University of London

Composing High-Level Plans for Declarative Agent Programming

Felipe Meneguzzi &

Michael Luck

King's College London

Outline

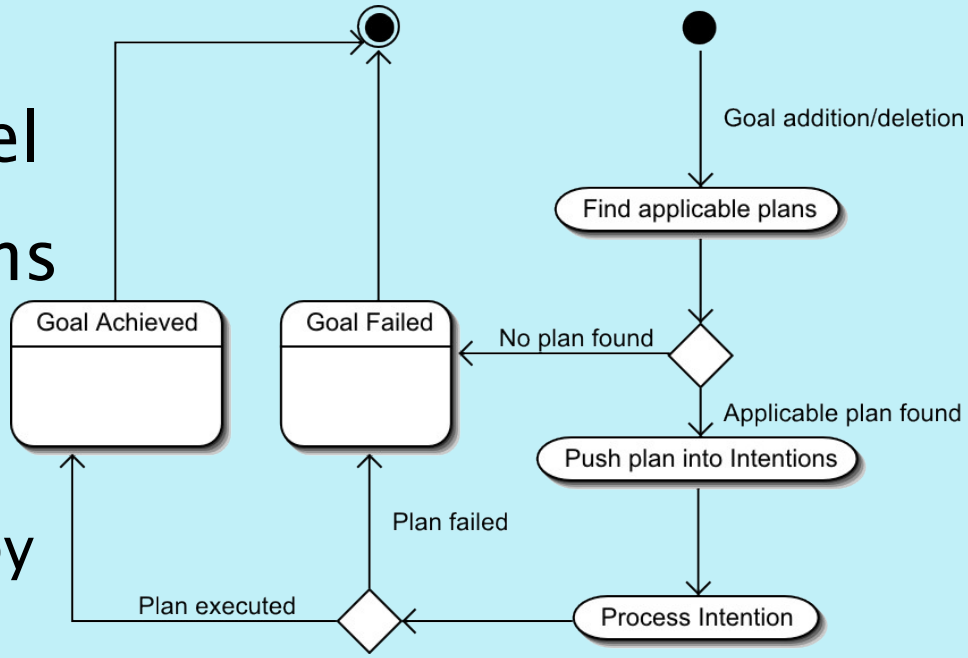
- Procedural versus Declarative
- AgentSpeak(L)
- AgentSpeak-PL
  - The planning action
  - Chaining plans
  - AgentSpeak <---> STRIPS
  - Plan Execution
- Experiments

Procedural versus Declarative

- Procedural agents select black box plans to handle events in the environment
- Declarative agents select capabilities to bring about a specific world state

AgentSpeak(L)

- Procedural agent language
- Based on the BDI model
- Designer specifies plans in a library
  - Plans encode procedures
  - Plans are characterised by trigger and context condition
  - Goals are implicit in the plans

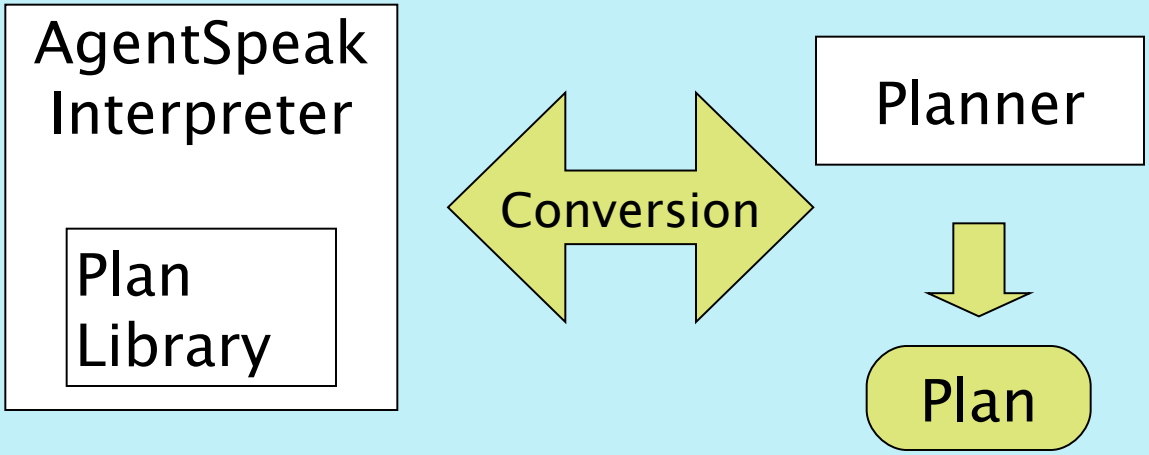


AgentSpeak(L)

- Actual goal is implicit
  - +!move(Destination) : true
  - ← !buy(iceCream).
- We want to be able to say
- !achieve(at(Dest))
- But this would require a very different interpreter…
  - Not exactly

AgentSpeak-PL

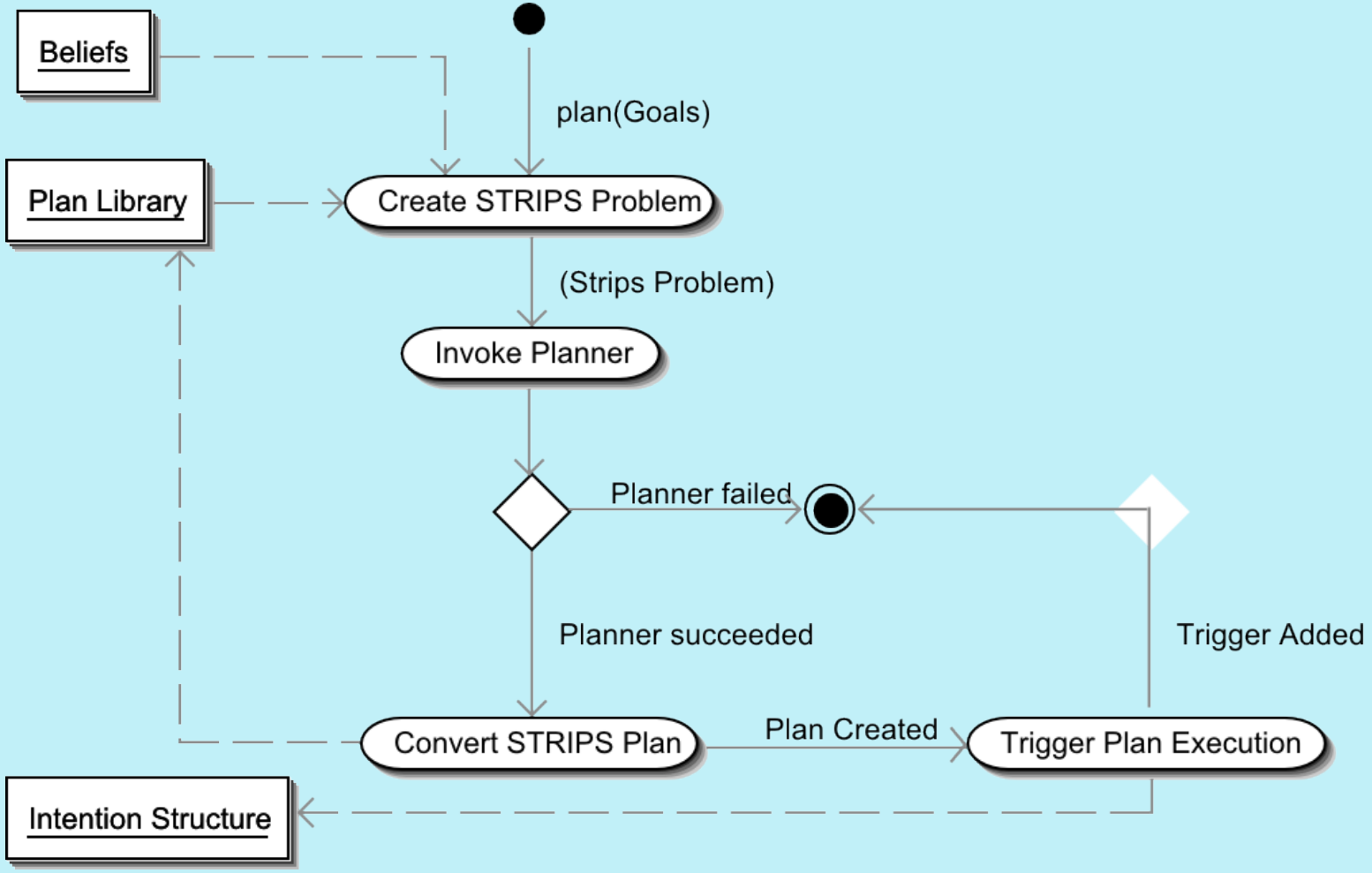
- AgentSpeak(L) + Planning
  - Standard AgentSpeak(L) language
  - Planner invoked through an atomic action
- In principle, any state-space planner can be used



The planning action

- Point of connection between interpreter and planner
- Encapsulates a conversion process
  - From AgentSpeak to STRIPS/PDDL
  - From a STRIPS plan to AgentSpeak
- Takes a conjunction of beliefs as input
- Succeeds if planner is able to generate a plan to achieve its input

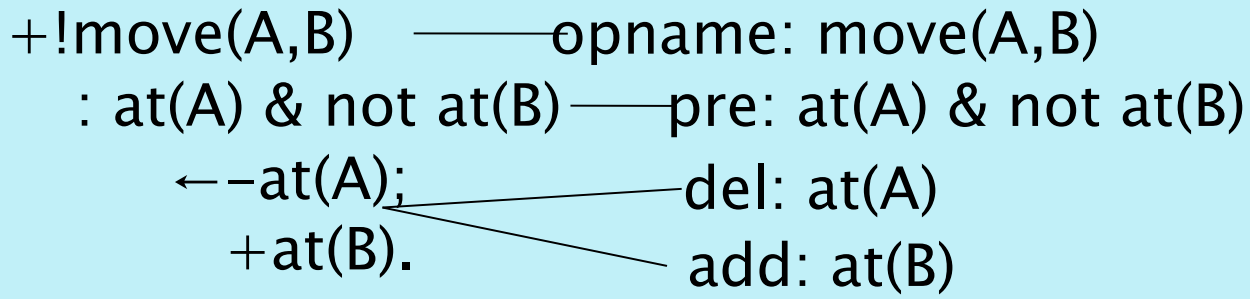
The planning action



Chaining Plans

- Traditional AgentSpeak programming is similar to procedural languages
  - Lower-level basic procedures (subroutines)
  - Higher-level procedures that use them
- AgentSpeak-PL uses a planner to create new high-level plans by chaining basic plans

AgentSpeak to STRIPS



- Relies on clear similarities between AgentSpeak plans and STRIPS operators
- Desired world state becomes the planner's goal
- Belief base becomes the planner's start state

STRIPS to AgentSpeak

STRIPS Plan
to achieve
battery(full):

```
move(1,1)
move(1,2)
charge
```

AgentSpeak(L):

```
+!goal_conj(battery(full))
   : true
 <- !move(1,1);
    !move(1,2);
    !charge.
```

- Each STRIPS action correspond to a low-level AgentSpeak(L) plan
- Plans amount to a series of AgentSpeak(L) subgoals

Plan Execution

- Generated plans are added to the plan library
- The event that caused planning is re-posted
- Generated plan is executed as a regular AgentSpeak(L) plan

Tests: Cleaning Robot

- Taken from Rao's paper on AgentSpeak
- Cleaning Robot must switch lanes to pick up waste and move it to a bin

Cleaning Robot

```
AgentSpeak(L)
% Plan 1
+location(waste, X)
 : location(robot,X) &
   location(bin,Y)
 <- pick(waste);
    !location(robot,Y);
    drop(waste).
% Plan 2
+!location(robot, X)
 : location(robot,X)
 <- true.
% Plan 3
+!location(robot, X)
 : location(robot,Y) &
   not X = Y &
   adjacent(Y,Z)&
   not location(car,Z)
 <- move(Y,Z);
    !location(robot,X).
```

```
AgentSpeak-PL
+location(waste,X)
 : location(robot,X) &
   location(bin,Y)
 <- pick(waste);
    +goal_conj([location(robot,Y)]);
    drop(waste).

+!move(X,Y)
 : location(robot,X) &
   not X = Y &
   not location(car,Y) &
   adjacent(X,Y)
 <- -location(robot,X);
    +location(robot,Y);
    move(X,Y).
```

Conclusions

- Planning augments the runtime flexibility of BDI agents
- In the case of AgentSpeak(L), planning allows it to be used with declarative goals
- Previous efforts focused on HTN planning, which is not declarative

Future Work

- Refine conversion process
  - Remove irrelevant beliefs from start state
  - Derive minimum context condition for generated plans
- Investigate issues with plan addition
  - Interactions of new plans and the plan library
  - Positive and negative plan interaction
- Study planning agents in a society