AgentSpeak(ER): An Extension of AgentSpeak(L) improving Encapsulation and Reasoning about Goals

Extended Abstract

Alessandro Ricci
DISI, University of Bologna
Cesena, Italy
a.ricci@unibo.it

Rafael H. Bordini
POLI-PUCRS
Porto Alegre, RS, Brazil
r.bordini@pucrs.br

Jomi F. Hübner
DAS, Federal University of Santa Catarina
Florianópolis, SC, Brasil
jomi.hubner@ufsc.br

Rem Collier
University College of Dublin
Dublin, Ireland
rem.collier@ucd.ie

ABSTRACT

In this paper we introduce AgentSpeak(ER), an extension of the AgentSpeak(L) language tailored to support encapsulation. The AgentSpeak(ER) extension aims at improving the style of BDI agent programming along relevant aspects, including program modularity and readability, failure handling, and reactive as well as goal-based reasoning.

CCS CONCEPTS

• **Computing methodologies → Intelligent agents**; • **Software and its engineering → Context specific languages**;

KEYWORDS

Agent-Oriented Programming; Agent Programming Languages; BDI; AgentSpeak(L); Jason; ASTRA; AgentSpeak(ER)

ACM Reference Format:
Alessandro Ricci, Rafael H. Bordini, Jomi F. Hübner, and Rem Collier. 2018. AgentSpeak(ER): An Extension of AgentSpeak(L) improving Encapsulation and Reasoning about Goals. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 INTRODUCTION

AgentSpeak(L) was introduced in [8] with the purpose of defining an expressive, abstract language capturing the main aspects of the Belief-Desire-Intention architecture [3, 6], featuring a formally defined semantics and an abstract interpreter. The starting point to define the language was implemented systems such as the Procedural Reasoning System (PRS) [7]. Various Agent Programming Languages extended AgentSpeak(L) with constructs and mechanisms making it practical from a programming point of view [2, 4].

Along this line, this paper describes a novel extension of the AgentSpeak(L) model — called AgentSpeak(ER) — featuring *plan encapsulation*, i.e. the possibility to define plans that fully encapsulate the strategy to achieve the corresponding goals, integrating both pro-active and reactive behaviour.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

2 MOTIVATION

The main motivation behind AgentSpeak(ER) comes from the experience using agent programming languages based on the AgentSpeak(L) model, Jason and ASTRA in particular. Yet, these issues are relevant for any language based on the BDI architecture.

In the BDI model, plans are meant to specify some means by which an agent can satisfy an end [8]. In AgentSpeak(L), a plan consists of a rule of the kind e : c <- b. The head of a plan consists of a triggering event e and a context c. The triggering event specifies why the plan was triggered, i.e., the addition or deletion of a belief or goal. In the following, we refer to plans triggered by event goals as *g-plans*, and plans triggered by belief change (including percepts) as *e-plans*. The context specifies what should hold given the agent's current mental state if the plan is to be triggered. The body of a plan is a sequence of actions or (sub-)goals.

In this approach — as well as in planning, in general — the *means* to achieve a goal (i.e., the plan body) is meant to be fully specified in terms of the actions the agent should execute and the (sub-)goals the agent should achieve or test. In practice, when programming such systems, it is often the case that the strategy (the means) adopted to achieve some goal (the end) naturally includes ractions — i.e., reacting to events asynchronously perceived from the environment, including changes in the beliefs. This reflects more than just the ability of an agent to change/adapt its course of actions; it allows the integration of reactivity as a core ingredient of the strategy to achieve some goal. This revised notion of a plan is not just a programming feature; it also occurs naturally in human activity. For example, a fisherman with the goal of catching fish waits for the event of a tug on their line indicating a fish is on the hook. Reactivity is a key ingredient of many activities that we perform to achieve specific goals, not only to handle events that represents errors or unexpected situations (for the current courses of actions). It follows that this is also an opportunity to extend the plan model so as to fully *encapsulate* reactions that are part of the strategy to achieve the goal, as well as the subgoals that are specific to that particular goal.

Let us consider the robot cleaning example used to describe plans in [8]. One of the plans is:

```
+location(waste,X) : location(robot,X) & location(bin,Y)
  <- pick(waste); !location(robot,Y); drop(waste).
```

That is, as soon as the robot perceives that there is waste at its location, then it can pick it up and bring it to the bin. This e-plan is an essential brick of the overall strategy to achieve the goal of cleaning the environment. The problem here is that it is an implicit rather than explicit goal of the agent (since it is an e-plan, it is executed regardless of the agent currently having the goal of keeping the environment clean). In practice, we adopt a maintenance goal [5] to clean the environment, which includes reacting to cleaning up waste when we see it. In the above program, this notion cannot be represented and remains in the mind of the programmer/designer; as there is no g-plan for it, there is no explicit trace in the agent mental structures about this goal. It is the ability to explicitly implement this type of behaviour that motivates the work presented in this paper.

3 A TASTE OF AGENTSPEAK(ER)

AgentSpeak(ER) extends the plan model of AgentSpeak(L) beyond the simple sequence of actions and goals, so as to *(i)* include the possibility of specifying reactive behaviour encapsulated within the plan, by means of e-plans; *(ii)* require that e-plans are always defined within the scope of some g-plan (i.e., reactions occur always in the context of some explicit designed goal to be achieved). The AgentSpeak(ER) syntax for plans extends AgentSpeak(L) as follows:

```
+!goal : context <: goal_cond {
  <- ... // main sequence (body actions)

  /* encapsulated e-plans */
  +e1 : c1 <- ...
  +e2 : c2 <- ....

  /* encapsulated g-plans */
  +!g1 {
    <- ...
    +e3 : c3 <- ... // possible old-style plans
  }

  /* encapsulated plans catching failures */
  -!g1 : ... <- ... // handles failures in pursuing g1
}
```

A plan becomes the *scope* of (*i*) a sequence of actions (referred as *body actions*), (*ii*) a set of *e-plans*, specifying a reactive behaviour which is active at runtime only when the goal is being pursued, and (*iii*) a set of *g-plans*, specifying plans to achieve subgoals that are relevant only in the scope of that plan. The e-plans and g-plans are referred to as *sub-plans*. The sub-plans may include also reactions to failures occurring when the plan is executed.

The robot cleaning example becomes:

```
+!clean_env  {
   +location(waste,X) : location(robot,X) & location(bin,Y)
      <- pick(waste); !location(robot,Y); drop(waste). }
```

We can give an explicit reason for the reactive behaviour by encapsulating the e-plan within a g-plan, with an explicit goal `clean_env`. This is also a particular case where the body of the g-plan happens not to have any actions.

Informal semantics of the extended plan model:

- The sub-plans are part of the strategy which can be applied to achieve the g-plan. The main sequence (body actions) can be empty — this is typical of purely reactive behaviour. If an event matches the triggering event of a sub-plan while the g-plan is executing (i.e., the main sequence is in execution, and the sub-plan is applicable according to the context), then the body of the sub-plan is stacked on top of the stack of the current (g-plan related) intention. The effect is like an asynchronous interruption of the main sequence, to execute the body of the sub-plan first.
- The goal is considered achieved if/when the condition described by the `goal_cond` expression is met. If `goal_cond` is not specified, by default the condition is that all actions of the main sequence have been executed and completed, in compatibility with the AS model, unless the case in which the main plan body is empty: in that case the condition is `false` by default.
- The body of the plan provides a syntactical and runtime scope of the sub-plans, that is: variables used in the goal/context expressions are visible also to sub-plans; the lifetime/availability of the sub-plans is limited to the time in which the g-plan is in execution.
- Failures generated by either the main sequence or by sub-plans generate a `-!g` event.
- An AgentSpeak(L) plan `+!g : c <- a.` is an AgentSpeak(ER) g-plan with no sub-plans.

This extension turns out to bring a number of important benefits to agent programming based on the BDI model, namely:

- improving the overall readability of the agent source code, reducing fragmentation and increasing modularity;
- promoting a cleaner goal-oriented programming style by enforcing encapsulation of reactive plans within goal plans while still permitting purely reactive behaviour;
- improving intention management, enforcing a one-to-one relation between intentions and goals — so every intention is related to a single (top-level) goal;
- improving failure handling, to support management of failures related to plans for reacting to environment events.

4 FINAL REMARKS

We formalised the main changes required in the existing formal semantics of AgentSpeak(L) and experimentally evaluated on initial prototype implementations of AgentSpeak(ER), on top of the ASTRA and Jason platforms[1]. Results will be described in a longer version of this paper.

As with any new programming language, there is much future work, including improving the prototype implementations and comparing performances. More generally, full understanding and evaluation of a programming language takes many years. We expect in the long term to use AgentSpeak(ER) in the practical development of multi-agent systems, both for real-world as well as academic systems (e.g., for the multi-agent programming contest [1]). However, besides the actual programming practice, we

[1] Available at https://github.com/agentspeakers.

expect AgentSpeak(ER) to contribute to formal work as well. Assessing how formal verification of AgentSpeak(ER) systems compares to the original language is also planned as future work.

REFERENCES

[1] Tobias Ahlbrecht, Niklas Fiekas, and Jürgen Dix. 2018. Multi-Agent Programming Contest 2016. *International Journal of Agent Oriented Software Engineering* (2018). To appear.

[2] Rafael Bordini and Jomi Hübner. 2006. BDI Agent Programming in AgentSpeak Using Jason. In *CLIMA VI*, Francesca Toni and Paolo Torroni (Eds.). LNAI, Vol. 3900. Springer, 143–164.

[3] Michael E. Bratman, David J. Israel, and Martha E. Pollack. 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence* 4 (1988), 349–355. https://doi.org/10.1111/j.1467-8640.1988.tb00284.x

[4] Rem W. Collier, Sean Edward Russell, and David Lillis. 2015. Reflecting on Agent Programming with AgentSpeak(L). In *PRIMA 2015: Principles and Practice of Multi-Agent Systems - 18th International Conference, Bertinoro, Italy, October 26-30, 2015, Proceedings (Lecture Notes in Computer Science)*, Qingliang Chen, Paolo Torroni, Serena Villata, Jane Yung-jen Hsu, and Andrea Omicini (Eds.), Vol. 9387. Springer, 351–366.

[5] Simon Duff, James Harland, and John Thangarajah. 2006. On Proactivity and Maintenance Goals. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '06)*. ACM, New York, NY, USA, 1033–1040. https://doi.org/10.1145/1160633.1160817

[6] Michael P. Georgeff and Amy L. Lansky. 1987. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 2 (AAAI'87)*. AAAI Press, 677–682. http://dl.acm.org/citation.cfm?id=1863766.1863818

[7] Francois F. Ingrand, Michael P. Georgeff, and Anand S. Rao. 1992. An Architecture for Real-Time Reasoning and System Control. *IEEE Expert: Intelligent Systems and Their Applications* 7, 6 (Dec. 1992), 34–44. https://doi.org/10.1109/64.180407

[8] Anand S. Rao. 1996. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In *Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, January 22-25, 1996, Proceedings (Lecture Notes in Computer Science)*, Walter Van de Velde and John W. Perram (Eds.), Vol. 1038. Springer, 42–55.