AgentSpeak & Jason

AgentSpeak

- Programming language for building ***rational agents***
- Belief-Desire-Intention paradigm:
  - **Belief**: the agents knowledge of the world
    - Based on its **perception** and/or **inference rules**
  - **Desire**: situation the agent would like to be in
    - Goals are desires the agent is currently going for
  - **Intention**: a desire that the agent is committed to
    - Plan: action sequence for a specific intention
- Jason is an implementation of the AgentSpeak language using Java

BDI Interpreter Loop

- Agent receives **events**. Event can be:
  - External: from environment and/or perception
  - Internal: e.g.: a belief changed, a plan is dropped
- Agent searches for **plans** that match the **event**
- Chooses one plan to handle the event:
  - This becomes the new **intention**
- Executes the **actions** specified by the plan
  - Executing the plan, new events may be generated

Belief

- First-order logic atoms
- Example
  - father(darth_vader, luke);father(darth_vader, leia)
  - sibling(luke,leia)
- Strong negation operator (~) is available:
  - ~father(uncle_ben, luke) :
    - I know that Uncle Ben is NOT Luke's father
- Closed world assumption (not operator):
  - not father(uncle_ben, luke)
    - I don't know that Uncle Ben is Luke's father nor can I derive it from the known facts

Rule

Syntax: HEAD :- BODY.

Body can be complex using & (and) and | (or) operators

Example:

sibling(A,B) :- sibling(B,A).

sibling(A,B) :- father(C,A) & father(C,B).

Note: Names with capital letters mean variables. All variables have to be ground in an expression to succeed.

Annotation

- Believes can be annotated to distinguish them in some way
- Syntax: belief[annotation]

Example:

- father(darth_vader,luke)[source(darth_vader)]
- can_use(luke,force)[source(self)]

Typical use:

- Source of belief is always annotated.

Goals

- Achievement goal:
  - A goal that we want to achieve
  - Syntax: **!destroyed(death_star)**
- Test goal:
  - A goal that we want to know
  - Syntax: **?father(darth_vader, luke)**
  - Used for querying during plan execution. Example:
    - ?sibling(Target, X)
    - If Target is ground, X will be equal to one of Targets siblings

Events

Change of believes or goals:

-Belief : belief is dropped

+Belief: belief is added

-!Goal: achievement goal is dropped

+!Goal: achievement goal is added

-?Goal: test goal is dropped

+?Goal: test goal is added

Plan

- The program of an agent is a set of plans
- Syntax: **event : context <- action1; action2; action3.**
- Event:
  - A plan should be designed to handle an event
  - This event triggers the plan
- Context:
  - The circumstance under which the plan can be executed
- Body:
  - Sequence of actions that will be executed

Plan

Example1:

```
+arrived(Object, yavin_4) :
    Object == death_star & is_at(luke, yavin_4)
    <-  !befriend(luke, ewoks);
        !initiate_attack
        +under_attack(Object).
```

Means: This plan is triggered when a new belief is added that an Object arrived at Yavin 4. This plan will be executed if the Object is the Death Star, and Luke is at Yavin 4. The plan is: Luke befriends the Ewoks, we initiate the attack and we will believe that the Object is under attack.

Internal actions

Always starts with a dot:

- .stopMAS(): stops the system
- .time(H,M,S): query the time
- .wait(X): pause for X milliseconds
- .random(X): query a random number 0 <= X <= 1
- .print(…): prints to console

Communication

.send(Reciever, Type, Message)

Types:

- tell/untell: adds/removes a belief
- achieve/unachieve: adds/removes an achievement goal
- askOne, askIf: test goals
  - .send(obi_wan, askOne, ?father(luke), Answer)
  - .send(obi_wan, askIf, ?father(luke))