Prolog Agents in JAM
Laboratory of Multiagent Systems LM
Laboratorio di Sistemi Multiagente LM

Elena Nardini
elena.nardini@unibo.it
Ingegneria Due
Alma Mater Studiorum—Università di Bologna a Cesena

Academic Year 2010/2011
1 2P in JAM

2 Integration of TuProlog in JAM

3 Exercises

4 Bibliography
JAM Middleware

Features
- Supports agent creation, management and interaction by exploiting the Java technology—an object-oriented technology
- Provides the Runtime Agent Middleware (RAM)
  - Based on Java RMI and the Agent Directory Service Layer (ADSL)
  - Supports agent hosting, finding, execution, and message exchange among agents

Drawbacks
- Java is not an agent language
- JAM does not provide MAS abstractions in an explicit way
Prolog

Features

- Is a declarative, logic programming language
- Is capable of inference
- Is goal oriented
  - Suitable to do symbolic elaboration
  - From AI: The intelligence is based on the ability to automatically manipulate symbols
  - Simulates intelligence

Drawbacks

- Is not an agent language
- Prolog goal is not a general agent goal
- As logic language is not able to model in terms of program structures
tuProlog (2P)

A hybrid solution
- Java to support modelling in terms of program structures
- Prolog to support symbolic computation

tuProlog
- Is a Java-based Prolog
- Realised by APICe group: http://tuprolog.alice.unibo.it/
  → Allows Prolog and Java to be exploited in an integrated way
Towards 2P in JAM

Towards a “strong” notion of agent

- In a strong view, an agent is provided with a sort of intelligence
- From AI: The intelligence is based on the ability to automatically manipulate symbols
  - Declarative and logic programming languages like Prolog are more suitable to model agents than imperative languages like Java

2P agents in JAM

- The knowledge representation and the agent behaviours can be modelled in Prolog
- The integration between Prolog agents and RAM can be done through Java
A Prolog agent is modelled through an object, instance of the JAMPrologAgent class, extending the JAMAgent class.

It is added:

- A Prolog engine
- A Prolog theory containing the knowledge base associated to the agent and the set of behaviours

```
JAMPrologAgent
- prologEngine : Prolog
- theoryText : String
- theoryInputStream : InputStream
+ registerObject()
+ start()
```
Integration between Prolog and Java

- The method `registerObject` allows to link a Prolog theory to Java objects

Example

- The following instruction it is used in the JAMPrologAgent constructor:
  ```java
  registerObject(new Struct("agent"), this);
  ```

- It allows to use the following Prolog instruction:
  ```prolog
  agent ← readOldTemperature returns X
  ```

- That executes the method `readOldTemperature` of the current `JAMPrologAgent` object and unifies the method results with the variable `X`
New tasks for the method start

- Sets the Prolog theory of an agent to the related Prolog engine
- Provides each behaviour with the Prolog engine
A Prolog agent behaviour is modelled through an object, instance of the JAMPrologBehaviour class, extending the JAMBehaviour class.

It is added:

- The Prolog engine of the agent
- Prolog is goal-oriented → The (Prolog and agent) goal associated to the specific behaviour

The method action

Corresponds to solve the Prolog goal associated to the particular agent behaviour

```
JAMPrologBehaviour
-prologEngine : Prolog
-goalText : String
+setPrologEngine()
+run()
+action()
```
Exercise 1

- Download the new library JAM.jar also containing the package JAM.TuProlog
  - JAM.TuProlog contains the classes JAMPrologAgent and JAMPrologBehaviour
- Download Exercise1.zip
  - Provides an example of library JAM.TuProlog use
- Study the simple agent application and run it through the main provided by the class ThermostatAgent

Thermostat Agent task

- Check the environment temperature $T$
- Until $T$ is not: $> 18$ and $< 22$:
  - Decrease $T$ of one unit if the temperature is $\geq 22$
  - Increase $T$ of one unit if the temperature is $\leq 18$
Thermostat Agent by Exploiting JAM.TuProlog

What it is needed to model?

- Thermostat Agent
- Environment
- Interaction between the agent and the environment
Class **ThermostatAgent** extending the class **JAMPrologAgent**
  - Only redefines the two constructors of the super-class

Class **ThermostatAgentBehaviour** extending the class **JAMPrologBehaviour**
  - Redefines the constructor of the super-class
  - Implements the methods `setup` and `dispose`
In JAM there is not an explicit abstraction of environment

In JAM does not support interactions with the physical environment

→ It is needed to simulate the physical environment

→ The environment is simulated through a Prolog engine and a theory

State of the environment as a Prolog term

temp(25).
Interaction Between the Agent and the Environment

Agent provided with a sensor and an actuator
[Wooldridge and Jennings, 1995]

- **Class TemperatureSensor**
  - Has a model of the environment
  - Method `getTemperature` sensing the environment temperature

- **Class TemperatureActuator** extending the class `JAMPrologBehaviour`
  - Has a model of the environment
  - Method `setTemperature` setting the environment temperature

- **Method registerObject** to register the sensor and the actuator in `ThermostatAgent`
Agent Behaviour in a Prolog Theory

start :- check_temperature.

check_temperature :-
    sensor <- getTemperature returns X,
    X >= 22, !,
    T is X - 1,
    change_temperature(T).

check_temperature :-
    sensor <- getTemperature returns X,
    X =< 18, !,
    T is X + 1,
    change_temperature(T).

check_temperature :-
    sensor <- getTemperature returns X,
    X > 18, X < 22.

change_temperature(X) :-
    actuator <- setTemperature(X),
    write(X),
    check_temperature.

→ The agent behaviour goal: start.
Two interacting agents

- **Thermostat Agent:**
  - Senses the environment temperature and sends the value to Temperature Manager Agent
  - Sets the environment temperature with the new value given by Temperature Manager Agent

- **Temperature Manager Agent:**
  - Reads the temperature provided by ThermostatAgent
  - Calculates the new temperature and sends the new value to ThermostatAgent
Exercise 2 II

Two new Prolog theory
- `tempManager` related to Temperature Manager Agent
- `tempSensAct` related to Thermostat Agent

Agent Interaction
- Exploit the RAM provided by JAM and the method `registerObject`
Open Points, Again!

Questions

- Which differences with an implementation exploiting the object paradigm?
- Environment not explicitly modelled from the middleware: which are the drawbacks?
- Direct communication among agents: which are the drawbacks?

Bibliography

Prolog Agents in JAM
Laboratory of Multiagent Systems LM
Laboratorio di Sistemi Multiagente LM

Elena Nardini
elena.nardini@unibo.it

Ingegneria Due
Alma Mater Studiorum—Università di Bologna a Cesena

Academic Year 2010/2011