aligned} &\forall (n : \gamma \texttt{<-} \beta \mid \pi) \in \mathcal{R}\ \forall \pi' \in CS(\pi) : \\ &\quad OS(\pi') = \{\alpha_1 \cdots \alpha_m\} \quad \Longrightarrow \\ &\quad \mathtt{r}(n, \iota(\pi')) \rightarrow (\mathtt{o}(\alpha_1, 1) \wedge \ldots \wedge \mathtt{o}(\alpha_m, m)) \in \Theta_\mathcal{R} \end{aligned} \tag{2}$$

The set of rules $\mathcal{R}$ is here assumed to be fixed, and for any action α_i which is part of some observable sequence, let $i \in \mathbb{N}$ denote that action's position in the sequence. The translation of the set of rules $\mathcal{R}$ into the theory $\Theta_\mathcal{R}$ then allows for abductively inferring instances of $\mathtt{r}/2$ based on observations, which are defined in the next section.

3.3 Observables and Abducibles

The definition of abductive explanation $\mathrel{|\!\approx}$ in Section 3.1 does not explicitly specify the domain of the hypothesis H, which is common in treatises concerning logical abduction. However, in computational approaches to abduction the hypotheses are typically restricted to a set of *abducibles* [15]. This approach is adopted here, and $\mathcal{ABD} = \{\mathtt{r}(n, c) \mid c, n \in \mathbb{N} \,\&\, c, n \geq 1\}$ is introduced as the set of abducibles. Furthermore, the set $\mathcal{OBS} = \{\mathtt{s}(\alpha, n) \mid \alpha \in \mathsf{Act} \,\&\, n \in \mathbb{N} \,\&\, n \geq 1\}$ of *observables* (i.e. possible observations) is introduced, based on the following predicate.

- $\mathtt{s}(\alpha, n)$: The action α is seen as the n'th action.

Because the theory $\Theta_{\mathcal{R}}$ generated from the set $\mathcal{R}$ of APL rules, as defined in the previous section, specifies only the actions which are *observable* (as opposed to actually observed, or *seen*), a relation is made between observation and observability using the operator ω, defined as follows.

$$\omega(\{\mathtt{s}(\alpha, n), \ldots, \mathtt{s}(\alpha', n')\}) = \mathsf{o}(\alpha, n) \wedge \ldots \wedge \mathsf{o}(\alpha', n')$$

The rationale behind the ω-operator is that if some actions have been observed then those actions naturally must be observable, and this latter fact can be utilized to abduce, with respect to some $\Theta_{\mathcal{R}}$, the pair of rule and computation sequence (i.e. some instance of $\mathtt{r}/2$) which accounts for the observability of those particular actions.[1]

In order to treat mental state abduction (Section 2.2) as a case of classical abduction, some further refinements to the background theory must be made. This is illustrated with the following example, in which the set of APL rules $\mathcal{R} = \{1 : p \texttt{<-} p' \mid a; b,\ 2 : q \texttt{<-} q' \mid c; d\}$ and theory $\Theta_{\mathcal{R}}$ is as follows, based on Formulae 1 and 2.

$$\begin{aligned}\Theta_{\mathcal{R}} = \{&\mathtt{r}(1,1) \to (\mathsf{g}(p) \wedge \mathsf{b}(p')), \mathtt{r}(2,2) \to (\mathsf{g}(q) \wedge \mathsf{b}(q')),\\ &\mathtt{r}(1,1) \to (\mathsf{o}(a,1) \wedge \mathsf{o}(b,2)),\ \mathtt{r}(2,2) \to (\mathsf{o}(c,1) \wedge \mathsf{o}(d,2))\}\end{aligned}$$

Observe that for observation $O = \{\mathtt{s}(b,2) \wedge \mathtt{s}(d,2)\}$ holds $\Theta_{\mathcal{R}}, \omega(O) \mathrel{|\!\approx} \{\mathtt{r}(1,1), \mathtt{r}(2,2)\}$, such that application of both rules 1 and 2 and execution of two computation sequences of the corresponding plans is the *minimal* explanation for this observation. This O states different actions to be observed as the second action, whereas our assumption is that observation is sequential such that only a single observation can occur at any moment. For this reason some restrictions are imposed on any *actual observation* $O \subseteq \mathcal{OBS}$, as follows.

$$\begin{aligned}&\forall \alpha \in \mathsf{Act}\ \forall n \in \mathbb{N} :\\ &\quad [(\mathtt{s}(\alpha, n) \in O)\ \&\ (n > 1) \Longrightarrow\\ &\qquad (\exists \alpha' \in \mathsf{Act} : \mathtt{s}(\alpha', n-1) \in O)]\ \&\\ &\quad [\forall \alpha' \in \mathsf{Act} : (\mathtt{s}(\alpha, n), \mathtt{s}(\alpha', n) \in O) \Longrightarrow\\ &\qquad (\alpha = \alpha')]\end{aligned} \tag{3}$$

The constraint of Formula 3 ensure that instances of $\mathtt{s}/2$ in O are numbered consecutively, starting with 1, and that for each n at most a single action is observed.

However, consider $O' = \mathtt{s}(a,1) \wedge \mathtt{s}(d,2)$ and verify it to respect the constraint of Formula 3, yet observe that again $\Theta_{\mathcal{R}}, \omega(O') \mathrel{|\!\approx} \{\mathtt{r}(1,1), \mathtt{r}(2,2)\}$. In order to rule out such explanations, it must be explicitly assumed that the agent's observed behavior stems from a single computation sequence of some plan belonging to a single rule, as follows.

[1]An interesting insight, provided by the effort to reformulate mental state abduction as a case of classical abduction, is that mental states can be treated as *observables*, on par with actions. In the case where actions are the observable input to abductive explanation, rules are abduced which account for those actions. Given an input of a mental state as observable — on grounds of the agent communicating its goals and/or beliefs to the observer, for example — rules are abduced which could be applied by the agent given that particular mental state. And, taking this train of thought yet a station further, nothing withstands having both actions and beliefs/goals as observable input, such that abduction of rules which account for the observed actions, given that particular (fragment of the) agent's mental state, occurs.

$$\begin{aligned}&\forall (m : \gamma \texttt{<-} \beta \mid \pi), (n : \gamma' \texttt{<-} \beta' \mid \pi') \in \mathcal{R}\\ &\forall i \in \{\iota(\pi'') \mid \pi'' \in CS(\pi)\}, j \in \{\iota(\pi''') \mid \pi''' \in CS(\pi')\} :\\ &\quad (m \neq n) \Longrightarrow \neg(\mathtt{r}(m,i) \wedge \mathtt{r}(n,j)) \in \Theta_{\mathcal{R}} \quad \&\\ &\quad (i \neq j) \Longrightarrow \neg(\mathtt{r}(m,i) \wedge \mathtt{r}(n,j)) \in \Theta_{\mathcal{R}}\end{aligned} \tag{4}$$

Given the above, the following holds.

Theorem 1. *Let $\mathcal{R}$ be a set of APL rules, theory $\Theta_{\mathcal{R}}$ the smallest set closed under Formulae 1, 2, and 4. Then $\forall H \subseteq \mathcal{ABD} : \Theta_{\mathcal{R}}, \omega(O) \mathrel{|\!\approx} H \Longrightarrow H$ is a singleton (set).*

Proof. *Consider any two distinct $h, h' \in \mathcal{ABD}$, and let $H \subseteq \mathcal{ABD}$ such that $\Theta_{\mathcal{R}}, \omega(O) \mathrel{|\!\approx} H$. If $\{h, h'\} \subseteq H$ but $\neg\exists(h \to \psi) \in \Theta_{\mathcal{R}}$ or $\neg\exists(h' \to \psi) \in \Theta_{\mathcal{R}}$, then $\Theta_{\mathcal{R}}, \omega(O) \mathrel{|\!\not\approx} H$ because H is not minimal. If $\exists(h \to \psi) \in \Theta_{\mathcal{R}}$ and $\exists(h' \to \psi) \in \Theta_{\mathcal{R}}$, then $\neg(h \wedge h') \in \Theta_{\mathcal{R}}$ and $\Theta_{\mathcal{R}}, \omega(O) \mathrel{|\!\not\approx} H$ because $\Theta_{\mathcal{R}} \cup H \models \bot$. Thus, H cannot contain two distinct elements and must be a singleton $H = \{h\}$ for some $h \in \mathcal{ABD}$.* $\square$

3.4 Proof of Correspondence

In this section correspondence is shown between the functional approach of mental state abduction by means of $\mathsf{msa}_{\mathcal{R}}$, as restated in the previous section, and the classical abductive approach based on the theory $\Theta_{\mathcal{R}}$, as put forward in the current section.

Theorem 2. *Let $\mathcal{R}$ be a set of APL rules, and theory $\Theta_{\mathcal{R}}$ the smallest set closed under Formulae 1, 2, and 4. Then, given $l(\delta) = \{\mathtt{s}(\alpha_1, 1), \ldots, \mathtt{s}(\alpha_n, n)\}$ iff $\delta = \alpha_1 \cdots \alpha_n$,*

$$\begin{aligned}&\forall \delta \in \mathcal{L}_\Delta\, \forall \gamma, \beta \in \mathcal{L}_0 : \qquad (\gamma, \beta) \in \mathsf{msa}_{\mathcal{R}}(\delta) \qquad \Longleftrightarrow\\ &\quad \exists H \subseteq \mathcal{ABD} : \Theta_{\mathcal{R}}, \omega(l(\delta)) \mathrel{|\!\approx} H\ \&\ \Theta_{\mathcal{R}} \cup H \models \mathsf{g}(\tau(\gamma)) \wedge \mathsf{b}(\tau(\beta))\end{aligned}$$

Proof. *($\Rightarrow$) Choose any $\delta \in \mathcal{L}_\Delta$ and $(\gamma, \beta) \in \mathsf{msa}_{\mathcal{R}}(\delta)$, and note that $\exists(i : \gamma \texttt{<-} \beta \mid \pi) \in \mathcal{R} \exists \delta' \in OS(\pi) : \delta \preccurlyeq \delta'$ must hold. Choose any such $i : \gamma \texttt{<-} \beta \mid \pi$ and let $\delta' = \alpha_1 \cdots \alpha_n$, so that $\exists(\mathtt{r}(i,j) \to \mathsf{o}(\alpha_1, 1) \wedge \ldots \wedge \mathsf{o}(\alpha_n, n)) \in \Theta_{\mathcal{R}}$ on grounds of Formula 2. Let $\delta = \alpha_1 \cdots \alpha_m$ and note that $\mathsf{o}(\alpha_1, 1) \wedge \ldots \wedge \mathsf{o}(\alpha_n, n) \to \omega(l(\delta))$, so $\Theta_{\mathcal{R}} \cup \{\mathtt{r}(i,j)\} \models \omega(l(\delta))$ holds. Thus $\Theta_{\mathcal{R}}, \omega(l(\delta)) \mathrel{|\!\approx} \{\mathtt{r}(i,j)\}$, observing that the requirements of $\mathrel{|\!\approx}$ are met and $\{\mathtt{r}(i,j)\} \subseteq \mathcal{ABD}$. On grounds of Formula 1 holds $\exists(\mathtt{r}(i,j) \to \mathsf{g}(\tau(\gamma)) \wedge \mathsf{b}(\tau(\beta))) \in \Theta_{\mathcal{R}}$, such that $\Theta_{\mathcal{R}} \cup \{\mathtt{r}(i,j)\} \models \mathsf{g}(\tau(\gamma)) \wedge \mathsf{b}(\tau(\beta))$.*

($\Leftarrow$) Take some $\delta \in \mathcal{L}_\Delta$ and assume $\exists H \subseteq \mathcal{ABD}\, \exists \gamma, \beta \in \mathcal{L}_0 : \Theta_{\mathcal{R}}, \omega(l(\delta)) \mathrel{|\!\approx} H \& \Theta_{\mathcal{R}} \cup H \models \mathsf{g}(\tau(\gamma)) \wedge \mathsf{b}(\tau(\beta))$. It has been proven in Theorem 1 that H must be a singleton, say $H = \{\mathtt{r}(i,j)\}$ for some $i, j \geq 1$. Then $\exists(\mathtt{r}(i,j) \to \mathsf{g}(\tau(\gamma)) \wedge \mathsf{b}(\tau(\beta))) \in \Theta_{\mathcal{R}}$ such that $\exists(i : \gamma \texttt{<-} \beta \mid \pi) \in \mathcal{R}$, because $\Theta_{\mathcal{R}}$ is the smallest set closed under Formula 1, 2 and 4. Because it is given that $\Theta_{\mathcal{R}}, \omega(l(\delta)) \mathrel{|\!\approx} \{\mathtt{r}(i,j)\}$ it must be the case that $\exists(\mathtt{r}(i,j) \to \psi) \in \Theta_{\mathcal{R}}$ such that $\psi \to \omega(l(\delta))$. Let $\psi = \mathsf{o}(\alpha_1, 1) \wedge \ldots \wedge \mathsf{o}(\alpha_n, n)$, and observe that $\exists \pi' \in CS(\pi) : \iota(\pi') = j$ such that $OS(\pi') = \{\alpha_1 \cdots \alpha_n\}$ must then be the case. Let $\delta = \alpha_1 \cdots \alpha_m$ with $m \leq n$, such that $\omega(l(\delta)) = \mathsf{o}(\alpha_1, 1) \wedge \ldots \wedge \mathsf{o}(\alpha_m, m)$ and, because any output of the function l respects the constraint of Formula 3, it holds that $\delta \preccurlyeq \alpha_1 \cdots \alpha_n$. If that is the case, then $(\gamma, \beta) \in \mathsf{msa}_{\mathcal{R}}(\delta)$. $\square$

4. IMPLEMENTATION

This section presents the implementation of a logic program, based on the logical theory presented in Section 3, assuming use of the grounder Lparse [22] and the solver clasp [11], winner of several recent competitions.

4.1 Answer Set Programming

ASP (answer set programming, or Answer Set Prolog [12]) is a logic programming language with answer set semantics, whose main programming construct are rules of the form

$$l_i \;:-\; l_{i+1}, \ldots, l_j, \texttt{not}\; l_{j+1}, \ldots, \texttt{not}\; l_k.$$
$$m\{l_1, \ldots, l_i\}n \;:-\; l_{i+1}, \ldots, l_j, \texttt{not}\; l_{j+1}, \ldots, \texttt{not}\; l_k.$$

where each l denotes a literal (i.e. an atom or its strong negation), l_i or $m\{l_1, \ldots, l_i\}m$ is the *head* and $l_{i+1}, \ldots, \texttt{not}\; l_k$ the *body* of the rule. Negation in literals $-p$ is referred to as *strong negation*, and negation in *extended literals* $\texttt{not}\; l$ as *default negation*. Rules of the above form can be interpreted (informally) as stating that the head should be satisfied if the body is, where the head is either a single literal or a *choice literal* of the form $m\{l_1, \ldots, l_i\}n$ stating that a minimum of m and maximum of n literals of $\{l_1, \ldots, l_i\}$ should be in candidate answer sets if the body of the rule is satisfied [22]. If the head of a rule is omitted then it is a *constraint rule*, which (informally) states that a candidate answer set which satisfies the body should be discarded. A rule with an empty body is called a *fact* and is always satisfied.

The crux of ASP semantics is the generation of *minimal sets* that satisfy a program [12]; cf. the minimality criterion in the definition of $\approx\!\!\mid$. Essentially, those sets are models of the logic program, and if every answer set of a program $\mathcal{P}$ satisfies some ϕ it is written $\mathcal{P} \models \phi$. The above introduction to ASP is, by necessity, incomplete and superficial. However, literature abounds with practical and theoretical expositions on ASP: a concise and recent overview with pointers to further reading is given by Gelfond [12].

4.2 From Theory to Program

The logical theory $\Theta_\mathcal{R}$ presented in Section 3 can be translated to a logic program quite straightforwardly. First of all, implications occurring in the theory are translated as choice rules, stating that each of the conjoined literals in the consequent of the implication should be in the answer set if the antecedent is. Translation to the program $\mathcal{P}_\mathcal{R}$ is then as follows, where $\Theta_\mathcal{R}$ is a theory based on some set of rules $\mathcal{R}$, i.e. the smallest set closed under Formulae 1, 2, and 4.

$$\begin{aligned} \phi \to (\psi_1 \land \ldots \land \psi_n) \in \Theta_\mathcal{R} \quad &\Longrightarrow \\ n\{\psi_1, \ldots, \psi_n\}n \;:-\; \phi. \;\in \mathcal{P}_\mathcal{R} & \end{aligned} \tag{5}$$

The translation on grounds of Formula 5 is illustrated below, given $\mathcal{R} = \{1 : p \texttt{<-} p' \mid a; b,\ 2 : q \texttt{<-} q' \mid c; d\}$ and $\Theta_\mathcal{R}$ as in Section 3.3, such that program $\mathcal{P}_\mathcal{R}$ is as follows.

$$2\{\texttt{g}(p), \texttt{b}(p')\}2 \;:-\; \texttt{r}(1,1).$$
$$2\{\texttt{o}(a,1), \texttt{o}(b,2)\}2 \;:-\; \texttt{r}(1,1).$$
$$2\{\texttt{g}(q), \texttt{b}(q')\}2 \;:-\; \texttt{r}(2,2).$$
$$2\{\texttt{o}(c,1), \texttt{o}(d,2)\}2 \;:-\; \texttt{r}(2,2).$$

This program as such does not yet implement mental state abduction under complete observation, though, because there is neither mention of abducibles nor of observables.

4.2.1 Abducibles

In contrast to Section 3, which takes a constructive approach in the sense that a hypothesis H is abduced *if it meets* the requirements of the relation $\approx\!\!\mid$, the implementation takes a deconstructive approach in the sense that each hypothesis is considered as possible explanation and ruled out *if it does not meet* those requirements. Consideration of abducibles as possible explanations is reflected in program $\mathcal{P}_\mathcal{R}$ by the following relation with the theory $\Theta_\mathcal{R}$.

$$\begin{aligned} \Theta_\mathcal{R} = \{&\texttt{r}(m,i) \to \psi, \ldots, \texttt{r}(n,j) \to \psi', \\ &\neg(\texttt{r}(m,i) \land \texttt{r}(n,j)), \ldots\} \\ &\quad\Longrightarrow \quad 1\{\texttt{r}(m,i), \ldots, \texttt{r}(n,j)\}1. \;\in \mathcal{P}_\mathcal{R} \end{aligned} \tag{6}$$

Informally, this means that single distinct instances of $\texttt{r}/2$ are considered as explanations. Theorem 1 shows that an abduced explanation H must be a singleton subset of the abducibles $\mathcal{ABD}$ because of the constraint of Formula 4, which is reflected in the numerical bounds on the choice literal $1\{\texttt{r}(m,i), \ldots, \texttt{r}(n,j)\}1$ stating that only a single instance of $\texttt{r}/2$ is considered as explanation. The condition under which some candidate answer sets are discarded is implemented in the following section in relation to observables.

4.2.2 Observables

The translation from theory to program discussed in the previous section shows that instances of $\texttt{o}/2$ are entailed on grounds of instances of $\texttt{r}/2$. In Section 3.3 it was explained how abduction takes place on grounds of a set $O \subseteq \mathcal{OBS}$ of observations, respecting the constraints of Formula 3. The predicate $\texttt{s}/2$ was defined to denote observed (seen) actions, and instances of this predicate are assumed to be *facts* in the answer set. This is achieved in relation to observations as follows, where $\mathcal{P}_\mathcal{R}$ is the program derived from theory $\Theta_\mathcal{R}$ (the smallest set closed under Formulae 1, 2 and 4 in relation to a set of APL rules $\mathcal{R}$), and $O \subseteq \mathcal{OBS}$ is an observation respecting the constraint of Formula 3.

$$\forall \texttt{s}(\alpha, n) \in O: \quad \texttt{s}(\alpha, n). \;\in \mathcal{P}_\mathcal{R} \tag{7}$$

Note that the constraints of Formula 3 could be implemented as constraints in the answer set program as well, but that this is unnecessary if instances of $\texttt{s}/2$ are derived from O.

As said, the implementation presented in this paper takes a deconstructive approach by ruling out invalid candidate answer sets. Elimination of candidate answer sets takes place on grounds of the single following constraint, which expresses that a candidate answer set with one or more actions that have been seen but are not deemed observable at that particular step, should be discarded.

$$:-\; \texttt{s}(A,T), \texttt{not}\; \texttt{o}(A,T). \tag{8}$$

In Section 3.3 the function ω was defined to consider observed actions in terms of their observability for the sake of abduction; in effect, Formula 8 is the (deconstructive) counterpart of that function.

Theorem 3. *Let $\mathcal{R}$ be a set of APL rules, $\Theta_\mathcal{R}$ the smallest set closed under Formulae 1, 2, and 4, $O \subseteq \mathcal{OBS}$ a non-empty observation, and program $\mathcal{P}_\mathcal{R}$ as derived from $\Theta_\mathcal{R}$ and O with Formulae 5, 6, 7, and 8. Then*

$$\forall H \subseteq \mathcal{ABD}: \qquad (\Theta_\mathcal{R}, \omega(O) \approx\!\!\mid H) \quad \Longleftrightarrow$$
$$\mathcal{P}_\mathcal{R} \textit{ has an answer set } S\textit{, such that } S \models \bigwedge H \land \bigwedge O$$

Proof. *($\Rightarrow$) Assume the antecedent to be the case, and observe that for each possibly abducible H a candidate answer set exists on grounds of Formula 6. Let $H = \{h\}$, and note that if $\Theta_\mathcal{R}, \omega(O) \approx\!\!\mid H$ then $\exists(h \to \psi) \in \Theta_\mathcal{R}$ such that*

$\psi \to \omega(O)$. Note that, on grounds of Formula 5, ψ is represented in the candidate answer set S for which $\bigwedge\{h\} \in S$, and, since $\psi \to \omega(O)$, that Formula 8 does not rule out S, which means that S is a valid answer set of $\mathcal{P}_{\mathcal{R}}$.

($\Leftarrow$) Assume the antecedent to be the case, and note that, on grounds of Formula 6, a single $h \in \mathcal{ABD}$ is in the answer set S. Because S is not ruled out on grounds of Formula 8, it holds that this constraint does not apply. Thus, if $O = \{\mathtt{s}(\alpha_1, 1), \ldots, \mathtt{s}(\alpha_m, m)\}$, then on grounds of Formula 5 holds $\exists(n\{\mathtt{o}(\alpha_1, 1), \ldots, \mathtt{o}(\alpha_n, n)\}n \ :-\ h.) \in \mathcal{P}_{\mathcal{R}}$ for some $m \leq n$. If that is the case, then $(h \to \mathtt{o}(\alpha_1, 1), \ldots, \mathtt{o}(\alpha_n, n)) \in \Theta_{\mathcal{R}}$ must hold, and $\Theta_{\mathcal{R}}, \omega(O) \approx\{h\}$. □

COROLLARY 1. *Given the conditions of Theorem 3,*

$$\forall H \subseteq \mathcal{ABD}\,\forall \phi, \phi' \in \mathcal{L}_0 :$$
$$(\Theta_{\mathcal{R}}, \omega(O) \approx H \ \&\ \Theta_{\mathcal{R}} \cup H \models \mathtt{g}(\tau(\phi)) \land \mathtt{b}(\tau(\phi'))) \iff$$
$$\mathcal{P}_{\mathcal{R}} \text{ has an answer set } S, \text{ s.t. } S \models \mathtt{g}(\tau(\phi)) \land \mathtt{b}(\tau(\phi'))$$

PROOF SKETCH. *Observe that H entails a single instance of $\mathtt{b}/1$ and $\mathtt{g}/1$ in both $\Theta_{\mathcal{R}}$ and S (also cf. Theorem 3).* □

COROLLARY 2.

$$\forall \delta \in \mathcal{L}_\Delta\,\forall \gamma, \beta \in \mathcal{L}_0 : \qquad \exists(\gamma, \beta) \in \mathsf{msa}_{\mathcal{R}}(\delta) \qquad \iff$$
$$\mathcal{P}_{\mathcal{R}} \text{ has an answer set } S, \text{ s.t. } S \models \mathtt{g}(\tau(\gamma)) \land \mathtt{b}(\tau(\beta))$$

PROOF. *From Theorems 2 and 3, and Corollary 1.* □

5. ASCRIPTION OF MENTAL STATES

This section builds upon previous sections, focusing on ascription of particular goals and beliefs.

5.1 Ascription of Rule Preconditions

In Section 3.2 the function τ was defined to translate APL terms — consisting of atoms composed by means of conjunction, disjunction, or negation — to a logical theory. Terms were translated to a functional representation, and given as arguments to predicates $\mathtt{g}/1$ and $\mathtt{b}/1$. The truth conditions of those connectives are lost in this translation, but they can be regained by decomposition of the functional arguments to constants, as shown below. It should hereby be noted that the LPARSE grounder requires particular syntactic conditions to hold with respect to the program, such that recursion cannot be used the same way as in, say, Prolog; for our implementation this means that decomposition must be specified for each level of nesting. This is not problematic, though, since the translation derives from some fixed set of APL rules for which the maximum level of nesting is known. Below are the ASP rules for decomposition of $\mathtt{conj}/2$ and $\mathtt{disj}/2$ based on the semantics of $\land$ and $\lor$, as well as elimination of double negation and DeMorgan's laws. It is here assumed that the predicate $\mathtt{form}/1$ denotes that its argument is a valid formula, and it should be noted that the definition below is given for $\mathtt{b}/1$ but is likewise for $\mathtt{g}/1$.

$$\begin{aligned}
&1\{\mathtt{b}(F1), \mathtt{b}(F2)\}2 \ :-\ \mathtt{form}(F1; F2), \mathtt{b}(\mathtt{disj}(F1, F2)).\\
&2\{\mathtt{b}(F1), \mathtt{b}(F2)\}2 \ :-\ \mathtt{form}(F1; F2), \mathtt{b}(\mathtt{conj}(F1, F2)).\\
&\mathtt{b}(F) \ :-\ \mathtt{form}(F), \mathtt{b}(\mathtt{neg}(\mathtt{neg}(F))).\\
&\mathtt{b}(\mathtt{disj}(\mathtt{neg}(F1), \mathtt{neg}(F2))) \ :-\\
&\qquad \mathtt{form}(F1; F2), \mathtt{b}(\mathtt{neg}(\mathtt{conj}(F1, F2))).\\
&\mathtt{b}(\mathtt{conj}(\mathtt{neg}(F1), \mathtt{neg}(F2))) \ :-\\
&\qquad \mathtt{form}(F1; F2), \mathtt{b}(\mathtt{neg}(\mathtt{disj}(F1, F2))).
\end{aligned} \tag{9}$$

As expected, in case of conjunction of two formulae, both conjuncts should be satisfied. The logical connective $\lor$ means that either or both of two disjuncts should be satisfied; this interpretation is adhered to here in decomposition of `disj`. If a criterion of minimality is maintained then it could alternatively be stated that either single disjunct must be satisfied in order for the disjunction to be satisfied (i.e. $1\{\ldots\}1$).

It is recognized in the literature that a drawback of ASP is that it is not well suited for reasoning with complex logical formulae [12]. The approach sketched above is therefore not computationally efficient if APL terms are complex, but because having single instances of $\mathtt{b}/1$ and $\mathtt{g}/1$ that point to particular β and γ does allow for straightforward proofs in preceding sections, this approach is preferred for technical simplicity. An alternative approach, left unexplored here because of lack of space, is to perform translation of compound formulae outside of ASP by bringing APL terms to a normal form and translating this directly to ASP representation.

5.1.1 Negation

In translation of negated atoms it should be noted that answer set programming offers two kinds of negation: default negation and strong (or classical) negation [12]. In principle, both can be used for interpretation of the function symbol `neg` where it pertains to constants. The informal semantics of default negation of p are that `not` p is the case in absence of the atom p, whereas strong negation of p holds true only if the literal $-p$ is in the answer set. Because presence of $\mathtt{b}(\mathtt{neg}(F))$ or $\mathtt{g}(\mathtt{neg}(F))$, for some constant F, in the answer set occurs on grounds of evidence (observed actions), the use of strong negation is warranted for expressing that F is known not to be in the agent's belief/goal base; even if APL utilizes negation-as-failure (cf. [9]). Since ultimately our interest is in facts which are, or are not, ascribed to the agent as goals or beliefs, the predicates $\mathtt{bel}/1$ and $\mathtt{goal}/1$ are introduced which in contrast to $\mathtt{b}/1$ and $\mathtt{g}/1$ accept only fluents as argument (the term 'fluent' is preferred over 'constant' because valuation is fixed using these predicates).

$$\begin{aligned}
\mathtt{bel}(F) &\ :-\ \mathtt{fluent}(F), \mathtt{b}(F).\\
-\mathtt{bel}(F) &\ :-\ \mathtt{fluent}(F), \mathtt{b}(\mathtt{neg}(F)).
\end{aligned} \tag{10}$$

Similarly, $\mathtt{goal}/1$ is defined in relation to $\mathtt{g}/1$. Thus, observed actions warrant ascription of some fact not being the agent's goal or belief, opposed to warranting mere absence of ascription that it is. The predicates $\mathtt{bel}/1$ and $\mathtt{goal}/1$ denote fluents being (or, if negated, being not) the observed agent's belief or goal, respectively.

5.2 Plan-Based Ascription

The function *CS* was mentioned in Section 3.2, and defined to generate computation sequences and preserve test actions (as opposed to *OS*). Knowledge of successful test actions gives information about the agent's mental state, which can be informally characterized as follows: if the agent is presumed to have applied some rule and is observed to perform some action, then if this observed action is preceded by a test action, this test must have been successful.

Incorporating presumption of successful test actions into ascription requires a notion of dynamics, because tests are performed in a state which may change as a result of actions. Consider, by means of example, a plan that states that if a certain device is believed to be on, then it should be switched off. The test which succeeds before the action of switching off the device might not succeed after performing the action. For this reason a *state argument* is added to the predicates $\mathtt{b}/1$ and $\mathtt{g}/1$. It should be noted in this regard that all occurrences of $\mathtt{b}/1$ and $\mathtt{g}/1$ in Section 4 pertain to the abduced *preconditions* of the observed agent's rule, such that those hold in the state preceding all observed actions. This state is given the number 0, such that all $\mathtt{b}(F)$ and $\mathtt{g}(F)$ in Section 4 are extended to $\mathtt{b}(F,0)$ and $\mathtt{g}(F,0)$.

The expressions in Formula 9 and 10 describe general relations, such that occurrences of $\mathtt{b}/1$ and $\mathtt{g}/1$ (and, correspondingly, of $\mathtt{bel}/1$ and $\mathtt{goal}/1$) in those expressions can simply be extended with a state variable S, in regard to which the predicate $\mathtt{st}/1$ is defined to refer to valid states. Instances of this predicate can be derived from $\mathtt{o}/2$ by means of rules, or specified as facts. For example, in Formula 10 $\mathtt{bel}(F)\ :-\ \mathtt{fluent}(F), \mathtt{b}(F)$ then becomes

$$\mathtt{bel}(F,S)\ :-\ \mathtt{fluent}(F), \mathtt{st}(S), \mathtt{b}(F,S).$$

The above formulation employing a state argument is strongly reminiscent of the *situation calculus* [20], the main difference being that $\mathtt{bel}/2$ and $\mathtt{goal}/2$ refer to fluents ascribed as belief or goal to an observed agent, whereas in approaches based on the situation calculus the predicate $\mathtt{holds}/2$ refers to an agent's own beliefs about its environment.

Ascription on grounds of computation sequences is then implemented as in Formula 11, which captures the informal notion mentioned in the first paragraph of this section and extends it to observed sequences.

$$\begin{aligned}
&\forall (n : \gamma \texttt{<-} \beta \mid \pi) \in \mathcal{R} \\
&\forall \alpha_1, \ldots, \alpha_n, \phi_1?, \ldots, \phi_k?, \pi' \in \mathcal{L}_\Pi : \\
&\quad [\ (\pi' \in CS(\pi)\ \&\ OS(\pi') = \{\alpha_1 \cdots \alpha_i \cdots \alpha_n\}\ \& \\
&\quad\quad \pi' = \cdots; \alpha_{i-1}; \phi_1?; \cdots; \phi_k?; \alpha_i; \cdots)\ \Longrightarrow \\
&\quad\quad\quad (\ k\{\tau'(\phi_1, i-1), \ldots, \tau'(\phi_k, i-1)\}k\ :- \\
&\quad\quad\quad\quad \mathtt{r}(n, \iota(\pi')), \mathtt{s}(\alpha_1, 1), \ldots, \mathtt{s}(\alpha_i, i).\ )\ \in \mathcal{P}_\mathcal{R}\ ]
\end{aligned} \tag{11}$$

The function τ' is required because plans, as defined in Section 2.1, contain test actions on expressions $\phi \in \mathcal{L}_0$ prefixed by either $\mathbf{B}$ or $\mathbf{G}$. A test action $\mathbf{B}\phi?$ is evaluated with respect to the agent's beliefs and $\mathbf{G}\phi?$ with respect to its goals, and such tests can be understood as belief/goal introspection. Translation through τ' is therefore as follows.

$$\tau'(\mathbf{B}\phi, n) = \mathtt{b}(\tau(\phi), n) \qquad \tau'(\mathbf{G}\phi, n) = \mathtt{g}(\tau(\phi), n)$$

This approach has the following useful property.

PROPOSITION 1 (PROVEN FOR $\mathbf{B}$, LIKEWISE FOR $\mathbf{G}$.). *Computation sequences with mutually inconsistent tests on conjoined literals give rise to inconsistent answer sets.*

PROOF. *Given* $\mathsf{Lit} = \{p, \neg p \mid p \in \mathsf{Atom}\}$, *let* $\phi = \bigwedge \Phi, \psi = \bigwedge \Psi$ *for some* $\Phi, \Psi \subseteq \mathsf{Lit}$. *Furthermore let* $(n : \gamma \texttt{<-} \beta \mid \pi) \in \mathcal{R}$ *be such that* $\pi' = \cdots; \alpha_{i-1}; \mathbf{B}\phi?; \cdots; \mathbf{B}\psi?; \alpha_i; \cdots \in CS(\pi)$, *where* $\iota(\pi') = c$. *Assume that* $\{\phi, \psi\} \models \bot$, *such that for some* $q \in \mathsf{Lit}$ *holds* $\phi \rightarrow q$ *and* $\psi \rightarrow \neg q$, *and observe that on grounds of Formulae 9, 10, and 11 it holds that every answer set satisfying* $\mathtt{r}(n,c)$ *and* $\mathtt{s}(\alpha_1, 1), \ldots, \mathtt{s}(\alpha_i, i)$ *must satisfy* $\mathtt{bel}(q, i-1)$ *and* $-\mathtt{bel}(q, i-1)$, *and be inconsistent.* $\Box$

Proposition 1 states that the observer program which incorporates inference on grounds of presumed test actions, along the lines of Formula 11, reflects the fact that observed actions could have not been generated by a sequence which has unsatisfiable tests preceding those actions.

6. EXAMPLE

Throughout this paper short examples have been used to illustrate certain sections. In this section a more elaborate example is given, which has been implemented based on the approach presented in this paper using the grounder LPARSE [22] and the solver CLASP [11]. The example implements a (fictional) virtual character from a life-simulation game like THE SIMS that has the following two rules, in the APL syntax of Section 2.1.

```
seen_movie <- playing_movie |
  enter_mall; if B(not have_cash) then withdraw_cash
  else skip; watch_movie
have_book  <- good_book or not playing_movie |
  enter_mall; if B(not have_cash) then withdraw_cash
  else skip; buy_book
```

Based on the approach sketched in this paper, those rules give rise to the following fragment of the answer set program $\mathcal{P}_\mathcal{R}$ of the observer in relation to Formulae 5, 6, and 8; abbreviations should be evident.

$$\begin{aligned}
2\{\mathtt{g}(\mathtt{s\_m},0), \mathtt{b}(\mathtt{p\_m},0)\}2\ &:-\ \mathtt{r}(1,1). \\
2\{\mathtt{g}(\mathtt{s\_m},0), \mathtt{b}(\mathtt{p\_m},0)\}2\ &:-\ \mathtt{r}(1,2). \\
3\{\mathtt{o}(\mathtt{e\_m},1), \mathtt{o}(\mathtt{w\_c},2), \mathtt{o}(\mathtt{w\_m},3)\}3\ &:-\ \mathtt{r}(1,1). \\
2\{\mathtt{o}(\mathtt{e\_m},1), \mathtt{o}(\mathtt{w\_m},2)\}2\ &:-\ \mathtt{r}(1,2). \\
2\{\mathtt{g}(\mathtt{h\_b},0), \mathtt{b}(\mathtt{disj}(\mathtt{g\_b},\mathtt{neg}(\mathtt{p\_m})),0)\}2\ &:-\ \mathtt{r}(2,3). \\
2\{\mathtt{g}(\mathtt{h\_b},0), \mathtt{b}(\mathtt{disj}(\mathtt{g\_b},\mathtt{neg}(\mathtt{p\_m})),0)\}2\ &:-\ \mathtt{r}(2,4). \\
3\{\mathtt{o}(\mathtt{e\_m},1), \mathtt{o}(\mathtt{w\_c},2), \mathtt{o}(\mathtt{b\_b},3)\}3\ &:-\ \mathtt{r}(2,3). \\
2\{\mathtt{o}(\mathtt{e\_m},1), \mathtt{o}(\mathtt{b\_b},2)\}2\ &:-\ \mathtt{r}(2,4). \\
1\{\mathtt{r}(1,1), \mathtt{r}(1,2), \mathtt{r}(2,3), \mathtt{r}(2,4)\}1&. \\
:-\ \mathtt{s}(A,T), \mathtt{not}\ \mathtt{o}(A,T)&.
\end{aligned}$$

Based on Formula 11, the rest of the program is as follows.

$$\begin{aligned}
1\{\mathtt{b}(\mathtt{neg}(\mathtt{h\_c}),1)\}1\ &:-\ \mathtt{r}(1,1), \mathtt{s}(\mathtt{e\_m},1), \mathtt{s}(\mathtt{w\_c},2). \\
1\{\mathtt{b}(\mathtt{h\_c},1)\}1\ &:-\ \mathtt{r}(1,2), \mathtt{s}(\mathtt{e\_m},1), \mathtt{s}(\mathtt{w\_m},2). \\
1\{\mathtt{b}(\mathtt{neg}(\mathtt{h\_c}),1)\}1\ &:-\ \mathtt{r}(2,3), \mathtt{s}(\mathtt{e\_m},1), \mathtt{s}(\mathtt{w\_c},2). \\
1\{\mathtt{b}(\mathtt{h\_c},1)\}1\ &:-\ \mathtt{r}(2,4), \mathtt{s}(\mathtt{e\_m},1), \mathtt{s}(\mathtt{b\_b},2).
\end{aligned}$$

Let $\mathcal{P}' = \mathcal{P} \cup \{\mathtt{s}(\mathtt{e\_m},1)\}$. Focusing on the predicates $\mathtt{bel}/2$ and $\mathtt{goal}/2$, note that the program $\mathcal{P}'$ has answer sets with $S = \{\mathtt{goal}(\mathtt{s\_m},0), \mathtt{bel}(\mathtt{p\_m},0)\}$ on grounds of explanations $\mathtt{r}(1,1)$ and $\mathtt{r}(1,2)$, and $S' = \{\mathtt{goal}(\mathtt{h\_b},0), \mathtt{bel}(\mathtt{g\_b},0)\}$, $S'' = \{\mathtt{goal}(\mathtt{h\_b},0), -\mathtt{bel}(\mathtt{p\_m},0)\}$, as well as $S' \cup S''$ on grounds of $\mathtt{r}(2,3)$ and $\mathtt{r}(2,4)$. Now let $\mathcal{P}'' = \mathcal{P}' \cup \{\mathtt{s}(\mathtt{w\_m},2)\}$ and observe that $\mathcal{P}'' \models \mathtt{goal}(\mathtt{s\_m},0) \wedge \mathtt{bel}(\mathtt{p\_m},0) \wedge \mathtt{bel}(\mathtt{h\_c},1)$ because $\mathcal{P}'' \models \mathtt{r}(1,2)$. The example thus illustrates the theory of preceding sections in relation to specific atoms being ascribed as (not) the agent's belief/goal, and shows conclusiveness of the observer after observation of two actions.

7. RELATED WORK

The approach presented in this paper can be categorized in the area of plan/intention recognition. Although this traditionally has been an active area of A.I. research, there has been little work *specific to agent programming.* An exception is that of Goultiaeva & Lespérance [13], who present a formal model meant for inclusion in the ConGolog programming language, which is based on the situation calculus. The approach focuses on the procedural aspect of incrementally matching observed behavior to an annotated library of plans.

An important difference with the work in this paper is the fact that our approach uses a general logical abstraction of plans and it is therefore not restricted to a particular agent programming language, except for the fact that it supposes the availability of an ASP interpreter. Furthermore, our work utilizes a mature automated (nonmonotonic) reasoning paradigm, and, last but not least, it is difficult to see how the work of Goultiaeva & Lespérance could extend to incomplete observation (i.e. missing actions), whereas our work is founded on an approach that handles this case. Of further interest is the work of Baral et al. [4] because it uses ASP to model agents knowledge about others' knowledge, albeit using a radically different approach. Apart from logic-based approaches such as the above, there exists a multitude of statistical approaches (see [7]), including some in the game domain. Such approaches require significant amounts of run-time data as input, though, and are not BDI-specific.

8. CONCLUSION AND FUTURE RESEARCH

This paper presents an answer set programming implementation of mental state ascription based on observed primitive actions of a BDI-based agent. The approach is based on earlier work ([21], cf. Section 2.2), which is reformulated here in terms of abduction in classical logical in Section 3. This logical theory gives rise to an implementation in Section 4 which utilizes the suitability of ASP for nonmonotonic reasoning, and is expanded in Section 5 to incorporate ascription based on test actions the observed agent can be presumed to have performed, hereby employing concepts of the situation calculus. Formal proof is given of useful and interesting properties of our approach.

It should be noted that for space and simplicity the implementation presented here assumes complete observation, meaning that all of the agent's actions are observed. In [21] we also formalized *incomplete observation* by means of additional structural relations which allow 'gaps' to occur in matching observed to observable sequences. By implementing those relations our implementation can be generalized to cases of incomplete observation as well. Future research can furthermore focus on exploring the relation with the situation calculus, as indicated in Section 5.2, which would furthermore open up possibilities for implementation of an agent that reasons about its own environment and mental state in relation to that ascribed to others. Existing work on ASP for dynamic domains can then be of use [12]. Furthermore, prediction of agents' actions can be considered, and, last but not least, investigating the practice of integrating logic programming interpreters into demanding applications such as games is mandatory for (industrial) deployment.

9. REFERENCES

[1] N. Afonso and R. Prada. Agents that relate: Improving the social believability of non-player characters in role-playing games. In *Proc. of 7th Conf. on Entertainment Comp. (ICEC)*, pages 34–45, 2008.

[2] A. Aliseda-Llera. *Seeking Explanations: Abduction in Logic, Philosophy of Science, and Artificial Intelligence.* PhD thesis, Univ. of Amsterdam, 1997.

[3] J. Baeten and W. Weijland. *Process Algebra.* Cambridge University Press, 1990.

[4] C. Baral, G. Gelfond, E. Pontelli, and T. C. Son. Using answer set programming to model multi-agent scenarios involving agents' knowledge about others' knowledge. In *Proceedings of 9th Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 259–266, 2010.

[5] R. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Tools and Applications.* Springer, 2009.

[6] R. I. Brafman and M. Tennenholtz. Belief ascription and mental-level modelling. In *Proceedings of the 4th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pages 87–98, 1994.

[7] S. Carberry. Techniques for plan recognition. *User Modeling & User-Ad. Interaction*, 11(1-2):31–48, 2001.

[8] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[9] M. Dastani. 2APL: A practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16:214–248, 2008.

[10] D. Dennett. *The Intentional Stance.* MIT Press, 1987.

[11] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. CLASP: A conflict-driven answer set solver. In *Proc. of the Ninth Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 260–265, 2007.

[12] M. Gelfond. Answer sets. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, pages 285–316. Elsevier Science, 2008.

[13] A. Goultiaeva and Y. Lespérance. Incremental plan recognition in an agent programming framework. *Proc. of Plan, Activity and Intent Recognition (PAIR)*, 2007.

[14] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic.* MIT Press, 2000.

[15] A. C. Kakas, R. A. Kowalski, and F. Toni. *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5, chapter The Role of Abduction in Logic Programming, pages 235–324. Oxford University Press, 1998.

[16] J. Laird. It knows what you're going to do: Adding anticipation to a Quakebot. In *Proc. of 5th Intl. Conf. on Autonomous Agents*, pages 385–392, 2001.

[17] A. B. Loyall. *Believable Agents.* PhD thesis, Carnegie Mellon University, 1997.

[18] P. Novák. Jazzyk: A programming language for hybrid agents with heterogeneous knowledge representations. In *Proceedings of ProMAS 2008*, pages 72–87, 2009.

[19] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proc. of the 2nd Intl. Knowl. Repr. Conf. (KR)*, pages 473–484, 1991.

[20] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380, 1991.

[21] M. P. Sindlar, M. Dastani, F. Dignum, and J.-J.Ch. Meyer. Mental state abduction of BDI-based agents. In *Proc. of the 6th Intl. Workshop on Declarative Agent Lang. and Tech. (DALT)*, pages 161–178, 2008.

[22] T. Syrjänen. LPARSE manual. (`http://www.tcs.hut.fi/Software/smodels/`).