Multi-Agent Programming

Brian Logan

School of Computer Science
University of Nottingham

Midlands Graduate School
8th – 12th April 2013

Exercise: Programming a BDI vacuum cleaner

Outline

- SimpleAPL: a simplified agent programming language
- SimpleAPL syntax
- exercise: writing a simple SimpleAPL program ...

SimpleAPL

3APL/2APL

3APL (Dastani, van Riemsdijk, Dignum and Meyer: "*A programming language for cognitive agents: Goal directed 3APL*" Proc. ProMAS 2003, LNCS 3067 (2004))
`http://www.cs.uu.nl/3apl`

2APL (Dastani: "2APL: A practical agent programming language" *JAAMAS* 16(3) (2008))
`http://www.cs.uu.nl/2apl`

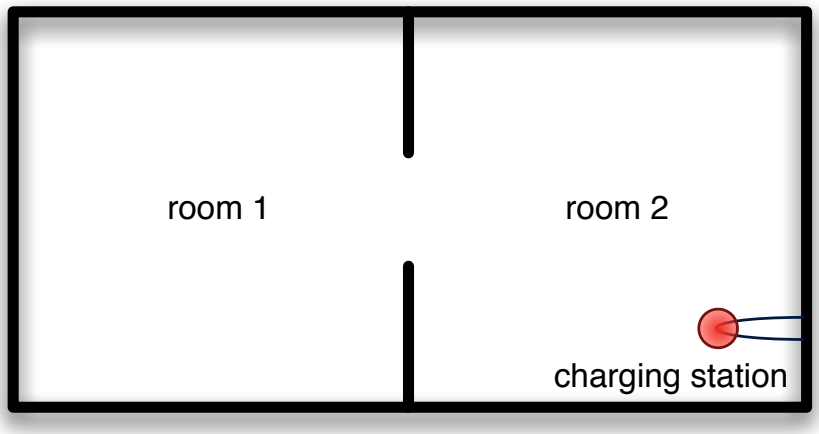
SimpleAPL is a fragment of Goal-directed 3APL and 2APL—(Alechina, Dastani, Logan and Meyer: "*A Logic of Agent Programs*", Proc. AAAI 2007 (2007))

SimpleAPL

- SimpleAPL allows the implementation of agents with beliefs, goals, actions, plans, and planning rules
- main features of 3APL/2APL not present in SimpleAPL are
  - a first order language for beliefs and goals
  - some actions, e.g., abstract plans, communication actions
  - some rule types, e.g., rules for revising plans and goals and for processing events

Example: robot vacuum cleaner

- based on a simple agent described in Russell & Norvig (2003)
- agent has to clean two rooms: *room1* and *room2*
- agent has sensors that tell it if a room is clean and whether its battery is charged
- vacuuming a room results in the room being clean and discharges the agent's battery
- agent can recharge its battery at a recharging station in *room2*



SimpleAPL beliefs

- the beliefs of a SimpleAPL agent represent its information about its environment and itself
- beliefs are represented by a set of positive literals
- the initial beliefs of an agent are specified by its program
- e.g., the agent may initially believe that it's in *room1* and its battery is charged:

```
Beliefs:
  room1, battery
```

SimpleAPL goals

- the agent's goals represent situations the agent wants to realise (not necessarily all at once)
- goals are represented by a set of arbitrary literals
- the initial goals of an agent are specified by its program
- e.g., the agent may initially want to achieve a situation in which both *room1* and *room2* are clean

```
Goals:
  clean1, clean2
```

Declarative goals

- the beliefs and goals of an agent are related to each other
  - if an agent believes p, then it will not pursue p as a goal
  - if an agent does not believe that p, it will not have $\neg p$ as a goal
- these relationships are enforced by the agent architecture

SimpleAPL basic actions

- *basic actions* specify the capabilities of the agent (what it can do independent of any particular agent program)
- 3 types of basic actions:
  - **belief test actions**: test whether the agent has a given belief
  - **goal test actions:** test whether the agent has a given goal
  - **belief update actions:** "external" actions which change the agent's beliefs

Belief and goal test actions

- a *belief test action* ϕ? tests whether a boolean belief expression ϕ is entailed by the agent's beliefs, e.g.:

  ```
  (room2 and -battery)?
  ```

  tests whether the agent believes it is in *room2* and its battery is not charged

- a *goal test action* ψ! tests whether a disjunction of goals ψ is entailed by the agent's goals, e.g.:

  ```
  clean2!
  ```

  tests if the agent has a goal to clean *room2*

Belief update actions

- *belief update actions* change the beliefs (and goals) of the agent
- a belief update action is specified in terms of its pre- and postconditions (sets of literals), e.g.:

```
{room1, battery} suck {clean1, -battery}
```

- an action can be executed if one of its pre-conditions is entailed by the agent's current beliefs
- executing the action updates the agent's beliefs to make the corresponding postcondition entailed by the agent's beliefs

Belief and goal entailment

- a belief query (a belief test action or an action precondition) is entailed by the agent's belief base if
  - all positive literals in the query are contained in the agent's belief base, and
  - for every negative literal $\neg p$ in the query, p is not in the belief base
  - i.e., we use entailment under the closed world assumption
- goal entailment corresponds to a formula being classically entailed by *one* of the goals in the goal base

Belief and goal update

- executing a belief update action
  - adds all positive literals in the corresponding postcondition to the belief base, and
  - for every negative literal $\neg p$ in the postcondition, p is removed from the agent's belief base
- goals which are achieved by the postcondition of an action are dropped
- for simplicity, we assume that the agent's beliefs about its environment are always correct and its actions in the environment are always successful

SimpleAPL plans

- *plans* are sequences of basic actions composed by plan composition operators:
  - **sequence:** "$\pi_1 ; \pi_2$" (do π_1 then π_2)
  - **conditional choice:** "`if` ϕ `then` $\{\pi_1\}$ `else` $\{\pi_2\}$"
  - **conditional iteration:** "`while` ϕ `do` $\{\pi\}$"
- e.g., the plan:

```
if room1 then {suck} else {moveL; suck}
```

causes the agent to clean *room1* if it's currently in *room1*, otherwise it first moves (left) to *room1* and then cleans it

SimpleAPL rules

- *planning goal rules* are used for plan selection based on the agent's current goals and beliefs
- a planning goal rule $\kappa \leftarrow \beta \mid \pi$ consists of three parts:
  - κ: an (optional) *goal query* which specifies which goal(s) the plan achieves
  - β: a *belief query* which characterises the situation(s) in which it could be a good idea to execute the plan
  - π: a plan
- a rule can be applied if κ is entailed by the agent's goals and β is entailed by the agent's beliefs
- applying the rule adds π to the agent's plans

Example SimpleAPL PG rules

- e.g., the PG rule

```
clean1 <- battery | if room1 then {suck}
                    else {moveL; suck}
```

states that "if the agent's goal is to clean *room1* and its battery is charged, then the specified plan may be used to clean the room"

SimpleAPL BNF 1

⟨*APL_Prog*⟩ ::= "BeliefUpdates:" ⟨*updatespecs*⟩
| "Beliefs:" ⟨*pliterals*⟩
| "Goals": ⟨*literals*⟩
| "PG rules:" ⟨*pgrules*⟩
⟨*updatespecs*⟩ ::= [⟨*updatespec*⟩ ("," ⟨*updatespec*⟩)*]
⟨*updatespec*⟩ ::= "{" ⟨*literals*⟩ "}" ⟨*aliteral*⟩ "{"⟨*literals*⟩"}"
⟨*pliterals*⟩ ::= [⟨*pliteral*⟩ ("," ⟨*pliteral*⟩)*]
⟨*literals*⟩ ::= [⟨*literal*⟩ ("," ⟨*literal*⟩)*]
⟨*plan*⟩ ::= ⟨*baction*⟩ | ⟨*seqplan*⟩ | ⟨*ifplan*⟩ | ⟨*whileplan*⟩

where ⟨*literal*⟩ (⟨*pliteral*⟩) denotes belief and goal literals (positive literals).

SimpleAPL BNF 2

⟨*baction*⟩ ::= ⟨*aliteral*⟩ | ⟨*testbelief*⟩ | ⟨*testgoal*⟩
⟨*testbelief*⟩ ::= ⟨*bquery*⟩ "?"
⟨*testgoal*⟩ ::= ⟨*gquery*⟩ "!"
⟨*bquery*⟩ ::= ⟨*literal*⟩ | ⟨*bquery*⟩ "and" ⟨*bquery*⟩ |
⟨*bquery*⟩ "or" ⟨*bquery*⟩
⟨*gquery*⟩ ::= ⟨*literal*⟩ | ⟨*gquery*⟩ "or" ⟨*gquery*⟩
⟨*seqplan*⟩ ::= ⟨*plan*⟩ ";" ⟨*plan*⟩
⟨*ifplan*⟩ ::= "if" ⟨*bquery*⟩ "then {" ⟨*plan*⟩ "}" ["else {" ⟨*plan*⟩ "}"]
⟨*whileplan*⟩ ::= "while" ⟨*bquery*⟩ "do {" ⟨*plan*⟩ "}"
⟨*pgrules*⟩ ::= [⟨*pgrule*⟩ ("," ⟨*pgrule*⟩)*]
⟨*pgrule*⟩ ::= [⟨*gquery*⟩] "<-" ⟨*bquery*⟩ "|" ⟨*plan*⟩

where ⟨*aliteral*⟩ denotes the name of a belief update action

Exercise: write a SimpleAPL program to clean both rooms

Hints

- write a set of action specifications (pre- and postconditions) for belief update actions actions, e.g., `moveR`, `moveL`, `suck`, `charge`
- write a set of plans that will achieve the goals `clean1`, `clean2`
- write PG rules that select appropriate plans given the agent's goals