Multiagent Systems

3b. Jason: Handling plan failure

B. Nebel, C. Becker-Asano, S. Wölfl

Albert-Ludwigs-Universität Freiburg

May 16, 2014

Recommended reading

Bordini, Hübner, Wooldridge, "Programming multi-agent systems in AgentSpeak using Jason", Wiley, 2007

3b.1 Handling plan failure

AgentSpeak & plan failure

So far, we only learned how to design plans in AgentSpeak. Their bodies can contain any combination of:

- **actions** (internal or external)
- **subgoals** (achievement or test)
- **mental notes**
- **expressions**

But what happens if a plan fails?
⇒ Well, it depends ...

Plan failure example

Consider the following simple agent:

```
!g1. // initial goal

@p1 +!g1    : true <- !g2(X); .print("end g1 ",X).
@p2 +!g2(X) : true <- !g3(X); .print("end g2 ",X).
@p3 +!g3(X) : true <- !g4(X); .print("end g3 ",X).
@p4 +!g4(X) : true <- !g5(X); .print("end g4 ",X).
@p5 +!g5(X) : true <- .fail.

@f1 -!g3(failure) : true <- .print("in g3 failure").
```

Initial goal g1 creates subgoal g2 ⇒ creates g3 ⇒ creates g4 ⇒ creates g5 ⇒ g5 fails.
Only g3 has a plan to handle failure.

Plan failure example

When agent starts running:

1. Only p1 is **relevant** (*triggering event* matches) and **applicable** (*context* true)
2. New intention created, leads to event +!g2(X)
3. Plan p2 is selected to handle this

The intention stack has two **intended means**:

```
+!g2(X)
  <-  !g3(X); .print("end g2 ", X).
```

```
+!g1
  <-  !g2(X); .print("end g1 ", X).
```


Plan failure example

This execution goes on until activation of g5, then the intention stack looks like this:

```
+!g5(X)
  <-  .fail.
```

```
+!g4(X)
  <-  !g5(X); .print("end g4 ", X).
```

```
+!g3(X)
  <-  !g4(X); .print("end g3 ", X).
```

```
+!g2(X)
  <-  !g3(X); .print("end g2 ", X).
```

```
+!g1
  <-  !g2(X); .print("end g1 ", X).
```

Plan failure example

When g5 fails, interpreter looks for goal in intention stack that has relevant failure plan ⇒ only `!g3(x)`
Event `-!g3(X)` is created, which results in execution of `@f1`:

```
-!g3(failure)                                {X ↦ failure}
  <-  .print("in g3 failure").
```

```
+!g5(X)
  <-  .fail.
```

```
+!g4(X)
  <-  !g5(X); .print("end g4 ", X).
```

```
+!g3(X)
  <-  !g4(X); .print("end g3 ", X).
```

```
+!g2(X)
  <-  !g3(X); .print("end g2 ", X).
```

```
+!g1
  <-  !g2(X); .print("end g1 ", X).
```


Plan failure example

When `@f1` finishes, intention continues in the same way as if g3 had been **achieved successfully**! ⇒ second formula of plan for `+!g2` is executed:

```
+!g2(X)                                      {X ↦ failure }
  <-  .print("end g2 ", X).
```

```
+!g1
  <-  !g2(X); .print("end g1 ", X).
```

The resulting output:

```
[a] saying: in g3 failure
[a] saying: end g2 failure
[a] saying: end g1 failure
```


Summary

In essence:

When no **applicable plan** for a goal *deletion* (or external events like *belief addition* or *deletions*) exists, the interpreter can do two things:

- discard whole intention
- repost the event by including it again in the event queue

This can be configured in the `*.mas2j` file.
`agents: <ag_type1_name> <source_file> <options>`:

- **events**: **discard**, requeue, retrieve
- **intBel**: **sameFocus**, newFocus
- **nrcbp**: **n**umber of **r**easoning **c**ycles **b**efore **p**erception
- **verbose**: number between 0 and 2
- **user settings**:
  e.g. `... agents: ag1 [file="an.xml",value=45]`