Rational Action in Agent Programs with Prioritized Goals

Sebastian Sardiña
Dept. of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
ssardina@cs.toronto.edu

Steven Shapiro
School of Computer Science and Engineering
University of New South Wales
Sydney, NSW 2052, Australia
steven@cse.unsw.edu.au

ABSTRACT

Agent theories and agent programs are two very different styles of specification of agent behavior. The former are declarative in nature, while the latter have an imperative flavor. In this paper, we combine ideas from both areas, yielding a powerful mode of agent specification that also gives the specifier a good deal of control over the complexity of the specified agent. In particular, we extend Shapiro et al.'s [16] agent theory to handle prioritized goals and then integrate it with the IndiGolog agent programming language. The result is a new IndiGolog construct that transforms a given nondeterministic, concurrent program δ into a new program δ' that can be described as a rational implementation of the original program, in the sense that δ' is an implementation of δ, and furthermore, δ' is the most rational of all implementations of δ relative to a given set of prioritized goals and the agent's knowledge. With this construct, we can specify an agent that will attempt to achieve as many goals as possible in priority order even if the agent does not know of a plan that is guaranteed to achieve all the goals. In this case, the agent will select a plan that she thinks has the best chance of achieving the goals.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Languages, Theory, Economics

Keywords

Agent Programming Languages, Rational Action, Situation Calculus

1. PRELIMINARIES

In general terms, this paper is concerned with how to conveniently specify the behavior of an intelligent agent or robot with respect to her knowledge and goals. Her conduct should, in general, be *rational* [13]: she will try her best to satisfy her goals according to what she knows.

Most of the approaches to agent systems [16, 6, 2, 11, 18] either do not represent goals explicitly or use a flat representation in the sense that all goals are equally important. This is clearly not adequate for some scenarios. For example, ensuring that the spaceship does not explode is much more important than any other goal for a space agent and cannot be sacrificed, in principle, for any number of other less important goals.

In this paper, we first extend the framework given in [16] to accommodate agent's goals at different levels of priorities. We then define what rational action is for such an agent. Finally, we show how to specify rational behavior by combining a procedural notion of action with a declarative representations of goals. In particular, we present a novel construct in the IndiGolog [4] agent framework that dictates rational behavior from an agent program and a set of goals.

1.1 Situation Calculus

The *situation calculus* [9, 12] is a second-order language specifically designed for representing dynamically changing worlds in which all changes are the result of named *actions*. There is a set of initial situations corresponding to the ways the agent believes the domain might be initially; the actual initial state of the domain is represented by the distinguished initial situation constant S_0. There is also a distinguished binary function symbol $do(a, s)$ denoting the successor situation to s resulting from performing action a. Relations whose truth values vary from situation to situation are called *fluents*, and are denoted by predicate symbols taking a situation term as their last argument; a special predicate $Poss(a, s)$ is used to state that action a is executable in s.

Within this language, we can formulate action theories that describe how the world changes as the result of the available actions. In Reiter [12], action theories of a special form, called *basic action theories*, were introduced. Such theories include domain-independent foundational axioms that describe the structure of the situations, one successor state axiom per fluent, one precondition axiom per action, and initial state axioms that describe what is true initially. Whereas successor state axioms offer a principled solution to the frame problem by providing an axiomatization of the effects and invariants of actions; precondition axioms state the conditions under which an action can be performed.

Suppose we want to model a world in which there is a safe with a combination lock [10]. If the safe is locked and the correct combination is dialed, then the safe becomes unlocked. However, if the incorrect combination is dialed, the safe explodes. The agent can only dial a combination if the safe is intact, and it is not possible to change the combination of the safe. Here are the axioms for the domain:[1]

$$
\begin{aligned}
&Exploded(do(a,s)) \equiv \\
&\quad [\exists c(a = dial(c) \wedge Comb(s) \neq c) \vee Exploded(s)] \\
&Locked(do(a,s)) \equiv \\
&\quad [\forall c(a \neq dial(c) \vee Comb(s) \neq c) \wedge Locked(s)] \\
&Comb(do(a,s)) = c \equiv [Comb(s) = c] \\
&Poss(dial(c), s) \equiv \neg Exploded(s)
\end{aligned}
$$

The first successor state axiom states that the safe has exploded after doing action a iff a denotes the action of dialing the wrong combination, or if the safe has already exploded. The last axiom is a precondition axiom and it says that it is possible to dial a combination number for the safe in s iff the safe has not exploded in s.

1.2 Knowledge and Perception

So far, in this scenario, the only agents that can definitely unlock the safe are ones that know the combination in advance because if an agent tries a random combination, the safe will likely explode. Suppose the correct combination is written on a piece of paper, and that the agent can read the combination from the paper. How can we model the effects on the world of reading the combination? Scherl and Levesque [15] called this type of action (e.g., perception and communication actions) knowledge-producing actions, and they provided an account of how to represent these actions in the situation calculus. Such actions affect the mental state of the agent rather than the state of the external world. For instance, after performing the action *readComb*, an agent might know the combination of the safe she is trying to open, i.e., the formula $\exists c\, \mathbf{Know}(Comb(\mathsf{now}) = c, do(readComb, s))$ holds.[2] Knowledge is represented by adapting the possible worlds model to the situation calculus (as first done in [10]). $K(s', s)$ represents the fact that in situation s, the agent thinks that she could be in situation s'. We call s' an *alternative situation to s*. $\mathbf{Know}(\phi(\mathsf{now}), s)$ is an abbreviation for the formula $\forall s'.K(s', s) \supset \phi(s')$.

Scherl and Levesque showed how to obtain a successor state axiom for K that completely specifies how knowledge is affected by actions. In our example, the only knowledge-producing action is the *readComb* action. The axiom for K can be specified as follows:

$$
\begin{aligned}
&K(s^*, do(a,s)) \equiv \\
&\quad \exists s'[K(s', s) \wedge s^* = do(a, s') \wedge Poss(a, s') \wedge \\
&\qquad (a = readComb \supset Comb(s') = Comb(s))]
\end{aligned}
$$

First note that for non-knowledge-producing actions (e.g., $dial(c)$), the specification ensures that the only change in knowledge that occurs in moving from situation s to situation $do(dial(c), s)$ is the fact that the action *dial* has been successfully performed. For the case of a knowledge-producing action such as *readComb*, the idea is that in moving from s to $do(readComb, s)$, the agent not only knows that the action has been performed, but also the value of the associated fluent *Comb*. Since in this case we require that $Comb(s') = Comb(s)$, *Comb* will have the same value in all s' such that $K(do(readComb, s'), do(readComb, s))$. Observe that for any situation s, $Comb(do(readComb, s)) = c$ iff $Comb(s) = c$. Therefore, *Comb* has the same value in all alternative situations s^* such that $K(s^*, do(readComb, s))$, and $\exists c\, \mathbf{Know}(Comb(\mathsf{now}) = c, do(readComb, s))$ holds. We require K to be transitive and Euclidean over initial situations. This ensures that the agent always knows whether he knows something (i.e., he has positive and negative introspection). Scherl and Levesque showed that these properties are preserved by the successor state axiom for K.

Throughout the paper we will use a simple extension of the safe problem suggested at the end of [16]: *the combination of the safe may be illegible.* If the paper is legible, then the agent knows the combination of the safe after reading it, as before. But if the combination of the safe is not legible, then it is not possible to read the combination, i.e., the precondition of action *readComb* does not hold. We add a new fluent $PaperLegible(s)$ which means that the paper is legible in s (whose value is initially unknown to the agent) and a new knowledge-producing action *senseLegible*, which tells the agent whether the combination is legible. We assume that there are four initial situations the agent thinks she may be in (depicted in Figure 1): (i) in S_0 the paper is legible and the combination number is 1; (ii) in S_0^2 the paper is legible and the safe's number is 2; (iii) in S_0^3 the paper is not legible and the safe's number is 1, and (iv) in S_0^4 the paper is not legible and the combination is 2.

In Figure 1, situations are nodes in the graph, and the edges are labeled by actions. Part of the K relation is represented by the dashed ovals around the nodes. If a situation s appears in the same box as another situation s', then $K(s', s)$. The figure illustrates that the agent knows neither the combination of the safe nor the state of the paper in S_0, since the value of *Comb* and the truth value of *PaperLegible* vary in the alternative situations S_0, S_0^2, S_0^3 and S_0^4. However, K only relates $do(senseLegible, S_0)$ to itself and to $do(senseLegible, S_0^2)$, therefore, in this situation, the agent does know that the paper is legible.

1.3 Strategies and Ability

With the addition of knowledge to the language, it becomes possible to specify what goals the agent *knows how* to achieve. In [8], $\mathbf{Can}(\phi, s)$ was defined to mean that the agent knows how to achieve ϕ starting in situation s. Intuitively, the definition specifies that in order for the agent to be able to achieve ϕ starting in s, she must know in s of some strategy that she can follow to achieve ϕ eventually. A strategy is formalized as a function from situations to actions, which we call an *action selection function* (ASF). As we will see, this way of formalizing strategies is quite expressive. In particular, it allows the agent's choice of action to vary depending on what knowledge she acquires as she acts.

To see how an ASF can be a model for a strategy, consider our safe example. One strategy the agent can use to unlock the safe is to verify that the piece of paper is legible, find out the combination of the safe by reading it from the paper, and then dial the combination. Notice that this strategy is not just a pair or sequence of actions, since the third action varies according to the actual combination ($dial(c)$ is a different action than $dial(c')$, if $c \neq c'$). The strategy allows

[1]From now on, free variables will be assumed to be universally quantified in the widest scope.

[2]Some formulae will contain a placeholder now instead of a situation argument, e.g., $\neg Locked(\mathsf{now})$. Where the intended meaning is clear, we suppress the placeholder, e.g., $\neg Locked$. $\phi(s)$ is the formula that results from substituting s for now in ϕ.

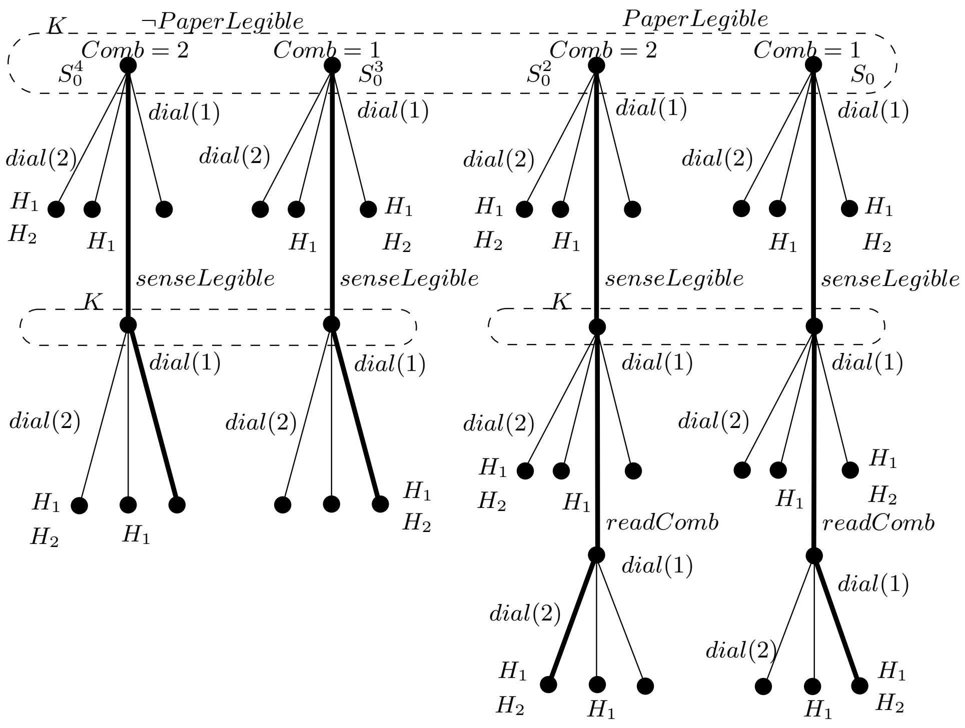

Figure 1: The possible world approach to knowledge and the ASF σ_1 (thick edges). H_i **stands for** $H(\sigma, i, s)$**; and an unlabeled edge represents a** *noOp* **action.**

the agent to take different actions depending on the knowledge she acquires as she follows the strategy. Let σ be an ASF, i.e., a mapping from situations to actions. Given a starting situation s_0, σ defines an infinite sequence of situations: $s_0, s_1, s_2, ...$ where $s_i = do(\sigma(s_{i-1}), s_{i-1})$ for $i \geq 1$. We define $\mathbf{OnPath}(\sigma, s, s')$ to mean that situation s' is in the situation sequence defined by σ and s:[3]

$$\mathbf{OnPath}(\sigma, s, s') \stackrel{\text{def}}{=} s \leq s' \wedge \forall a, s^*(s < do(a, s^*) \leq s' \supset \sigma(s^*) = a)$$

If an ASF σ maps different alternative situations to different actions, then the agent cannot 'follow' it. She also cannot follow it if a prescribed action is not possible. We are only interested in ASFs that the agent is able to follow. We formalize this notion as follows:

$$\mathbf{CanFollow}(\sigma, s) \stackrel{\text{def}}{=} \forall s'.\mathbf{OnPath}(\sigma, s, s') \supset \exists a\, \mathbf{Know}(\sigma(\mathsf{now}) = a, s') \wedge Poss(\sigma(s'), s') \quad (1)$$

The actions labeling the transitions between situations in the sequence σ can be thought of as a possible course of action for the agent to follow if she is in situation s. In fact, this is the case for every alternative situation s^* that the agent thinks she might be in when she really is in s, i.e., σ also defines a course of action from s^*.

In our example, the agent does not know initially (i.e., in S_0) whether the combination is 1 or 2, nor does she know whether the piece of paper is readable. An ASF σ_1 modeling the strategy outlined earlier has the following mappings:

[3]$s < s'$ means that there is a sequence of *possible* actions that can be performed starting in situation s and which results in situation s'. $s \leq s'$ is an abbreviation for $s < s' \vee s = s'$.

$$\begin{aligned}
&\sigma_1(S_0) = \sigma_1(S_0^2) = \sigma_1(S_0^3) = \sigma_1(S_0^4) = senseLegible\\
&\sigma_1(do(senseLegible, S_0)) = readComb\\
&\sigma_1(do(senseLegible, S_0^2)) = readComb\\
&\sigma_1(do(readComb, do(senseLegible, S_0))) = dial(1)\\
&\sigma_1(do(readComb, do(senseLegible, S_0^2))) = dial(2)\\
&\sigma_1(do(senseLegible, S_0^3)) = dial(1)\\
&\sigma_1(do(senseLegible, S_0^4)) = dial(1)
\end{aligned}$$

Observe that this strategy guesses the combination to be 1 when the paper is found to be illegible (last two mappings). The two interesting alternative strategies σ_2 and σ_0 are like σ_1 except that they prescribe different behavior whenever the paper is found to be illegible: σ_2 dictates dialing 2 whereas σ_0 dictates doing nothing, i.e., σ_2 (σ_0, resp.) prescribes action $dial(2)$ (*noOp*,[4] resp.) in situations $do(senseLegible, S_0^3)$ and $do(senseLegible, S_0^4)$.

Strategy σ_1 is illustrated in Figure 1 via thick edges. Observe that strategy σ_1 succeeds in all possible worlds, except when the agent cannot read the paper and the combination turns out to be 2 (i.e., in S_0^4).

It can be proven that if the agent knows in S_0 that the safe is intact and that the paper is legible (i.e., if situations S_0^3 and S_0^4 were not accessible from S_0,) then she can unlock the safe: the agent can use σ_1, σ_2, or σ_0 to achieve the goal. On the other hand, if the agent were unable to read the combination in situation S_0 (and still did not know the combination in S_0), we could show that $\neg\mathbf{Can}(\neg Locked, S_0)$ holds. Ability can be formally specified as follows (see [8]):

$$\mathbf{Achieve}(\phi, \sigma, s) \stackrel{\text{def}}{=} \mathbf{Know}(\mathbf{CanFollow}(\sigma, \mathsf{now}), s) \wedge \mathbf{Know}(\exists s'.\mathbf{OnPath}(\sigma, \mathsf{now}, s') \wedge \phi(s'), s)$$

$$\mathbf{Can}(\phi, s) \stackrel{\text{def}}{=} \exists\sigma.\mathbf{Achieve}(\phi, \sigma, s)$$

[4]*noOp* is the dummy action that has no effects.

2. PRIORITIZED GOALS

In [16], a flexible way of specifying the goals of an agent was presented. Shapiro et al. characterized the goals of the agent by specifying the paths (sequences of situations, which were modeled using ASFs) in which all the goals (both maintenance goals and achievement goals) are achieved. A (happy) predicate $H(\sigma, s)$ was used to denote whether the path defined by σ starting at s satisfies the agent's goals. For instance,

$$H(\sigma, s) \equiv \mathbf{Eventually}(\neg \mathit{Locked} \wedge \neg \mathit{Exploded}, \sigma, s)$$

states that the agent's goal paths are those where eventually the safe is unlocked and intact. Relation $\mathbf{Eventually}(\alpha, \sigma, s)$[5] ($\mathbf{Always}(\alpha, \sigma, s)$, resp.) means that α will eventually (always, resp.) hold along the path defined by σ starting at s:

$$\mathbf{Eventually}(\alpha, \sigma, s) \overset{\text{def}}{=} \exists s^*.\mathbf{OnPath}(\sigma, s, s^*) \wedge \alpha(\sigma, s^*)$$
$$\mathbf{Always}(\alpha, \sigma, s) \overset{\text{def}}{=} \forall s^*.\mathbf{OnPath}(\sigma, s, s^*) \supset \alpha(\sigma, s^*)$$

Shapiro et al. defined the goals of the agent to be those formulae that completely characterize the agent's H-paths which are consistent with what she knows.

$$\mathbf{OGoal}(\phi, s) \overset{\text{def}}{=} \forall \sigma, s'.K(s', s) \supset (H(\sigma, s') \equiv \phi(\sigma, s'))$$

Finally, they defined $\mathbf{Rational}(\sigma, s)$, meaning that σ is a rational strategy in s w.r.t. the goals of the agent in the sense that there is no other alternative behavior that is known to the agent to be strictly better than σ. A drawback of this account of goals is that it may be overly flat for some problems due to the fact that all goals are equally important. The consequence of this limitation is that, as Shapiro et al. explained, it could sometimes be rational for the agent to guess the combination (as strategies σ_1 and σ_2 dictate after realizing the paper is not readable.) Although this allows the agent to unlock the safe in some possible worlds, in most, the safe will explode! Clearly, this might be a poor strategy for the agent to take.

What we are missing is a way of specifying goals with different priorities: *even though we strongly want the safe unlocked, it is more important to keep the safe intact.* In this setting, a rational agent would never jeopardize the safe in order to try to unlock it, if she is unsure of the combination.

We begin our account of rational action w.r.t. a set of prioritized goals by introducing different levels of goals; $H(\sigma, s)$ is replaced by $H(\sigma, n, s)$, where n is a positive integer, denoting those paths that satisfy the agent's level n goals. The lower the level n, the higher the priority, the more important the goal is. Here, the high-priority goal of keeping the safe intact would be specified as a level 1 goal (formally, by an axiom $H(\sigma, 1, s) \equiv \mathbf{Always}(\neg \mathit{Exploded}, \sigma, s)$); and the low-priority goal of unlocking the safe would be specified at level 2 (that is, $H(\sigma, 2, s) \equiv \mathbf{Eventually}(\neg \mathit{Locked}, \sigma, s)$).

In order to represent a finite number k of prioritized goals, we make $H(\sigma, i, s)$ identically true for every path whenever $i > k$. We use an axiom parameterized by k, called $I(k)$, for this purpose:

$$I(k) \overset{\text{def}}{=} \forall i, \sigma, s.i > k \supset (H(\sigma, i, s) \equiv \mathit{TRUE}) \qquad (2)$$

[5]Some formulae will contain two placeholders sit and asf. Again, we suppress the placeholder where possible. $\phi(\sigma, s)$ is the formula that results from replacing asf with σ and sit with s in ϕ.

Given a specification for H, we can formally state what we mean by a goal. In contrast with [16], our framework distinguishes goals at different levels. $\mathbf{OGoal}(\alpha, n, s)$ holds if α is the level n goal of the agent in situation s:

$$\mathbf{OGoal}(\phi, n, s) \overset{\text{def}}{=} \forall \sigma, s'.[K(s', s) \supset (H(\sigma, n, s') \equiv \phi(\sigma, s'))]$$

From now on, we will use $\mathcal{D}$ (possibly with decorations) to denote a basic action theory (including the axiom for K), and $\mathcal{H}$ (possibly with decorations) to denote a set of axioms defining H, i.e., a set of axioms of the following form: for some $r > 0$, $\phi_1, \ldots, \phi_r$: $\{\forall \sigma, s.H(\sigma, i, s) \equiv \phi_i(\sigma, s) \mid 1 \leq i \leq r\} \cup \{I(r)\}$. In particular, if we let $\mathcal{D}_{Safe}$ and $\mathcal{H}_{Safe}$ stand for the safe example axiomatization, then $\mathcal{H}_{Safe}$ would be formed by the above axioms for $H(\sigma, 1, s)$ and $H(\sigma, 2, s)$ together with the axiom $I(2)$. We can then show that, initially, the agent has a high-priority goal of keeping the safe intact and a low-priority goal of unlocking the safe:

$$\mathcal{D}_{Safe} \cup \mathcal{H}_{Safe} \models \mathbf{OGoal}(\mathbf{Always}(\neg \mathit{Exploded}), 1, S_0) \wedge \mathbf{OGoal}(\mathbf{Eventually}(\neg \mathit{Locked}), 2, S_0)$$

We write $(\phi_1 > \phi_2 > ... > \phi_n)_s$ when the agent has n (non-trivial) priority goals, and ϕ_i is an $\mathbf{OGoal}$ in s at level i (e.g., $(\mathbf{Always}(\neg \mathit{Exploded}) > \mathbf{Eventually}(\neg \mathit{Locked}))_{S_0}$). Observe that while the goal at level 1 is a maintenance goal, the goal at level 2 is an achievement goal.

3. RATIONAL ACTION

In this section, we define what we mean by rational action in the context of prioritized goals. Acting rationally involves both knowledge and desires; what we know and what we want. The latter is represented by our goals, which should constrain our future actions.

If an agent is acting rationally, then *to the best of her abilities* she is acting to bring about (and maintain) her goals [13]. In other words, if an agent's actions are rational starting in situation s, then, ideally, she is following an ASF σ such that $\forall n.H(\sigma, n, s)$. But since the agent may be uncertain as to which situation the world is actually in, she ought to be following a course of action that she *knows* will achieve her goals, i.e., $\forall s'.K(s', s) \supset \forall n.H(\sigma, n, s')$. However, there may not always be such a σ for the agent to follow. In many cases, although it may not be possible for the agent to guarantee success w.r.t. (all) her goals, it is rational to try a strategy with the best chance of success while respecting the goals' priorities. To that end, we will define an ordering over ASFs for each situation. We say that σ_1 *is as preferred as* σ_2 *up to priority* n *in situation* s (i.e., $P(\sigma_1, \sigma_2, n, s)$), if: (i) σ_1 is as preferred as σ_2 in all higher priorities (i.e., in all lower priority levels); and (ii) σ_1 *dominates* σ_2 at level n (i.e, $\sigma_1 \succeq^s_n \sigma_2$), that is, it achieves the agent's goals at priority level n in all the alternative situations in which σ_2 achieves these goals. We formalize this ordering with the following definition and recursive axiom:

$$\sigma_1 \succeq^s_n \sigma_2 \overset{\text{def}}{=} \forall s'.K(s', s) \wedge H(\sigma_2, n, s') \supset H(\sigma_1, n, s')$$

$$\begin{aligned} P(\sigma_1, \sigma_2, n, s) \equiv\ & \sigma_1 \succeq^s_n \sigma_2 \wedge \\ & (\forall n'.n' < n \wedge P(\sigma_2, \sigma_1, n', s) \supset P(\sigma_1, \sigma_2, n', s)) \end{aligned} \qquad (3)$$

Finally, we say that a strategy σ_1 is as good as σ_2 in s iff whenever σ_2 is as preferred as σ_1 up to some level n, it is the case that σ_1 is as preferred as σ_2 up to that level too.

$$\mathbf{AsGood}(\sigma_1, \sigma_2, s) \overset{\text{def}}{=} \forall n.P(\sigma_2, \sigma_1, n, s) \supset P(\sigma_1, \sigma_2, n, s)$$

The following theorem makes explicit the relation between dominance at each level and the **AsGood** relation. It says that σ_1 is as good as σ_2 in s iff either σ_1 dominates σ_2 at all levels where it is dominated by σ_2, or this holds up to some level m and σ_1 strictly dominates σ_2 at level m.

THEOREM 1. [6]

$$\begin{aligned}\models \forall \sigma_1, \sigma_2, s.\mathbf{AsGood}(\sigma_1, \sigma_2, s) \ \equiv \\ (\forall n.\sigma_2 \succeq^s_n \sigma_1 \supset \sigma_1 \succeq^s_n \sigma_2) \ \vee \\ \exists m.\sigma_1 \succ^s_m \sigma_2 \wedge \forall i.i < m \supset (\sigma_2 \succeq^s_i \sigma_1 \supset \sigma_1 \succeq^s_i \sigma_2)\end{aligned}$$

We now give some further properties of these definitions. The first one says that if σ_1 is strictly preferred to σ_2 for some level n, then σ_2 will never be as preferred as σ_1 for any greater level and, as a consequence, strategy σ_1 will be strictly better than σ_2.

Property 1:[7]

(i) $\{(3)\} \models \forall s, m, n, \sigma_1, \sigma_2.P^{\not\approx}(\sigma_1, \sigma_2, n, s) \ \wedge$
$m \geq n \supset \neg P(\sigma_2, \sigma_1, m, s)$, and

(ii) $\{(3)\} \models \forall s, \sigma_1, \sigma_2.(\exists n.P^{\not\approx}(\sigma_1, \sigma_2, n, s)) \ \supset$
$\mathbf{AsGood}^{\not\approx}(\sigma_1, \sigma_2, s)$

The next property, and the most important, says that it is preferable to perform as well as possible w.r.t. a particular priority k goal, than to satisfy any number—even *all*—of the lower priority goals. In this sense, goal priorities are *strict*.

Property 2:

$$\begin{aligned}\{(3)\} \models \forall \sigma_1, \sigma_2, s, k.(\forall j.j < k \supset (\sigma_1 \succeq^s_j \sigma_2 \equiv \sigma_2 \succeq^s_j \sigma_1)) \ \wedge \\ \sigma_1 \succ^s_k \sigma_2 \supset \mathbf{AsGood}^{\not\approx}(\sigma_1, \sigma_2, s)\end{aligned}$$

We are now able to define what we mean by rational activity with respect to a set of prioritized goals. In a nutshell, we say that an ASF σ describes a rational course of action in s, if the agent knows that it is possible to execute σ starting from s, and as far as the agent knows, σ is as good as any other (known executable) strategy.

$$\begin{aligned}&\mathbf{RationalPrio}(\sigma, s) \overset{\text{def}}{=} \\ &\quad \mathbf{Know}(\mathbf{CanFollow}(\sigma, \mathsf{now}), s) \ \wedge \\ &\quad (\forall \sigma'.\mathbf{Know}(\mathbf{CanFollow}(\sigma', \mathsf{now}), s) \supset \mathbf{AsGood}(\sigma, \sigma', s))\end{aligned}$$

Again, we should remark that, under this account of rationality, the priority ordering among goals is strict, i.e., lexicographic, in that a rational agent will prefer a more important goal to even all lower priority ones together. In any case, we can show that whenever there is only one priority level of goals (i.e., level 1), our account coincides with the one proposed in [16]. Recall that $\mathbf{Rational}(\sigma, s)$ stands for rational action in Shapiro et al.'s framework [16].

THEOREM 2. *Let $\mathcal{H}$ be the axioms $\{I(1), \forall \sigma, s.H(\sigma, s) \equiv H(\sigma, 1, s)\}$. Then,*

$$\begin{aligned}\mathcal{H} \cup \{(3), (\forall \sigma, s.\mathbf{CanFollow}(\sigma, s))\} \ \models \\ \forall \sigma, s.\mathbf{RationalPrio}(\sigma, s) \equiv \mathbf{Rational}(\sigma, s)\end{aligned}$$

In this way, Shapiro et al.'s approach is subsumed by ours. On the other hand, the next result shows that our framework is more general in that the agent respects goal priorities.

[6] $\sigma_1 \succ^s_n \sigma_2$ stands for $\sigma_1 \succeq^s_n \sigma_2 \wedge \neg(\sigma_2 \succeq^s_n \sigma_1)$
[7] $P^{\not\approx}(\sigma_1, \sigma_2, n, s)$ stands for $P(\sigma_1, \sigma_2, n, s) \wedge \neg P(\sigma_2, \sigma_1, n, s)$. Similarly, $\mathbf{AsGood}^{\not\approx}(\sigma_1, \sigma_2, s)$ stands for $\mathbf{AsGood}(\sigma_1, \sigma_2, s) \wedge \neg\mathbf{AsGood}(\sigma_2, \sigma_1, s)$.

THEOREM 3. *For any formula $\phi(\mathsf{asf}, \mathsf{now})$:*

$$\begin{aligned}&\{(3)\} \ \models \\ &\forall s, n, \sigma_\phi.\mathbf{OGoal}(\mathbf{Eventually}(\phi), n, s) \ \wedge \\ &\quad \mathbf{Achieve}(\phi, \sigma_\phi, s) \wedge \mathbf{RationalPrio}(\sigma, s) \ \wedge \\ &\quad \neg\mathbf{Know}(\mathbf{Eventually}(\phi, \sigma, \mathsf{now}), s) \ \supset \\ &\qquad \exists n'.n' < n \wedge P^{\not\approx}(\sigma, \sigma_\phi, n', s)\end{aligned}$$

Thus, whenever an agent has sufficient knowledge and capabilities to achieve a goal ϕ at some priority n, she would only risk failing to achieve that goal for the sake of achieving a goal with a higher priority. This type of behavior cannot be represented in Shapiro et al.'s framework, since they do not handle prioritized goals.

With this definition of rational action, we can show that if the agent neither knows the combination nor whether the paper is legible (as in Figure 1), then σ_0 is a provably rational course of action whereas σ_1 and σ_2 are provably irrational. The latter two strategies *do not* ensure that the higher priority goal of keeping the safe intact is achieved in a situation where the agent cannot read the combination.

4. AGENT PROGRAMS

So far we have seen how it is possible to represent prioritized goals and rational activity based on those goals. Under this framework, we expect to derive the agent's behavior merely from a description of her knowledge and goals; this approach is very similar to first-principle planning. Many researchers, though, favor the *agent programming* approach [17, 12, 7, 18] to action selection in which the user specifies not just the goals, but also constraints on how they are to be achieved, perhaps leaving small sub-tasks to be handled by an automatic planner. Efficiency and practical reasons for building agents are the most common arguments articulated in favor of this alternative approach. In this section, we will briefly go over IndiGolog [4], the most recent language in the Golog family [12, 2]. In the following section, we shall investigate how to incorporate the rationality account developed in Section 3 into this agent language so as to obtain a rich tool for specifying agent conduct.

In IndiGolog, the agent's behavior arises as the consequence of executing a high-level program. By a high-level program we mean one whose primitive instructions are domain-dependent actions of the robot, whose tests involve domain-dependent fluents affected by these actions, and whose code may contain nondeterministic choice points. Instead of looking for a legal sequence of actions achieving some goal, the task now is to find a sequence that constitutes a legal execution of a high-level program.

To informally introduce the syntax and some of the common constructs of this programming language, we show a possible program for our version of the safe scenario.

```
proc open_safe
    (πa.a)*; (holding(paper))?;
    senseLegible*; readComb*;
    (πc.dial(c) | (true)?);
    (open | call(locksmith));
end
```

where $\delta_1; \delta_2$ stands for sequence of programs δ_1 and δ_2; $\pi x.\delta(x)$ for nondeterministic choice of argument x; $\delta_1 | \delta_2$ for nondeterministic selection between programs δ_1 and δ_2; and δ^* for nondeterministic iteration of program δ (zero, one, or more times). Lastly, action $(\phi)?$ checks that condition ϕ holds and is the closest tool for representing a goal in a declarative fashion. It is easy to observe that the above program has many gaps due to nondeterministic choice points that need to be resolved by, e.g., an automated planner. For example, the first two complex actions $(\pi a.a)^*; (\text{holding(paper)})?$ require the agent to select some number of actions (e.g., drop the object she is holding, open the drawer, pick up the paper, etc.) such that after executing them she would have the paper in her hands. As the reader may have noticed, that particular sub-task is very similar to classical planning.[8]

4.1 Incremental Execution of Programs

Finding a legal execution of high-level programs is at the core of the whole approach. A sequence of actions standing for a program execution will ultimately be taken as the agent's behavior. Originally, Golog and ConGolog programs were conceived to be executed (verified) offline; the interpreter looks for a sequence of actions $[a_1, \dots, a_m]$ such that $Do(\delta, s, do(a_m, do(a_{m-1}, \dots, do(a_1, S_0))))$ is entailed by the specification, where $Do(\delta, s, s')$ is intended to say that situation s' represents a legal execution of program δ starting from situation s. Once a sequence like that is found, the agent or robot is supposed to execute it one action at a time. Clearly, this type of execution remains infeasible for large programs and precludes both runtime sensing information and reactive behavior. To deal with these drawbacks, De Giacomo and Levesque [4] provided a formal notion of interleaved planning, sensing, and action. In their account, they make use of two predicates defined in [2] in order to give a single-step semantics to programs by defining the so-called *online* or *incremental* executions: $Trans(\delta, s, \delta', s')$ says that program δ in situation s may execute one step, ending in situation s' with program δ' remaining; and $Final(\delta, s)$ indicates that program δ may terminate in situation s.

We will refer with $\mathcal{T}$ to the set of axioms for $Trans$ and $Final$, and those needed for the encoding of programs as first-order terms (see [2]). We omit the axioms here due to space limitations.

Roughly speaking, an incremental execution of program finds a next possible action, executes it in the real world, obtains sensing information afterward, and repeats the cycle until the program is finished. The fact that actions are quickly executed without much deliberation and sensing information is gathered after each step makes the approach realistic for dynamic and changing environments. However, given that such an execution framework requires committing in the world at each step and programs may contain nondeterministic points, some lookahead mechanism is required to avoid unsuccessful (dead-end) executions. To that end, the authors developed a new language construct Σ, the search operator, as a local, controlled form of offline verification such that the amount of lookahead to be performed is under the control of the programmer. As with all the other language constructs, a single-step semantics for it can be defined by giving the corresponding axioms for $Trans$ and $Final$: $\Sigma\delta$ selects from all possible transitions of (δ, s) those for which there exists a sequence of further transitions leading to a final configuration (δ', s').

[8]In fact, one would prefer to avoid this kind of sub-task and write more detailed programs since the search space required for such sub-tasks will be huge.

The point is that in IndiGolog one is able to specify quick reactive behavior as well as user-controlled deliberation. In what follows, we shall try to incorporate the treatment of rationality of Section 3 into this kind of limited deliberation.

5. RATIONAL ACTION AND PROGRAMS

One of the major drawbacks with most existing agent languages [12, 7, 11, 18] is that they usually underestimate the utility of expressing desires in a declarative way. Behavior modeling is therefore reduced to a procedural representation in which the only (implicit) goal is to find and follow a legal execution of a given agent program. Combining procedural goals, i.e., goals-to-do, and declarative goals has been investigated by some researchers [19, 6], but mainly ignored.

In addition, many agent programming frameworks have a restricted notion of declarative goals by reducing them to tests in programs. The agent is then required either to make the test true by selecting appropriate actions or to fail. This is the case with languages like Golog [12, 2], 3APL [7], and FLUX [18]. As a result, behavior like "just guess the combination number because there is nothing better to do" cannot be obtained unless explicitly programmed.

Here, we present our second contribution: a novel construct for IndiGolog which combines both a procedural representation of behavior and prioritized declarative goals. The idea is simple: given a non-deterministic IndiGolog program δ and a set of prioritized goals $\phi_1 > ... > \phi_n$, the new *rational-search* operator $\Delta_{rat}(\delta : \phi_1 > ... > \phi_n)$ will produce a simple and ready-to-execute plan whose execution will respect both the given program δ and the set of declarative objectives.

To that end, we start by relating *programs* with *strategies.* We say that a strategy is induced by a program in some situation if, starting in that situation, it dictates the same sequence of actions as some execution of the program.

$$\begin{aligned}&\mathbf{Induced}(\delta, \sigma, s) \overset{\text{def}}{=} \\ &\quad \forall s^*.\mathbf{OnPath}(\sigma, s, s^*) \supset \exists \delta'.Trans^*(\delta; \delta_{noOp}, s, \delta', s^*)\end{aligned}$$

where $\delta_{noOp} = \textbf{while}\ True\ \textbf{do}\ noOp\ \textbf{endWhile}$, and $Trans^*$ is the reflexive, transitive closure of $Trans$.

Notice that δ_{noOp} is used to generate activity after δ terminates. Hence, strategies induced by a terminating program dictate a *noOp* action in every situation that is beyond the program execution. The first thing to notice is that, because a program may be non-deterministic, it may induce *many* equivalence classes of strategies w.r.t. a situation. We say that two strategies are equivalent in a situation s if they dictate exactly the same actions if executed from situation s.

$$\begin{aligned}&\mathbf{Equivalent}(\sigma_1, \sigma_2, s) \overset{\text{def}}{=} \\ &\qquad \forall s^*.\mathbf{OnPath}(\sigma_1, s, s^*) \supset \sigma_1(s^*) = \sigma_2(s^*)\end{aligned}$$

Secondly, a program may induce no strategy at all, as is the case with the unexecutable program $(a|b); (FALSE)?$. Finally, a program may induce unexecutable strategies, that is, strategies that the agent cannot follow. That is the case with the program $\delta = \textbf{if}\ q\ \textbf{then}\ a\ \textbf{else}\ b$ in a situation where the agent does not know the truth value of proposition q.

Next, we incorporate programs of the form $\Delta_{rat}(\delta : \phi_1 > ... > \phi_n)$ into the language, where Δ_{rat} is the new rational search operator. We omit the details on how to do this, but we quickly point out that the task is achieved by extending the encoding of programs in [2] in two ways. First, a special *prioritized-goal* sort and terms are added by using the binary function $>$: (i) every formula ϕ is a prioritized-goal; (ii) if ϕ is a formula and G is a prioritized-goal, then $G > \phi$ is a prioritized goal; and (iii) nothing else is a prioritized goal. We assume that $\mathcal{T}$ contains the axioms encoding this sort. We also assume there are infinitely many variables $x_G, y_G, ...$ for this new sort. Second, a new program construct Δ_{rat} is added so that $\Delta_{rat}(\delta : G)$ is a legal program whenever δ is a program not mentioning Δ_{rat} and G is a prioritized-goal. Roughly speaking, an execution of program $\Delta_{rat}(\delta : G)$ is a legal execution of program δ that is rational w.r.t. goals G.

With this in hand, we are able to embed our account of rationality given in Section 3 into IndiGolog by adding a formula (corresponding to a goal of the agent) as an argument to previously defined predicates. First, we redefine our notion of dominance at a level as dominance w.r.t. a formula: $\sigma_1 \succeq^s_\phi \sigma_2$ means that strategy σ_1 is as least as good as σ_2 in s w.r.t. the single goal ϕ.

$$\sigma_1 \succeq^s_\phi \sigma_2 \stackrel{\text{def}}{=} \forall s'.K(s', s) \wedge \phi(\sigma_2, s') \supset \phi(\sigma_1, s')$$

Next, we define relation $AsGood(\sigma_1, \sigma_2, x_G, s)$ with the intended meaning that ASF σ_1 is as good as σ_2 in situation s for the sequence of goals x_G. Note that we use axioms instead of a definition here, since we define this relation recursively.

$$AsGood(\sigma_1, \sigma_2, \phi, s) \equiv \sigma_2 \succeq^s_\phi \sigma_1 \supset \sigma_1 \succeq^s_\phi \sigma_2$$
$$AsGood(\sigma_1, \sigma_2, x_G > \phi, s) \equiv AsGood(\sigma_1, \sigma_2, x_G, s) \wedge (\neg\sigma_2 \succeq^s_\phi \sigma_1 \vee \sigma_1 \succeq^s_\phi \sigma_2 \vee \neg AsGood(\sigma_2, \sigma_1, x_G, s))$$

We will refer with $\mathcal{R}$ to the set containing these three axioms. It is possible to prove that $AsGood(\sigma_1, \sigma_2, \phi, s)$ coincides exactly with $\mathbf{AsGood}(\sigma_1, \sigma_2, s)$:

THEOREM 4. *Let $r > 0$, and let $\mathcal{H}_2$ be a set of axioms of the form $\forall\sigma, s.H(\sigma, i, s) \equiv \phi_i(\sigma, s)$, for $i = 1 \dots r$, together with the extra axiom $I(r)$ from (2). Then:*

$$\mathcal{R} \cup \mathcal{H}_2 \cup \mathcal{T} \cup \{(3)\} \models \mathbf{AsGood}(\sigma_1, \sigma_2, s) \equiv AsGood(\sigma_1, \sigma_2, \phi_1 > ... > \phi_r, s)$$

We now have everything in place to give our definition of rationality in the new framework. For later convenience and flexibility we shall use a given program to restrict the set of ASFs to which the strategy in question will be compared.

$$\mathbf{RationalPrio}(\sigma, \delta, x_G, s) \stackrel{\text{def}}{=} \mathbf{Know}(\mathbf{CanFollow}(\sigma, \mathsf{now}), s) \wedge [\forall\sigma'.\mathbf{Know}(\mathbf{Induced}(\delta, \sigma', \mathsf{now}) \wedge \mathbf{CanFollow}(\sigma', \mathsf{now}), s) \supset AsGood(\sigma, \sigma', x_G, s)]$$

In other words, $\mathbf{RationalPrio}(\sigma, \delta, x_G, s)$ means that, from the agent's perspective in situation s, σ is as good w.r.t. prioritized goals x_G as any strategy that can obtained from program δ. The following is a corollary of Theorem 4 and it shows that the embedding of the framework given in Section 3 is correct if we take δ to be the 'universal' program.

COROLLARY 1. *Under the conditions in Theorem 4,*

$$\mathcal{R} \cup \mathcal{H}_2 \cup \mathcal{T} \cup \{(3)\} \models \forall\sigma, s.\mathbf{RationalPrio}(\sigma, s) \equiv \mathbf{RationalPrio}(\sigma, (\pi a.a)^*, \phi_1 > ... > \phi_r, s)$$

We have already seen that the idea in agent programming is that agents do not build plans from scratch; instead, they obtain their detailed plans based on supplied plan sketches, or abstract plans. Therefore, the task of the agent is to deliberate in order to synthesize the best plan possible from the given sketches, relative to her declarative goals. A program δ_r is considered rational w.r.t. another more abstract program δ if δ_r is the best (simple) plan contained in δ, relative to the goals x_G.

$$\mathbf{DRationalSol}(\delta, \delta_r, x_G, s) \stackrel{\text{def}}{=} \exists\sigma.\mathbf{Know}(\mathbf{UInduced}(\delta_r, \sigma, \mathsf{now}) \wedge \mathbf{Induced}(\delta, \sigma, \mathsf{now}), s) \wedge \mathbf{RationalPrio}(\sigma, \delta, x_G, s)$$

$$\mathbf{UInduced}(\delta, \sigma, s) \stackrel{\text{def}}{=} \mathbf{Induced}(\delta, \sigma, s) \wedge (\forall\sigma'.\mathbf{Induced}(\delta, \sigma', s) \supset \mathbf{Equivalent}(\sigma, \sigma', s))$$

In words, a plan δ_r is a (deterministic) *rational* solution of program δ in s w.r.t. goals x_G iff the following hold: (i) the agent knows of a particular strategy representing the *unique* equivalence class of induced strategies of δ_r in s; (ii) the agent knows that this strategy is also induced by the original program δ; and (iii) the strategy in question is rational among the possible strategies induced by the original program. This definition is important as it gives us an account of which detailed programs are seen as correct solutions of another abstract program such that the former are not only an executable solution of the latter, but also are rational w.r.t. the agent's knowledge and desires.

Notice that with rational *programs*, the most important goal, though implicit, is to be able to execute the program in question. Therefore, program executability is in fact the top-priority goal (and a difficult one to achieve), for no program can be rational if it cannot be executed by the agent.

We now have all the machinery needed to introduce our *rational* search operator Δ_{rat} for the IndiGolog agent programming language. We will assume that the background theory $\mathcal{D}$ is *epistemically accurate*, meaning that, among other things, what is known accurately reflects what the theory says about the dynamic system (see [3]). Here is the single-step semantics for Δ_{rat}:

$$Trans(\Delta_{rat}(\delta : x_G), s, \delta'_{dp}, s') \equiv \exists\delta_{dp}.Trans(\delta_{dp}, s, \delta'_{dp}, s') \wedge \mathbf{DRationalSol}(\delta, \delta_{dp}, x_G, s) \wedge \exists s_f.Do(\delta_{dp}, s, s_f)$$

$$Final(\Delta_{rat}(\delta : x_G), s) \equiv \mathbf{DRationalSol}(\delta, \delta_{noOp}, x_G, s)$$

Thus, configuration $(\Delta_{rat}(\delta : x_G), s)$ evolves to configuration (δ'_{dp}, s') if the latter is the remaining configuration after performing a single step over a rational and terminating solution δ_{dp} of δ in situation s. Finally, $(\Delta_{rat}(\delta : x_G), s)$ is a legal terminating configuration if not doing anything (i.e., doing *noOp*) is a rational behavior. Hence, unlike previous lookahead constructs proposed for Golog, it is *not* always rational to terminate a program whenever that is possible.

To illustrate how Δ_{rat} works, let us take $\mathcal{D}_1$ and $\mathcal{D}_2$ to be two versions of the safe problem. In the former, the agent does not know the combination in S_0; in the latter, she knows that the combination is 1 right from the start. In both cases, the agent does not know whether the paper is legible. Consider the following three programs:

$\delta_0 \stackrel{\text{def}}{=}$ pickup(paper); dial(1) ; open

$\delta_1 \stackrel{\text{def}}{=}$ pickup(paper); senseLegible;
  if PaperLegible **then** {readComb; dial(Comb); open}
  else call(locksmith)

$\delta_2 \overset{\text{def}}{=}$ pickup(paper); senseLegible;

$\quad$ **if** PaperLegible **then** {readComb; dial(Comb); open}

$\quad\quad$ **else** {dial(1); open}

Suppose $C_1 = (\Delta_{rat}(open\_safe : \mathbf{Always}(\neg Exploded) > \mathbf{Eventually}(\neg Locked)), S_0)$ and $C_2 = (\Delta_{rat}(open\_safe : \mathbf{Eventually}(\neg Locked)), S_0)$ are two program configurations that use procedure $open\_safe$ from Section 4. We can show the expected intuitive results. First, C_1 evolves to configuration $(\delta_1^-, do(pickup(paper), S_0))$[9] w.r.t. $\mathcal{D}_1$ and $\mathcal{D}_2$, since, unlike δ_2 and δ_0, δ_1 is a rational solution of $open\_safe$ in S_0 w.r.t. the goals. Second, C_2 evolves to configuration $(\delta_2^-, do(pickup(paper), S_0))$ w.r.t. both theories since, unlike δ_1 and δ_0, guessing (maybe wrongly) is a rational behavior when the only goal is to get the safe open. Lastly, both C_1 and C_2 have legal transitions to $(\delta_0^-, do(pickup(paper), S_0))$ w.r.t. theory $\mathcal{D}_2$ because the agent already knows the right combination of the safe in S_0.

6. CONCLUSIONS

In this article, we have developed an abstract account of rational activity for an agent with respect to a set of goals with different priorities and coupled it with an agent programming framework. All proofs can be found in an extended version of this paper [14]. Many issues need to be addressed. In particular, our approach provides a strict account of prioritized goals that, although perfectly adequate for many scenarios, may fail to address other domains where a softer notion of priorities is required. In such domains, doing well w.r.t. many lower priority goals may be preferred to guaranteeing just one high priority goal. We think it is possible to build a mixed framework with both *hard* and *soft* goals; using ideas from the economic decision-theory approach to rationality [5]. Indeed, DTGolog [1], a decision theoretic version of Golog, is strongly related to this paper in that it chooses a *preferable* execution of a program among all legal ones, though using utility functions instead of symbolic goals. We consider investigating how to combine ideas from DTGolog (including its implementation) with our own a promising area of future research.

Our notion of rationality differs substantially with the usual one in decision under strict uncertainty [5]. First, we deal with infinite set of choices (i.e., strategies); second, our definition of rationality does not always suggest a complete ranking (i.e., a weak-order) among strategies; and, third, there may be circumstances where no rational strategy exists. The point is that our rationality account is intended to be used as a conservative local deliberation process embedded in the more general and flexible account of agent behavior provided by the IndiGolog agent programming language. Further work is required to relate our account with the well-known topic of rationality under strict uncertainty in decision theory and, possibly, to accommodate other existing criteria (e.g., maximin profit, minimax regret, Laplace, etc.)

Finally, we would like to relate our approach to the work of De Giacomo et al. [3], in which a special kind of ‘good’ programs, called *epistemically feasible deterministic* programs (EFDPs), was defined semantically and an epistemic deliberation operator for IndiGolog was developed. We think that, in general, the rational search operator presented here (i.e., Δ_{rat}) subsumes the one developed in [3] (i.e., Δ_e).

[9] δ_i^- is δ_i with the first action, i.e., pickup(paper), removed.

7. ACKNOWLEDGMENTS

The authors would like to thank Hector Levesque, Yves Lespérance, and Craig Boutilier for very helpful insights and discussions on the topic of this paper.

8. REFERENCES

[1] C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Workshop on Decision-Theoretic Planning (KR-2000)*, 2000.

[2] G. De Giacomo, Y. Lespérance, and H. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.

[3] G. De Giacomo, Y. Lespérance, H. Levesque, and S. Sardiña. On the semantics of deliberation in IndiGolog – from theory to implementation. *Annals of Mathematics and Artificial Intelligence*, 2003. To be submitted. An earlier version appeared in Proc. of KR-2002, pages 603–614.

[4] G. De Giacomo and H. Levesque. An incremental interpreter for high-level programs with sensing. In H. J. Levesque and F. Pirri, editors, *Logical Foundation for Cognitive Agents: Contributions in Honor of Ray Reiter*, pages 86–102. Springer, Berlin, 1999.

[5] S. French. *Decision Theory: An Introduction to the Mathematics of Rationality.* Ellis Horwood, 1986.

[6] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J. C. Meyer. Agent programming with declarative goals. In *Proc. of ATAL-00*, pages 228–243, 2001.

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

[8] Y. Lespérance, H. Levesque, F. Lin, and R. Scherl. Ability and knowing how in the situation calculus. *Studia Logica*, 66(1):165–186, 2000.

[9] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[10] R. Moore. A formal theory of knowledge and action. In J. Hobbs and R. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex, 1985.

[11] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logica computable language. In H. J. Levesque and F. Pirri, editors, *Agents Breaking Away (LNAI)*, volume 1038, pages 42–55. Springer-Verlag, 1996.

[12] R. Reiter. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems.* MIT Press, 2001.

[13] S. Russell. Rationality and intelligence. In R. Elio, editor, *Common Sense, Reasoning, and Rationality*, pages 37–59. Oxford University Press, 2002.

[14] S. Sardiña and S. Steven. Rational action in agent programs with prioritized goals (extended version). `www.cs.toronto.edu/~ssardina/papers/aamas03.ps`, 2003.

[15] R. Scherl and H. Levesque. The frame problem and knowledge-producing actions. In *Proc. of AAAI-93*, pages 689–695, 1993.

[16] S. Shapiro, Y. Lespérance, and H. Levesque. Goals and rational action in the situation calculus - A preliminary report. In *Working Notes of the AAAI Fall Symposium on Rational Agency: Concepts, Theories, Models, and Applications*, pages 117–122, 1995.

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

[18] M. Thielscher. Programming of reasoning and planning agents with FLUX. In *Proc. of KR-02*, pages 435–336, 2002.

[19] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative & procedural goals in intelligent agent systems. In *Proc. of KR-02*, 2002.

APPENDIX

A. PROOFS

A.1 Properties

Proof of Property 1. (i) Take a model M of (3) and σ_1, σ_2, m, n such that $m > n$ (the case where $m = n$ follows trivially) and $M \models P(\sigma_1, \sigma_2, n, s) \wedge \neg P(\sigma_2, \sigma_2, n, s)$. Then, since $m > n$, we know that $M \models \neg\forall n'.n' < m \wedge P(\sigma_1, \sigma_2, n', s) \supset P(\sigma_2, \sigma_1, n', s)$ (just take $n' = n$). Then, by (3), $M \models \neg P(\sigma_2, \sigma_1, m, s)$.

(ii) Take a model M of (3) and strategies σ_1, σ_2 such that there exists a k for which $M \models P^{\not\approx}(\sigma_1, \sigma_2, k, s)$ (that is, $M \models P(\sigma_1, \sigma_2, k, s) \wedge \neg P(\sigma_2, \sigma_1, k, s)$). Hence, $M \models \neg\forall n.P(\sigma_1, \sigma_2, n, s) \supset P(\sigma_2, \sigma_1, n, s)$, and, by definition of **AsGood**, $M \models \neg\mathbf{AsGood}(\sigma_2, \sigma_1, s)$ (*).

Also, given that $M \models P(\sigma_1, \sigma_2, k, s)$, we know that $M \models \forall n'.n' < k \wedge P(\sigma_2, \sigma_1, n', s) \supset P(\sigma_1, \sigma_2, n', s)$. By the first part of Property 1, $M \models \forall m.m \geq k \supset \neg P(\sigma_2, \sigma_1, m, s)$. These together imply that $M \models \forall n.P(\sigma_2, \sigma_1, n, s) \supset P(\sigma_1, \sigma_2, n, s)$, and therefore, $M \models \mathbf{AsGood}(\sigma_1, \sigma_2, s)$ (**).

Finally, from (*) and (**), we get that $M \models \mathbf{AsGood}^{\not\approx}(\sigma_1, \sigma_2, s)$. □

Proof of Property 2. Take a model M of (3) and σ_1, σ_2, s such that there exists a k for which $M \models (\forall j_1.j_1 < k \supset (\sigma_1 \succeq^s_{j_1} \sigma_2 \equiv \sigma_2 \succeq^s_{j_1} \sigma_1)) \wedge \sigma_1 \succ^s_k \sigma_2$. It follows by induction on n that $M \models \forall n.n < k \supset P(\sigma_1, \sigma_2, n, s) \equiv P(\sigma_2, \sigma_1, n, s)$. This, together with the fact that $M \models \sigma_1 \succeq^s_k \sigma_2$, implies that $M \models P(\sigma_1, \sigma_2, k, s)$. On the other hand, because $M \models \neg(\sigma_2 \succeq^s_k \sigma_1)$, $M \models \neg P(\sigma_2, \sigma_1, k, s)$ follows trivially from (3). Therefore, $M \models P^{\not\approx}(\sigma_1, \sigma_2, k, s)$, and, by the second part of Property 1, $M \models \mathbf{AsGood}^{\not\approx}(\sigma_1, \sigma_2, s)$ follows. □

A.2 Theorems

Proof of Theorem 1. $\Rightarrow$) Take a model M of $\mathcal{D} \cup \mathcal{H}$ such that $M \models \mathbf{AsGood}(\sigma_1, \sigma_2, s)$ for some strategies σ_1 and σ_2 and some situation s.

Suppose further that $M \models \neg\forall n.\sigma_2 \succeq^s_n \sigma_1 \supset \sigma_1 \succeq^s_n \sigma_2$. Thus, there should exists a $k \geq 1$ such that:

(a) $M \models \sigma_2 \succeq^s_k \sigma_1 \wedge \neg(\sigma_1 \succeq^s_k \sigma_2)$, i.e., $M \models \sigma_2 \succ^s_n \sigma_1$, and,

(b) $M \models \forall i.i < k \supset (\sigma_2 \succeq^s_i \sigma_1 \supset \sigma_1 \succeq^s_i \sigma_2)$.

Now, we know that $M \models \forall n.P(\sigma_2, \sigma_1, n, s) \supset P(\sigma_1, \sigma_2, n, s)$ holds because $M \models \mathbf{AsGood}(\sigma_1, \sigma_2, s)$. Due to (a) we know that $M \models \neg P(\sigma_1, \sigma_2, k, s)$ and, hence, it should be the case that $M \models \neg P(\sigma_2, \sigma_1, n, s)$. But, given that it is in fact the case that $\sigma_2 \succeq^s_k \sigma_1$, it has to be the case that $M \models \neg\forall n'.n' < k \wedge P(\sigma_1, \sigma_2, n', s) \supset P(\sigma_2, \sigma_1, n', s)$. This implies that there exists some $n' < k$ such that $M \models P(\sigma_2, \sigma_1, n', s)$, but $M \models \neg P(\sigma_2, \sigma_1, n', s)$. Ultimately, by the definition of P, this is to say that there exists an $n' < k$ such that $M \models \sigma_1 \succeq^s_k \sigma_2$, but $M \models \neg(\sigma_2 \succeq^s_{n'} \sigma_1)$, i.e., $M \models \sigma_1 \succ^s_{n'} \sigma_2$. This, together with (b) above, and the fact that $n' < k$, implies the “only-if” direction of the theorem.

$\Leftarrow$) Take a model M of $\mathcal{D} \cup \mathcal{H}$. It is trivial to see, by the definition of P, that $M \models \mathbf{AsGood}(\sigma_1, \sigma_2, s)$ whenever $M \models \forall n.\sigma_2 \succeq^s_n \sigma_1 \supset \sigma_2 \succeq^s_n \sigma_1$.

Suppose then that there exists a $k \geq 1$, two strategies σ_1 and σ_2 and a situation s such that:

(a) $M \models \sigma_1 \succeq^s_k \sigma_2 \wedge \neg(\sigma_2 \succeq^s_k \sigma_1)$, i.e., $M \models \sigma_1 \succ^s_n \sigma_2$, and,

(b) $M \models \forall i.i < k \supset (\sigma_2 \succeq^s_i \sigma_1 \supset \sigma_1 \succeq^s_i \sigma_2)$.

Because of (b), $M \models \forall n.n < k \supset P(\sigma_2, \sigma_1, n, s) \supset P(\sigma_1, \sigma_2, n, s)$ holds. However, because of (a), $M \models P(\sigma_1, \sigma_2, k, s)$ but $M \models \neg P(\sigma_2, \sigma_1, n, s)$, i.e., $M \models P^{\not\approx}(\sigma_1, \sigma_2, k, s)$. Finally, by applying the first part of Property 1, we know that $M \models \forall n.n \geq k \supset \neg P(\sigma_2, \sigma_1, n, s)$.

From all this, we conclude that $M \models \forall n.P(\sigma_2, \sigma_1, n, s) \supset P(\sigma_1, \sigma_2, n, s)$ and $M \models \mathbf{AsGood}(\sigma_1, \sigma_2, s)$ follows. □

Proof of Theorem 2. Shapiro et al. [16] restricted their attention to a particular set of strategies by assuming a special axiom in the background theory. The axiom states that *only* strategies which the agent can *always* follow (i.e., can follow starting in *any* initial situation) are taken into account. Let us recall the **Rational**(σ, s) definition from [16]:

$$\mathbf{Rational}(\sigma, s) \overset{\text{def}}{=} \forall \sigma'.\sigma' \succeq^R_s \sigma \supset \sigma \succeq^R_s \sigma',$$

where

$$\sigma \succeq^R_s \sigma' \overset{\text{def}}{=} \forall s'.K(s', s) \wedge H(\sigma_2, s') \supset H(\sigma_1, s').$$

As noted in Section 2, predicate $H(\sigma, s)$ is used to denote whether the path defined by σ starting at situation s satisfies the agent's (flat) goals. It is not hard to see that:

$$\mathcal{H} \models \forall \sigma_1, \sigma_2, s.\sigma_1 \succeq^s_1 \sigma_2 \equiv \sigma_1 \succeq^R_s \sigma_2. \qquad (4)$$

It is also not hard to see that the definition of $\mathcal{H}$ simplifies Theorem 1 as follows:

$$\mathcal{H} \models \forall \sigma_1, \sigma_2, s.\mathbf{AsGood}(\sigma_1, \sigma_2, s) \equiv (\sigma_2 \succeq^s_1 \sigma_1 \supset \sigma_1 \succeq^s_1 \sigma_2),$$

which together with (4) implies:

$$\mathcal{H} \models \forall \sigma_1, \sigma_2, s.\mathbf{AsGood}(\sigma_1, \sigma_2, s) \equiv (\sigma_2 \succeq_s^R \sigma_1 \supset \sigma_1 \succeq_s^R \sigma_2). \tag{5}$$

Let $M \models \mathcal{H} \cup \{(3), \forall \sigma, s.\mathbf{CanFollow}(\sigma, s)\}$. Then, $M \models \forall \sigma, s.\mathbf{Know}(\mathbf{CanFollow}(\sigma, \mathsf{now}), s)$, and the theorem follows from (5). $\square$

Proof of Theorem 3. Take a model M of (3), and suppose that for some formula ϕ, situation s, number $n \geq 1$, and strategies σ_ϕ and σ, the following holds:

$$M \models \mathbf{OGoal}(\mathbf{Eventually}(\phi), n, s) \wedge \mathbf{Achieve}(\phi, \sigma_\phi, s) \ \wedge \ \mathbf{RationalPrio}(\sigma, s) \wedge \neg\mathbf{Know}(\mathbf{Eventually}(\phi, \sigma, \mathsf{now}), s)$$

Suppose, however, that the following is true:

$$M \models \forall n'.n' < n \wedge P(\sigma, \sigma_\phi, n', s) \supset P(\sigma_\phi, \sigma, n', s) \tag{6}$$

Intuitively, this means that strategy σ is *not* strictly better than strategy σ_ϕ as far as the agent knows.

Given that $M \models \mathbf{Achieve}(\phi, \sigma_\phi, s)$, we know that $M \models \forall s'.K(s', s) \supset \mathbf{Eventually}(\phi, \sigma_\phi, s')$; that is, in every accessible situation from s, ϕ is achieved by executing σ_ϕ. Therefore, σ_ϕ dominates any other strategy w.r.t. level n alone, i.e., $M \models \forall \sigma'.\sigma_\phi \succeq_n^s \sigma'$, and, in particular, $M \models \sigma_\phi \succeq_n^s \sigma$. This, together with (6), implies that $M \models P(\sigma_\phi, \sigma, n, s)$.

On the other hand, given that $M \models \neg\mathbf{Know}(\mathbf{Eventually}(\phi, \sigma, \mathsf{now}), s)$, it is the case that $M \models \neg(\sigma \succeq_n^s \sigma_\phi)$. For, there is one accessible world in which σ_ϕ achieves ϕ, but σ does not. Thus, $M \models \neg P(\sigma, \sigma_\phi, n, s)$ holds.

Putting it all together, $M \models P(\sigma_\phi, \sigma, n, s) \wedge \neg P(\sigma, \sigma_\phi, n, s)$, and therefore $M \models \neg\mathbf{AsGood}(\sigma, \sigma_\phi, s)$. Moreover, because $M \models \mathbf{Achieve}(\phi, \sigma_\phi, s)$, $M \models \mathbf{Know}(\mathbf{CanFollow}(\sigma_\phi, \mathsf{now}), s)$ holds. Thus, it follows that $M \models \neg\mathbf{RationalPrio}(\sigma, s)$ which contradicts the initial assumption. Thus, there has to be an $n' < n$ such that $M \models P(\sigma, \sigma_\phi, n', s) \wedge \neg P(\sigma_\phi, \sigma, n', s)$ (i.e., $M \models P^{\not\approx}(\sigma, \sigma_\phi, n', s)$). Intuitively, this means that although, as far as the agent knows, σ may fail to achieve the level n goal ϕ, σ is still preferred to strategy σ_ϕ due to some more important objective. Otherwise, the agent would have chosen σ_ϕ as she knows it actually *guarantees* the achievement of goal ϕ. $\square$

Proof of Theorem 4. Take any model M of $\mathcal{R} \cup \mathcal{H}_2 \cup \mathcal{T}\{(3)\}$. Let σ_1 and σ_2 be two strategies and let s be a situation in the domain.

For convenience we will use $P(\sigma, \sigma', \leq t, s)$ as a shorthand for $\forall n.n \leq t \supset (P(\sigma', \sigma, n, s) \supset P(\sigma, \sigma', n, s))$.

We shall prove, by induction on $1 \leq k \leq r$, that $M \models P(\sigma_1, \sigma_2, \leq k, s)$ iff $M \models \mathit{AsGood}(\sigma_1, \sigma_2, \phi_1 > ... > \phi_k, s)$.

If $k = 1$, then $M \models P(\sigma_1, \sigma_2, \leq 1, s)$ iff $M \models P(\sigma_2, \sigma_1, 1, s) \supset P(\sigma_1, \sigma_2, 1, s)$. By definition of P, $M \models P(\sigma_1, \sigma_2, \leq 1, s)$ iff $M \models \sigma_2 \succeq_1^s \sigma_1 \supset \sigma_1 \succeq_1^s \sigma_2$. Since $\mathcal{H}_2 \models \forall \sigma, \sigma'.\sigma \succeq_1^s \sigma' \equiv \sigma \succeq_{\phi_1}^s \sigma'$, $M \models P(\sigma_1, \sigma_2, \leq 1, s)$ iff $M \models \sigma_2 \succeq_{\phi_1}^s \sigma_1 \supset \sigma_2 \succeq_{\phi_1}^s \sigma_1$. It follows from $\mathcal{R}$ that $M \models P(\sigma_1, \sigma_2, \leq 1, s)$ iff $M \models \mathit{AsGood}(\sigma_1, \sigma_2, \phi_1, s)$.

Assume next that the claim holds for $1 \leq k < r$. Now, it is not hard to see the following: $M \models P(\sigma_1, \sigma_2, \leq k+1, s)$ iff $M \models P(\sigma_1, \sigma_2, \leq k, s)$ and $M \models P(\sigma_2, \sigma_1, k+1, s) \supset P(\sigma_1, \sigma_2, k+1, s)$.

First, by the induction hypothesis, $M \models P(\sigma_1, \sigma_2, \leq k, s)$ iff $M \models \mathit{AsGood}(\sigma_1, \sigma_2, \phi_1 > ... > \phi_k, s)$.

Second, $M \models P(\sigma_2, \sigma_1, k+1, s) \supset P(\sigma_1, \sigma_2, k+1, s)$ iff either (i) $M \models \neg P(\sigma_2, \sigma_1, k+1, s)$; or (ii) $M \models P(\sigma_1, \sigma_2, k+1, s)$. Then, by splitting case (i), $M \models P(\sigma_2, \sigma_1, k+1, s) \supset P(\sigma_{k+1}, \sigma_2, k+1, s)$ iff either (i) $M \models \neg(\sigma_2 \succeq_{k+1}^s \sigma_1)$; or (ii) there exists an $m \leq k$ such that $M \models P(\sigma_1, \sigma_2, m, s) \wedge \neg P(\sigma_2, \sigma_1, m, s)$; or (iii) $M \models \sigma_1 \succeq_{k+1}^s \sigma_2 \wedge P(\sigma_1, \sigma_2, \leq k, s)$.

Notice that, due to the induction hypothesis, $M \models \neg P(\sigma_2, \sigma_1, \leq m, s)$ for some $m \leq k$ iff $M \models \neg \mathit{AsGood}(\sigma_2, \sigma_1, \phi_1 > ... > \phi_m, s)$.

Putting it all together, $M \models P(\sigma_1, \sigma_2, \leq k+1, s)$ iff $M \models \mathit{AsGood}(\sigma_1, \sigma_2, \phi_1 > ... > \phi_k, s)$ and one of the following holds:

(a) $M \models \neg(\sigma_2 \succeq_{k+1}^s \sigma_1)$;

(b) $M \models \neg \mathit{AsGood}(\sigma_2, \sigma_1, \phi_1 > ... > \phi_m, s)$, for some $m \leq k$;

(c) $M \models \sigma_1 \succeq_{k+1}^s \sigma_2$.

Next, it follows from $\mathcal{R}$ that $M \models \neg \mathit{AsGood}(\sigma_2, \sigma_1, \phi_1 > ... > \phi_m, s) \supset \neg \mathit{AsGood}(\sigma_2, \sigma_1, \phi_1 > ... > \phi_k, s)$. Moreover, if $M \models \mathit{AsGood}(\sigma_1, \sigma_2, \phi_1 > ... > \phi_k, s)$ we know, by induction, that $M \models P(\sigma_1, \sigma_2, \leq k, s)$. Similarly, if $M \models \neg \mathit{AsGood}(\sigma_2, \sigma_1, \phi_1 > ... > \phi_k, s)$, we know that $M \models \neg P(\sigma_2, \sigma_1, \leq k, s)$, which means that there exists an $m \leq k$ such that $M \models P(\sigma_1, \sigma_2, m, s) \wedge \neg P(\sigma_2, \sigma_1, m, s)$ (i.e., $M \models P^{\not\approx}(\sigma_1, \sigma_2, m, s)$). By Property 1, $M \models \neg P(\sigma_2, \sigma_1, k+1, s)$. Putting it all together, if $M \models \mathit{AsGood}(\sigma_1, \sigma_2, \phi_1 > ... > \phi_k, s)$ and $M \models \neg \mathit{AsGood}(\sigma_2, \sigma_1, \phi_1 > ... > \phi_k, s)$, then $M \models P(\sigma_1, \sigma_2, \leq k+1, s)$. As a result, we can replace point (b) above and obtain that $M \models P(\sigma_1, \sigma_2, \leq k+1, s)$ iff $M \models \mathit{AsGood}(\sigma_1, \sigma_2, \phi_1 > ... > \phi_k, s)$ and one of the following holds:

(a) $M \models \neg(\sigma_2 \succeq_{k+1}^s \sigma_1)$;

(b) $M \models \neg \mathit{AsGood}(\sigma_2, \sigma_1, \phi_1 > ... > \phi_k, s)$;

(c) $M \models \sigma_1 \succeq_{k+1}^s \sigma_2$.

This is to say, by the second axiom in $\mathcal{R}$, the following is true:

$$M \models P(\sigma_1, \sigma_2, \leq k+1, s) \text{ iff } M \models \mathit{AsGood}(\sigma_1, \sigma_2, \phi_1 > ... > \phi_{k+1}, s)$$

We have just proven, by induction, $M \models P(\sigma_1, \sigma_2, \leq k, s)$ iff $M \models \mathit{AsGood}(\sigma_1, \sigma_2, \phi_1 > ... > \phi_k, s)$, for all $1 \leq k \leq r$. The theorem follows from the fact that $M \models P(\sigma_1, \sigma_2, \leq r, s) \equiv \mathbf{AsGood}(\sigma_1, \sigma_2, s)$. $\square$

LEMMA 1. *Every strategy is induced by the universal program* $(\pi a.a)^*$. *Formally,*

$$\mathcal{D} \cup \mathcal{T} \models \forall \sigma, s.\mathbf{Induced}((\pi a.a)^*, \sigma, s)$$

PROOF. Take a model M of $\mathcal{D} \cup \mathcal{T}$, and σ, s, s^* such that $M \models \mathbf{OnPath}(\sigma, s, s^*)$.
If $s^* = s$, then $M \models Trans^*((\pi a.a)^*; \delta_{noOp}, s, (\pi a.a)^*; \delta_{noOp}, s)$ follows by reflexivity of $Trans^*$. Otherwise, if $s^* = do([a_1, ..., a_n], s)$ for some actions $a_1, ..., a_n$ $(n \geq 1)$, then we know that each action a_i is possible, i.e., $M \models Poss(a_i, do([a_1, .., a_{i-1}], s))$, for all $i = 1..n$. This is due to the fact that $s^* > s$ implies that all actions between s and s^* are executable (i.e., legal w.r.t. the precondition axioms). Hence, from the semantics of $Trans$ for non-deterministic choice of action, non-deterministic iteration, and sequence, it follows that $M \models Trans^*((\pi a.a)^*; \delta_{noOp}, s, ((nil; (\pi a.a)^*); \delta_{noOp}), s^*)$. $\square$

PROOF OF COROLLARY 1. It follows trivially from Theorem 4, and the fact every strategy is induced by the universal program (see Lemma 1). $\square$