# SR-APL: A Model for a Programming Language for Rational BDI Agents with Prioritized Goals

# (Extended Abstract)

Shakil M. Khan
Dept. of Computer Science and Engineering
York University, Toronto, Canada
skhan@cse.yorku.ca

Yves Lespérance
Dept. of Computer Science and Engineering
York University, Toronto, Canada
lesperan@cse.yorku.ca

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, languages and structures*

## General Terms

Theory, Languages

## Keywords

Agent programming languages with declarative goals, rationality, prioritized goals, reasoning about goals and goal dynamics

## 1. MOTIVATION

Recently, there has been much work on incorporating *declarative goals* in Belief-Desire-Intention Agent Programming Languages (e.g. [3]). In a BDI APL with declarative goals (APLwDG), declarative goals are used essentially for monitoring goal achievement and performing recovery when a plan has failed, performing rational deliberation, and reacting in a rational way to changes in goals that result from communication. While APLwDGs have evolved over the past few years, to keep them tractable and practical, they sacrifice some principles of rationality. In particular, while selecting plans to achieve a declarative goal, they ignore other concurrent intentions of the agent. As a consequence, the selected plans may be inconsistent with other intentions. Also, these APLwDGs typically rely on syntactic formalizations of declarative goals, whose properties are often not well understood.

**An Example** Consider a blocks world domain, where there are four blocks, one of each color, blue, yellow, red, and green. There is only a stacking action $stack(b, b')$: $b$ can be stacked on $b'$ in state $s$ if $b \neq b'$, both $b$ and $b'$ are *clear*, and $b$ is *on the table* in $s$. Assume that the agent initially has the following two goals: $\phi_1$, i.e. to eventually have a 2 blocks tower with a green block on top and a non-yellow block underneath, and $\phi_2$, i.e. to have a 2 blocks tower with a blue block on top and a non-red block underneath. Also, her plan library has only two rules: if she has the goal that $\phi_1$ and knows about a green block $b$ and a distinct non-yellow block $b'$ that are clear and are on the table, then she should adopt the plan of stacking $b$ on $b'$, and similarly for the goal that $\phi_2$. Thus according to this library, one way of building a green non-yellow (and a blue non-red) tower is to construct a green-blue (a blue-green, resp.) tower. While these two plans are individually reasonable, they are inconsistent with each other, since the agent has only one block of each color. Thus a rational agent should not adopt these two plans. However, it can be shown that a typical APLwDG agent (that does not consider the overall consistency of her intentions) may adopt these two plans together, and may make the other goal impossible by executing one of them. The problem arises in part because actions are not reversible in this domain, a common occurrence.

**Cite as:** SR-APL: A Model for a Programming Language for Rational BDI Agents with Prioritized Goals (Extended Abstract), Shakil M. Khan and Yves Lespérance, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. XXX-XXX.

In this paper, we develop logical foundations for a rational BDI agent programming framework with prioritized declarative goals that addresses these deficiencies of previous APLwDGs.

## 2. A SIMPLE RATIONAL APL (SR-APL)

**Our Formal BDI Framework** We use a variant of our logical framework for modeling prioritized goals, subgoals, and their dynamics [2], that is built on top of the situation calculus, and incorporates a (possible-worlds) model of knowledge. Here, an agent can have multiple *temporally extended goals* or *desires* at different priority levels. We have a possible-worlds semantics for these goals. We specify how goals evolve when actions/events occur and the agent's knowledge changes. We also define the agent's *intentions*, i.e. the goals that she is actively pursuing, in terms of this goal hierarchy. The framework in [2] is modified so that the agents are more committed to their intentions. They will only drop an intention when it is achieved, or when it becomes impossible or inconsistent with other higher priority intentions. We also model the relationship between goals and subgoals by ensuring that if $\psi$ is a subgoal of $\phi$, then $\psi$ (along with $\psi$'s subgoals, and theirs, etc.) is dropped when the parent goal $\phi$ is dropped or becomes impossible.

**Components of SR-APL** First of all, we have a *theory* $\mathcal{D}$ specifying actions that can be done, the initial knowledge and (*both declarative and procedural*) goals of the agent, and their dynamics, as discussed above. Moreover, we have a *plan library* $\Pi$ with rules of the form: if the agent has the intention that $\phi$ and knows that $\Psi$, then she should consider adopting the plan that $\sigma$. The *plan language* for $\sigma$ is a simplified version of ConGolog [1] and includes primitive actions, waiting for a condition, sequence, and the special action for subgoal adoption, $adoptRT(\diamond\Phi, \sigma)$; here $\diamond\Phi$ is a subgoal to be adopted and $\sigma$ is the plan *relative to* which it is adopted. While our BDI theory can handle arbitrary temporally extended goals, we focus on achievement and procedural goals exclusively.

**Semantics of SR-APL** We use a subset of ConGolog to specify the semantics of plans. Here, $\mathrm{Do}(\sigma)$ means that there is a terminating execution of program $\sigma$ starting in the current situation, $(\sigma_1 \| \sigma_2)$ denotes the concurrent composition of plans $\sigma_1$ and $\sigma_2$, and $\Gamma^{\|}$ refers to the concurrent composition of the plans in list $\Gamma$.

Specifying such a language raises some fundamental questions about rational agency, for instance: *what does it mean for a BDI* *agent to be committed to concurrently execute a set of plans next while keeping the option of further commitments to other plans open, in a way that does not allow procrastination?* An SR-APL agent can work on multiple goals at the same time, and thus can have multiple intended plans. One way of specifying an agent's commitment to execute a plan $\sigma$ next is to say that she has the intention that $\mathrm{Do}(\sigma)$. However, this does not allow for the interleaved execution of several plans, since Do requires that $\sigma$ be executed before any other actions/plans. A better alternative is for the agent to have the intention that $\mathrm{DoAL}(\sigma)$, i.e. to execute *at least* the program $\sigma$ next, and possibly more. $\mathrm{DoAL}(\sigma)$ holds if there is a terminating execution of program $\sigma$, possibly interleaved with other actions *by the agent herself*. However, a new problem with this approach is that it allows the agent to procrastinate, i.e. to perform actions that are unnecessary. To deal with this, we include an additional component, a *procedural intention-base* $\Gamma$, to an SR-APL agent. $\Gamma$ is a list of plans that the agent is currently actively pursuing. To avoid procrastination, we require that any action that the agent actually performs comes from $\Gamma$.

We have a two-tier transition system: *plan-level transition rules* specify how a plan may evolve, while *agent-level transition rules* specify how an SR-APL agent may evolve. The former are simply a subset of the ConGolog transition rules. Below, we discuss the latter. First of all, we have a rule $\mathrm{A}_{sel}$ for *selecting and adopting a plan* from the plan library $\Pi$ for some realistic (i.e. consistent with knowledge) goal $\diamond\Phi$ in the theory $\mathcal{D}$. It allows the agent to adopt a plan $\sigma$ as a subgoal of $\diamond\Phi$ (i.e. execute $adoptRT(\mathrm{DoAL}(\sigma), \diamond\Phi)$), provided that $\mathcal{D}$ entails that the agent does not intend not to adopt $\mathrm{DoAL}(\sigma)$ w.r.t. $\diamond\Phi$ next; our BDI theory ensures that if this is the case, then $\mathrm{DoAL}(\sigma)$ is indeed consistent with $\mathrm{DoAL}(\Gamma^{\|})$, and the agent intends to execute $\mathrm{DoAL}(\sigma \parallel \Gamma^{\|})$ afterwards.

Secondly, we have a transition rule $\mathrm{A}_{step}$ for *executing an intended action* from $\Gamma$. If a program $\sigma$ in $\Gamma$ can make a program-level transition in $s$ by performing a primitive action $a$ with program $\sigma'$ remaining in $do(a, s)$, and $\mathcal{D}$ entails that $\mathrm{DoAL}(\sigma)$ is a realistic goal at some priority level in $s$, then the agent may execute $a$, updating $\Gamma$ and $s$ accordingly, provided that the transition is consistent with the agent's intentions in the theory $\mathcal{D}$ in the sense that she does not have the intention not to execute $a$ in $s$.

Thirdly, we have a rule $\mathrm{A}_{exo}$ for *accommodating exogenous actions*, i.e. actions occurring in the agent's environment that are not under her control. Fourthly, we have a rule $\mathrm{A}_{clean}$ for *dropping adopted plans from the procedural goal-base* $\Gamma$ *that are no longer intended in the theory* $\mathcal{D}$ . This might be required when the occurrence of an exogenous action forces the agent to drop a procedural goal from $\mathcal{D}$ by making it impossible to execute or inconsistent with her higher priority realistic goals/plans. Our theory automatically drops such plans from the agent's goal-hierarchy specified by $\mathcal{D}$. Finally, we have a rule $\mathrm{A}_{rep}$ for *repairing an agent's plans in case she gets stuck*, i.e. when for all programs $\sigma$ in $\Gamma$, the agent has the realistic goal that $\mathrm{DoAL}(\sigma)$ at some level $n$ (and thus all of these $\mathrm{DoAL}(\sigma)$ are still individually executable and collectively consistent), but together they are not concurrently executable without some non-$\sigma$ actions, i.e. $\Gamma^{\|}$ has no program-level transition in $s$. This could happen as a result of an exogenous action. We can show that when the agent has complete information, there must be a repair plan available to the agent if her goals are consistent.

Another question that we face is: *how to ensure consistency between an agent's adopted declarative goals and adopted plans, given that some of the latter might be abstract, i.e. might be only partially instantiated in the sense that they include subgoals for which the agent has not yet adopted a (concrete) plan?* We deal with this using a weak notion of consistency that does not require the agent to expand all adopted goals while checking for consistency. For instance, $\mathrm{A}_{sel}$ above does not guarantee that there is an execution of the program $(\sigma \parallel \Gamma^{\|})$ *alone* after the $adoptRT$ action happens, but rather ensures that this program possibly along with additional actions by the agent is executable. Also, $\mathrm{A}_{step}$ requires that when the agent executes an action $a$ from a plan in $\Gamma$, $a$ must be consistent with her intentions in $\mathcal{D}$; but it does not require that she be willing to execute the remainder of $\Gamma$ next without any extra actions. Such a requirement would be too strong, given that $\Gamma$ may include abstract plans for which the agent has not yet adopted a subgoal. While our weak consistency check does not perform full lookahead over $\Gamma^{\|}$, our semantics ensures that any action performed by the agent must not make the concurrent execution of all the adopted plans possibly with other actions impossible. A side effect of our weak consistency check is that the agent might get stuck, and trigger the $\mathrm{A}_{rep}$ rule to repair her plans.

## 3. RATIONALITY OF SR-APL AGENTS

We have shown that some key rationality properties are satisfied by SR-APL agents. We only consider the case where exogenous actions are absent, as it's not obvious what rational behavior means in contexts where exogenous actions can occur.

For our blocks world example, we can show that our SR-APL agent behaves rationally in this domain. In particular:

• There exists a complete trace for our blocks world agent.

• All traces of the agent are terminating and end with the agent achieving all of her goals.

For any SR-APL agent (in the absence of exogenous actions), we can prove the following general properties:

• $\mathcal{D} \models \forall s.\ \neg\mathrm{Know}(false, s) \wedge \neg\mathrm{Int}(false, s)$, i.e. an agent's knowledge and intentions as specified by $\mathcal{D}$ must be consistent.

• The plans in $\Gamma$ and the declarative and procedural goals in $\mathcal{D}$ remain consistent. More precisely, for any configuration in a complete trace, either the goals in $\Gamma$ and those in $\mathcal{D}$ are consistent, or there is a future configuration along the trace where consistency is restored (by a finite number of applications of the $\mathrm{A}_{clean}$ rule).

• Our agents evolve in a rational way w.r.t. $\mathcal{D}$, i.e. if an SR-APL agent performs the action $a$ in situation $s$, then it must be the case that she does not have the intention not to execute $a$ next in $s$; moreover, if $a$ is performed via $\mathrm{A}_{step}$, then she indeed intends to execute $a$ possibly along with some other actions next; finally, if $a$ is the action of adopting a (sub)goal $\phi$, then she does not have the intention in $s$ not to bring about $\phi$ next.

## 4. CONCLUSION

Our framework combines ideas from the situation calculus-based Golog family of APLs, our expressive semantic formalization of prioritized goals, and work on BDI APLs. We ensure that an agent's intended declarative goals and adopted plans are consistent with each other and with her knowledge. We try to bridge the gap between agent theories and practical APLs by providing a model and specification of an idealized BDI agent whose behavior is closer to what a rational agent does. As such, it allows us to understand how compromises made during the development of a practical APLwDG affect the agent's rationality. In the future, we would like to investigate restricted versions of SR-APL that are practical.

## 5. REFERENCES

[1] G. De Giacomo, Y. Lespérance, and H.J. Levesque. ConGolog, a Concurrent Programming Language Based on the Situation Calculus. *Artificial Intelligence*, 121:109–169, 2000.

[2] S.M. Khan and Y. Lespérance. A Logical Framework for Prioritized Goal Change. In *Proc. AAMAS'10*, pp. 283–290, 2010.

[3] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative and Procedural Goals in Intelligent Agent Systems. In *Proc. KR'02*, pp. 470–481, 2002.