Programming Declarative Goals Using Plan Patterns

Jomi F. Hübner[1], Rafael H. Bordini[2], Michael Wooldridge[3]

[1]University of Blumenau (Brazil)
jomi@inf.furb.br

[2]University of Durham (UK)
R.Bordini@durham.ac.uk

[3]University of Liverpool (UK)
mjw@csc.liv.ac.uk

DALT 2006 – Hakodate, Japan

Outline

Context

- The original AgentSpeak(L) logic-based programming language provides an elegant abstract framework for programming BDI agents
- Various extensions needed to make it a practical programming language (hence the creation of ***Jason***)
- AgentSpeak has only **procedural goals**: plans are courses of action to achieve goals

Example (robot location)

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

- Agent-based development also requires:
  - **Declarative goals**: a goal *g* is considered achieved when the agent believes in *g*.
  - Agent **commitment** to goals: e.g., blind and open-minded forms of commitment

Objectives

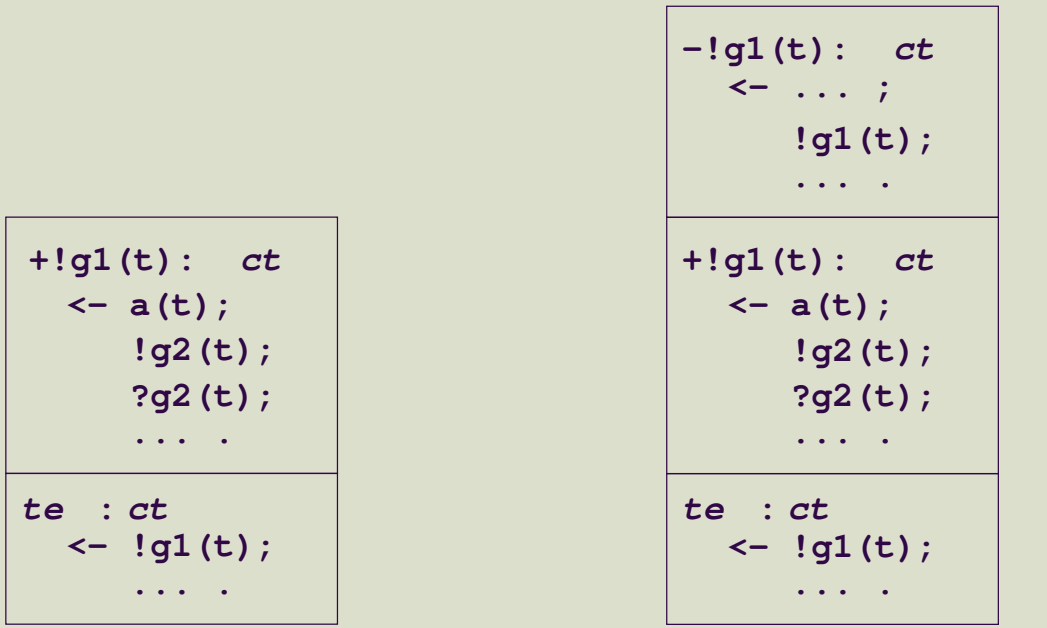
- To allow AgentSpeak programmers to use declarative goals and commitments (goals with complex temporal structures)
  - without changing the AgentSpeak syntax or semantics
  - without adding a separate “goal base”
- Our approach is to
  - define declarative goals and forms of commitments to them by using combinations of standard AgentSpeak plans (plus some feature already available in ***Jason***)
  - elaborate **patterns** to help programmers translate high-level declarative goals into AgentSpeak

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.

Example

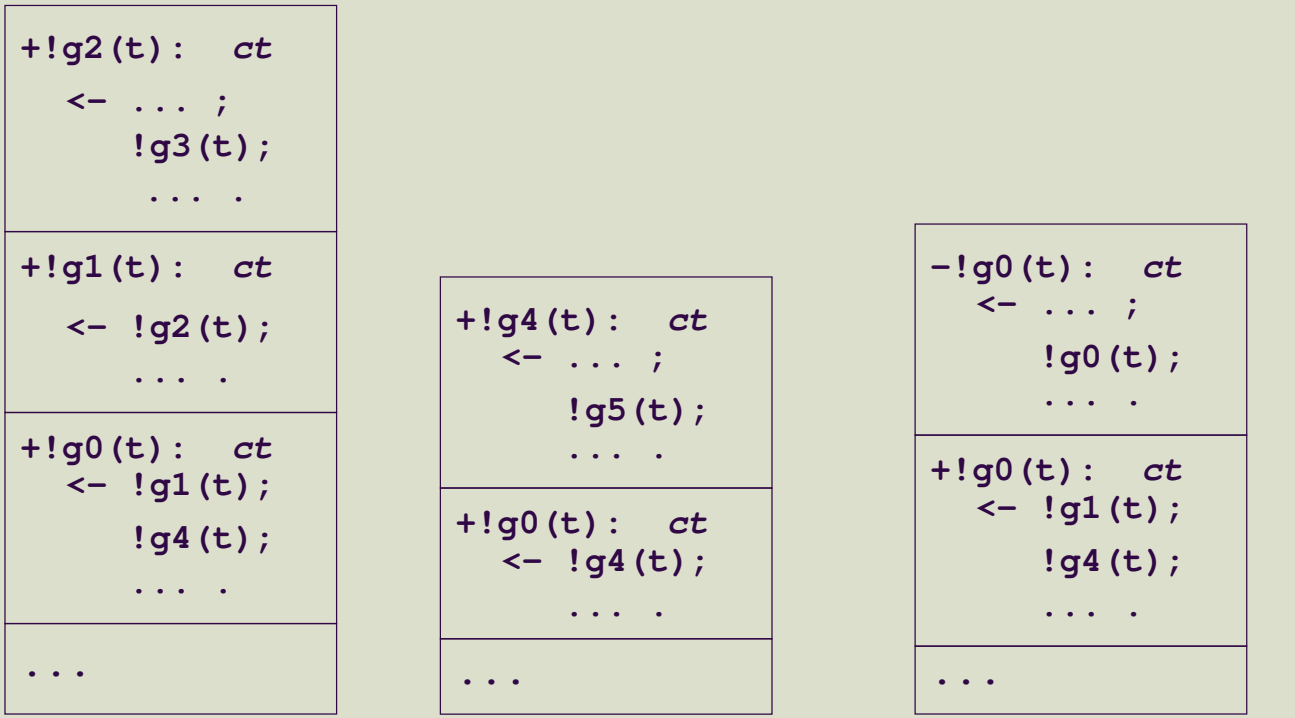


An Intention before Plan Failure After Plan Failure

Dropping goals

- `.dropGoal` internal action to fail/succeed a goal

Example



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

Definition (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) to ensure 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).
```

Declarative Goal (DG) Pattern

$$\frac{\begin{array}{l}+!g \ : \ c_1 \ \texttt{<-} \ p_1 . \\ +!g \ : \ c_2 \ \texttt{<-} \ p_2 . \\ \ldots \\ +!g \ : \ c_n \ \texttt{<-} \ p_n .\end{array}}{\begin{array}{l}+!g \ : \ g \ \texttt{<-} \ \texttt{true}. \\ +!g \ : \ c_1 \ \texttt{<-} \ p_1 ; \ ?g . \\ +!g \ : \ c_2 \ \texttt{<-} \ p_2 ; \ ?g . \\ \ldots \\ +!g \ : \ c_n \ \texttt{<-} \ p_n ; \ ?g . \\ +g \ : \ \texttt{true} \ \texttt{<-} \ \texttt{.dropGoal(}g\texttt{, true).}\end{array}} \ \mathbf{DG}_g \ \ (n \geq 1)$$

Definition (Backtracking Declarative Goal – BDG)

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

$$\frac{\mathcal{P}}{\mathbf{DG}_g\,(\mathcal{P})}\ \mathbf{BDG}_g$$

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

Example

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

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

```
+!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).
```

Definition (Blind Commitment Goal – BCG)

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

$$\frac{\mathcal{P}}{\textbf{\textit{BDG}}(\mathcal{P})}\ \textbf{BCG}_g$$

```
+!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).
```

Open-Minded Commitment (OMC) I

- Requires an agent to be attentive to:
  - Failure condition *f*: if *f* is believed, 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.

Open-Minded Commitment (OMC) II

Definition (OMC)

$$\frac{\mathcal{P}}{\mathbf{BCG}_g(\mathcal{P})}\ \mathbf{OMC}_{g,f,m}$$

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

Open-Minded Commitment (OMC) III

Example

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

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

```
+!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).
+obst(X,Y) : true <- .dropGoal(loc(X,Y), false).
-gold(X,Y) : true <- .dropGoal(loc(X,Y), true).
```

Definition (Maintenance Goal – MG)

The agent needs to ensure that the state of the world is such that g holds (or is believed to hold).
More specifically, this pattern is used to activate the associated achievement goal when the agent realises it has already failed in regards to a maintenance goal.

$$\frac{\mathcal{P}}{\mathbf{MG}_g}$$

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

$\mathbf{BCG}(\mathcal{P})$

Example

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

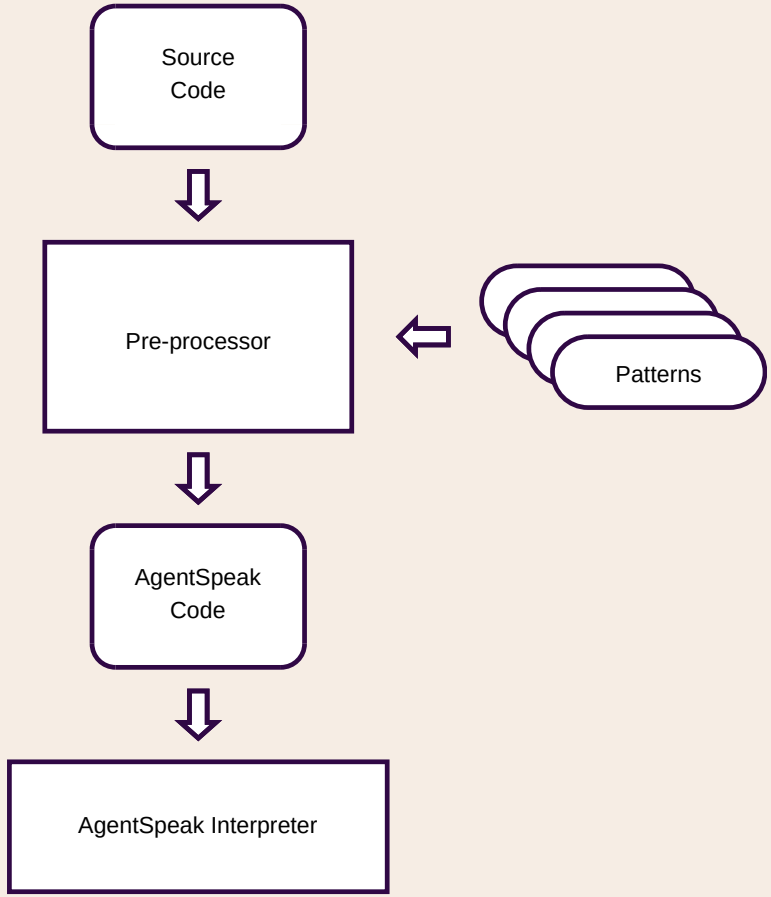
$\mathbf{MG}_{\texttt{battery\_charged}}$

```
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.
```

Pre-processing for *Jason*

- ***Jason*** is an interpreter for an extended version of AgentSpeak and is available *Open Source* under GNU LGPL at **`http://jason.sf.net`**.
- It incorporates a pre-processor that simplifies the use of patterns (not yet in the latest public release).



Example (Open-Minded)

```
{ begin
omc("loc(X,Y)","obstacle(X,Y)","gold(X,Y)") }

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

{ end }
```

Example (Maintenance Goal)

```
{ begin mg("battery_charged") }

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

{ end }
```

Summary

- Implementation of sophisticated types of goals and commitments in AgentSpeak.
- Definition of these goals by means of **patterns**.
- The language is kept “neat”, with no changes to syntax or semantics.
- Pre-processing facilitates the use of patterns.
- Programmers can take advantage of the flexibility of patterns, changing existing ones or creating their own.

Related work

- Most related work changed the syntax and/or semantics of the language to include declarative goals, e.g.:
  - M. Winikoff *et al.* Declarative and procedural goals in intelligent agent systems.
  - L. Braubach *et al*. Goal representation for BDI agent systems.
- Some approaches used (also) a “goal base”:
  - B. van Riemsdijk *et al.* Semantics of declarative goals in agent programming.
- Work by van Riemsdijk *et al.* (Subgoal semantics in agent programming) already used the same approach to define declarative goals based on procedural goals for 3APL. Our work goes further, adding new types of goals and commitments, with the flexibility of the use of **patterns**.