Plan Patterns for Declarative Goals in AgentSpeak

Jomi Fred Hübner — University of Blumenau (Brazil)
Rafael H. Bordini — University of Durham (UK)
Michael Wooldridge — University of Liverpool (UK)

Motivation

- The AgentSpeak(L) programming language [2] is based on logic programming and provides an elegant abstract framework for programming BDI agents.
- An AgentSpeak agent is defined by a set of beliefs (ground literals) which constitute the agents initial belief base, and a set of plans which form its plan library.
- The language lacks some practical features for multi-agent systems development.
- Goals in AgentSpeak are normally procedural goals; they trigger plans defining courses of actions to achieve that goal:

```
+!loc(X,Y) : battery_charged <- go(X,Y).
```

- Agent-based development also needs:
  - Declarative goals: a goal g is considered achieved when the agent believes g.
  - Agent commitment to goals: e.g., blind × open-minded commitment

Objectives

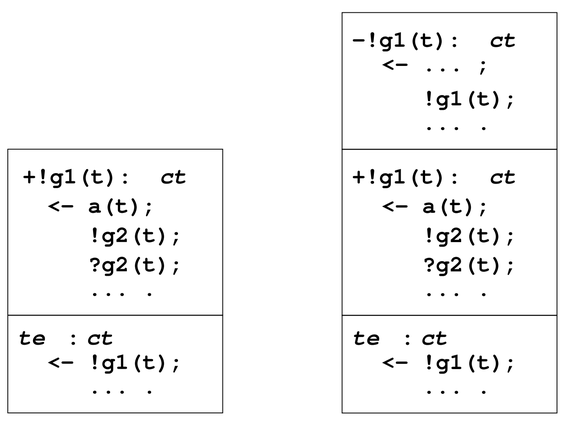
- To allow AgentSpeak programmers to use declarative goals with sophisticated temporal structures and define various forms of commitments towards goals:
  - without changing the AgentSpeak syntax or semantics
  - without adding a separate declarative goal base to the agent's architecture
- Our approach is to:
  - Define declarative goals and forms of commitments to them by using combinations of standard AgentSpeak plans (plus some feature already available in Jason)
  - Elaborate patterns to help programmers translate high-level declarative goals into AgentSpeak

Relevant *Jason* Features

1. Failure handling

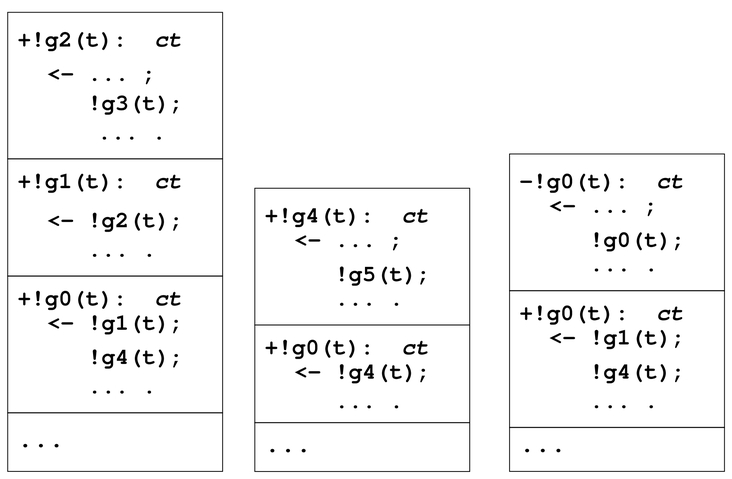
When a plan such as `+!g : b <- a` fails:

- the `-!g` event is created; and
- a plan to handle the failure is triggered.



An Intention before Plan Failure After Plan Failure

2. .dropGoal internal action to fail/succeed a goal



Initial Intention .dropGoal(g1(t),true) .dropGoal(g1(t),false)

Declarative Goals

1. Simple Declarative Goal

The agent should believe g when the plan finishes.

```
+!g : c <- p; ?g.
```

If the agent has a plan with context c and body p to achieve g, we add `?g` (a test goal) at the end of the plan to ensure that the agent believes g when the plan is finished; otherwise the plan will fail.

Example:

```
+!loc(X,Y)
   : battery_charged
   <- go(X,Y);
      ?l(X,Y).
```

Pattern:

```
+!g : c1 <- p1.
+!g : c2 <- p2.
...
+!g : cn <- pn.
```

DG$_g$ $(n \geq 1)$

```
+!g : g <- true.
+!g : c1 <- p1; ?g.
+!g : c2 <- p2; ?g.
...
+!g : cn <- pn; ?g.
+g : true <- .dropGoal(g,true).
```

2. Backtracking Declarative Goal

In this type of goal, when a plan fails to achieve the goal, an alternative plan (if available) is selected; $\mathcal{P}$ stands for a set of plans for g.

$\mathcal{P}$

BDG$_g$

DG$_g(\mathcal{P})$

```
-!g : true <- !g.
```

Example:

```
+!loc(X,Y) : true <- go(X,Y).
+!loc(X,Y) : true <- go(X1,Y1); go(X,Y).
```

BDG$_{\texttt{loc(X,Y)}}$

```
+!loc(X,Y) : loc(X,Y) <- true.
+!loc(X,Y) : true <- go(X,Y); ?loc(X,Y).
+!loc(X,Y): true
   <- go(X1,Y1); go(X,Y); ?loc(X,Y).
+loc(X,Y) : true <- .dropGoal(loc(X,Y),true).
-!loc(X,Y) : true <- !loc(X,Y).
```

3. Blind Commitment

Even if at some point there are no applicable plans, a blindly commited agent will continue to pursue the goal.

$\mathcal{P}$

BCG$_g$

BDG$(\mathcal{P})$

```
+!g : true <- !g.
```

Example:

```
+!loc(X,Y) : true <- go(X,Y).
+!loc(X,Y) : true <- go(X1,Y1); go(X,Y).
```

BCG$_{\texttt{loc(X,Y)}}$

```
+!loc(X,Y) : loc(X,Y) <- true.
+!loc(X,Y) : true <- go(X,Y); ?loc(X,Y).
+!loc(X,Y): true
   <- go(X1,Y1); go(X,Y); ?loc(X,Y).
+loc(X,Y) : true <- .dropGoal(loc(X,Y),true).
-!loc(X,Y) : true <- !loc(X,Y).
+!loc(X,Y) : true <- !loc(X,Y).
```

4. Open-Minded Commitment

- Requires an agent to be attentive to:
  - Failure condition f: if f is believd, the goal becomes impossible to achieve, thus the intention should be dropped with failure.
  - Motivation condition m: if the belief that is the motivation for the goal no longer holds, the intention can be dropped with success.

$\mathcal{P}$

OMC$_{g,f,m}$

BCG$_g(\mathcal{P})$

```
+f : true <- .dropGoal(g,false).
-m : true <- .dropGoal(g,true).
```

Example:

```
+!loc(X,Y) : true <- go(X,Y).
+!loc(X,Y) : true <- go(X1,Y1); go(X,Y).
```

OMC$_{\texttt{loc(X,Y)}}$

```
+!loc(X,Y) : loc(X,Y) <- true.
+!loc(X,Y) : true <- go(X,Y); ?loc(X,Y).
+!loc(X,Y): true
   <- go(X1,Y1); go(X,Y); ?loc(X,Y).
+loc(X,Y) : true <- .dropGoal(loc(X,Y),true).
-!loc(X,Y) : true <- !loc(X,Y).
+!loc(X,Y) : true <- !loc(X,Y).
+obstacle(X,Y) : true <- .dropGoal(loc(X,Y),false).
-gold(X,Y) : true <- .dropGoal(loc(X,Y),true).
```

5. Maintenance Goal

The agent needs to ensure that the state of the world is such that g holds (or is believed to hold).

$\mathcal{P}$

MG$_g$

```
g.
-g : true <- !g.
```

BCG$(\mathcal{P})$

Example:

```
+!battery_charged: energy_station(X,Y)
   <- go(X,Y); plugin.
```

MG$_{battery\_charged}$

```
battery_charged.
-battery_charged : true <- !battery_charged.

+!battery_charged : battery_charged <- true.
+!battery_charged: energy_station(X,Y)
   <- go(X,Y); plugin; ?battery_charged.
+battery_charged:true
   .dropGoal(battery_charged,true).
-!battery_charged : true <- !battery_charged.
+!battery_charged : true <- !battery_charged.
```

Conclusions and Related Work

- Implementation of sophisticated types of goals and commitments in AgentSpeak.
- Definition of these goals by means of patterns.
- The language is kept simple with no changes to syntax or semantics.

- Most related work changed the semantics of the language to include declarative goals [5, 1].
- Some approaches require a "goal base" [3].
- Work by van Riemsdijk *et al.* [4] uses a similar approach to define declarative goals based on procedural goals for 3APL. Our work goes much further, adding new types of goals and commitments, with the flexibility of the use of *patterns*.

References

[1] L. Braubach, A. Pokahr, W. Lamersdorf, and D. Moldt. Goal representation for BDI agent systems. *Proc. of ProMAS 2004*, pages 9–20, 2004.

[2] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. *Proc. of MAAMAW'96*, LNAI 1038, pp. 42–55, London, 1996. Springer-Verlag.

[3] B. van Riemsdijk, M. Dastani, and J.-J. C. Meyer. Semantics of declarative goals in agent programming. *Proc. of AAMAS 2005*, pages 133–140. ACM, 2005.

[4] M. B. van Riemsdijk, M. Dastani, and J.-J. C. Meyer. Subgoal semantics in agent programming. *Proc. of EPIA 2005*, LNCS 3808, pp. 548–559, 2005.

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

Full paper available in the proceedings of DALT-2006. J.F.Hübner, R.H.Bordini, M.Wooldridge; Programming Declarative Goals Using Plan Patterns.