Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Agentes Inteligentes
2020/2021

Filipe Gonçalves, Paulo Novais, César Analide

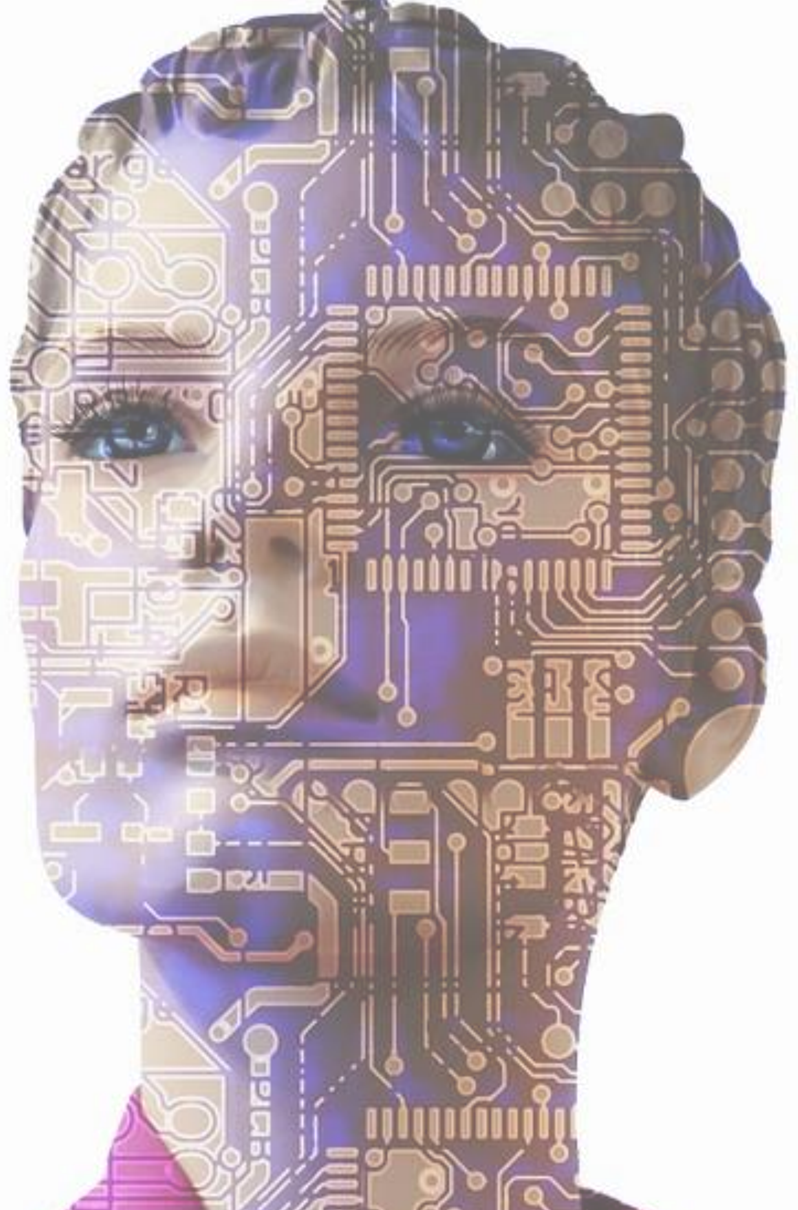
ISLab
Synthetic Intelligence Lab

Agentes Inteligentes @ 2020/2021

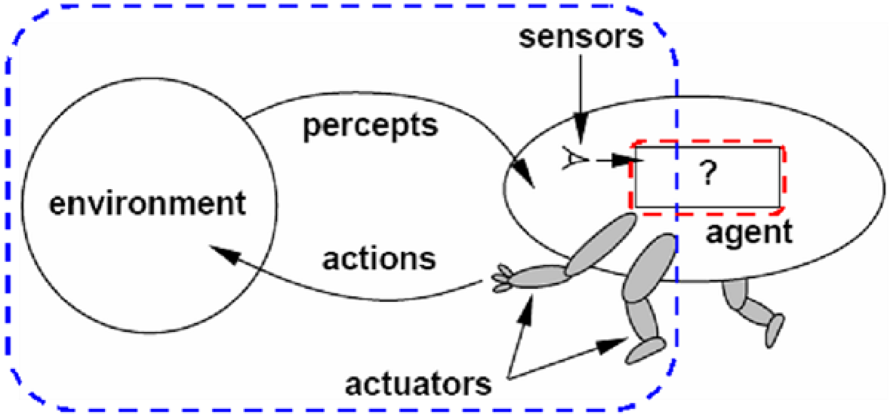
- Paulo Novais – pjon@di.uminho.pt
- César Analide – analide@di.uminho.pt
- Filipe Gonçalves – fgoncalves@algoritmi.uminho.pt

- Departamento de Informática
  Escola de Engenharia
  Universidade do Minho
- ISLab – (Synthetic Intelligence Lab)
- Centro ALGORITMI
  Universidade do Minho

JESS IN JADE



JESS used to implement the reasoning module of a JADE agent



JADE provides environment and facilitates the sending / receiving of messages

JESS enables the implementation of the agent's decision module in a declarative way

- JESS can be used in one of the many behaviours of an agent

Considerations:

- To embed Jess in a Java application (such as a JADE agent), you simply need to create a jess.Rete object

- To run the rule-based engine, the method Rete.run() is applied
    - Makes the engine consecutively fire applicable rules, and will return only when there are no more rules to fire
    - The JADE agent thread will be blocked while Rete engine is running!

Mitigation:

- Jess.Rete class includes another run method that allows us to specify the maximum number of cycles the engine should run.

JESS-JADE API - JessBehaviour

JessBehaviour

- Agent able to continuously reason by implementing a CyclicBehaviour consisting of running a Jess engine
- JessBehaviour starts by loading prepared CLP script
- Behaviour action executes limited number of passes
    - jess.run(x) returns number of passes executed
- If no passes are executed, block behaviour until triggered

```
class JessBehaviour extends CyclicBehaviour {
    // the Jess engine
    private jess.Rete jess;
    // maximum number of passes that a run of Jess can execute before giving control to the agent
    private static final int MAX_JESS_PASSES = 1;

    JessBehaviour(Agent agent, String jessFile) {
        super(agent);
        // create a Jess engine
        jess = new jess.Rete();
        // load the Jess file
        try {
            // open the Jess file
            FileReader fr = new FileReader(jessFile);
            // create a parser for the file
            jess.Jesp j = new jess.Jesp(fr, jess);
            // parse the input file into the engine
            try {
                j.parse(false);
            } catch (jess.JessException je) {
                je.printStackTrace();
            }
            fr.close();
        } catch (IOException ioe) {
            System.err.println("Error loading Jess file - engine is empty");
        }
    }

    public void action() {
        // to count the number of Jess passes
        int executedPasses = -1;
        // run jess
        try {
            // run a maximum number of steps
            executedPasses = jess.run(MAX_JESS_PASSES);
        } catch (JessException je) {
            je.printStackTrace();
        }
        // if the engine stopped, block this behaviour
        if(executedPasses < MAX_JESS_PASSES)
            block();
            // the behaviour shall be unblocked by a call to restart()
    }

    ...
} // end JessBehaviour class
```

ACLMessage & Facts

- In JADE, it is a good practice to implement methods to handle ACLMessage as a new String Fact

- **Required to create respective methods:**
    - **addFact(String fact)** - assert new information into the JESS engine (also unblocks JESSBehaviour)
    - **newMsg(ACLMessage msg)** - handle ACLMessage to add as a new Fact

```
boolean addFact(String jessFact) {
    // assert the fact into the Jess engine
    try {
        jess.assertString(jessFact);
    } catch(JessException je) {
        return false;
    }
    // if blocked, wake up!
    if(!isRunnable()) restart();
    // message asserted
    return true;
}
```

```
boolean newMsg(ACLMessage msg) {
    String jf = ... // use msg to assemble a Jess construct
    // "feed" Jess engine
    return addFact(jf);
}
```

ACLMessage & Facts

- In JADE, it is a good practice to implement methods to handle JESS Facts

- **Required to create respective methods:**
    - **runQuery(String queryName, ValueVector values) -** query facts from JESS engine
    - **removeFacts(String name)**

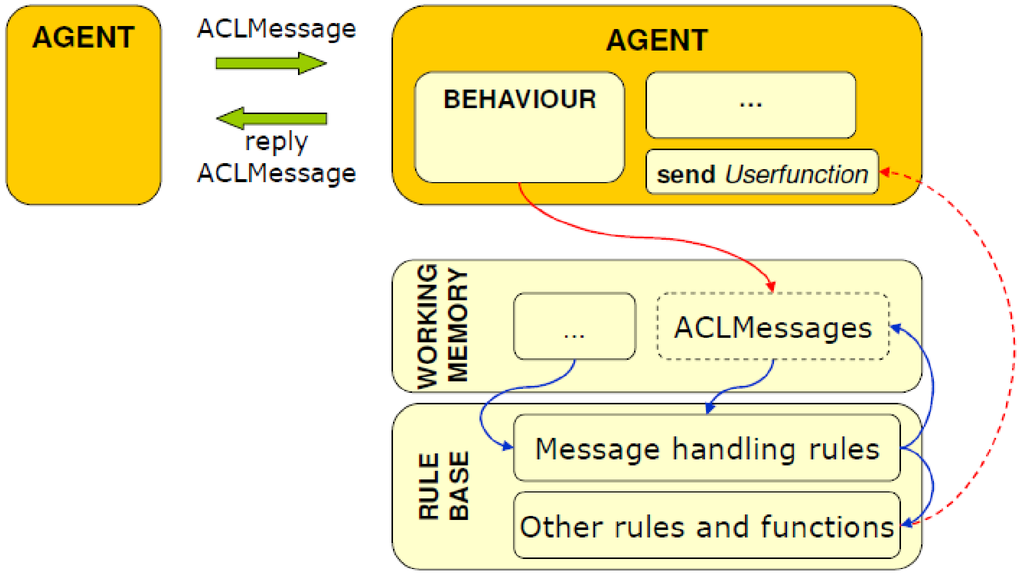
```java
public Iterator runQuery(String queryName, ValueVector values){
    Iterator it = null;
    try{
        it = jess.runQuery(queryName, values);
    }
    catch(JessException je){
        je.printStackTrace();
    }
    return it;
}
```

```java
public void removeFacts(String name)  {
    try  {
        jess.removeFacts(name);
    } catch  (JessException e)  {
        e.printStackTrace();
    }
}
```

MsgListening & JessBehaviour

- Based on the previous methods, we now integrate a Behaviour that handles incoming messages (which makes use of JessBehaviour)

- All messages received are directly provided into JessBehaviour, which in turn handles the rule-based system



```
class MsgListening extends CyclicBehaviour {
    // a reference to the JessBehaviour instance
    private JessBehaviour jessBeh;

    MsgListening(Agent agent, JessBehaviour jessBeh) {
        super(agent);
        // save reference to the JessBehaviour instance
        this.jessBeh = jessBeh;
    }

    public void action() {
        MessageTemplate mt = ... // some template
        ACLMessage msg = myAgent.receive(mt);
        if (msg != null) {
            // put into Jess engine
            if(jessBeh.newMsg(msg))
                ... // do something
            else
                ... // do something else
        } else
            block();
    }

} // end MsgListening class
```

- Guidelines:
  - Introduction to Programming with Jess in Java
  - Embedding Jess in a Java Application
  - Adding Commands to Jess

- Documentation:
  - Class Jess.Rete

- More examples available at JADE API
  - JADE API Examples download link: https://jade.tilab.com/dl.php?file=JADE-examples-4.5.0.zip
  - Directory: JADE-examples-4.5.0.zip\jade\src\examples\jess\*

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Agentes Inteligentes
2020/2021

Filipe Gonçalves, Paulo Novais, César Analide