Design and development of a visual environment for writing DyLOG programs

C. Schifanella, L. Lusso, M. Baldoni, C. Baroglio
Dipartimento di Informatica, Univ. degli Studi di Torino
Corso Svizzera 185, 10149 Turin (Italy)
{schi, baldoni, baroglio}@di.unito.it,
lussoluca@tiscali.it

Turin, Italy, 12-01-2004

Outline

- Introduction
- The DyLOG language
- A DyLOG implementation
- The visual environment
- A DyLOG ontology
- Conclusions
- Demo

Agent Oriented System Engineering

- Needing of tools to support
  - design
  - implementation
  - deploy
- Quality of tools influences the choice of the specification language

AOSE: some example environments

- AgentTool
  - Java-based graphical development environment
  - supports analysis, design and implementation
  - used to graphically define high-level system behaviours
- Zeus
  - by British Telecommunication
  - allows to specify and to implement collaborative agents
- DCaseLP
  - allows to integrate different specification and implementation languages in order to model and prototype MASs
- ...

The DyLOG language

- It is a logic language for reasoning about actions and change
- It allows the specification of individual, communicating agents, situated in a multiagent context
- It performs hypothetical reasoning about the effects of conversations on the agents mental state

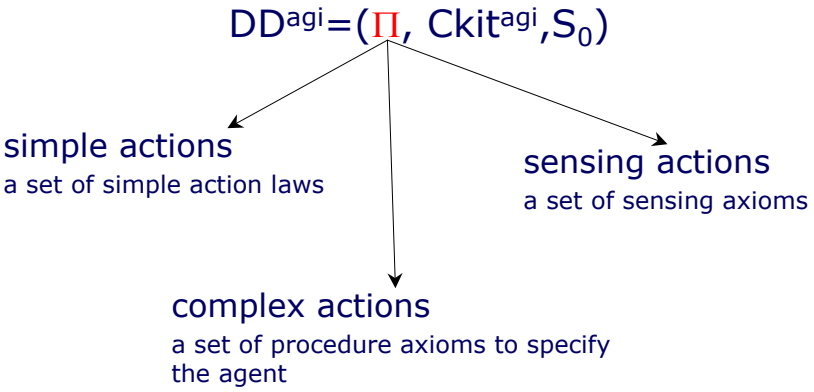
DyLOG + CKit

$$DD^{agi} = (\Pi, Ckit^{agi}, S_0)$$

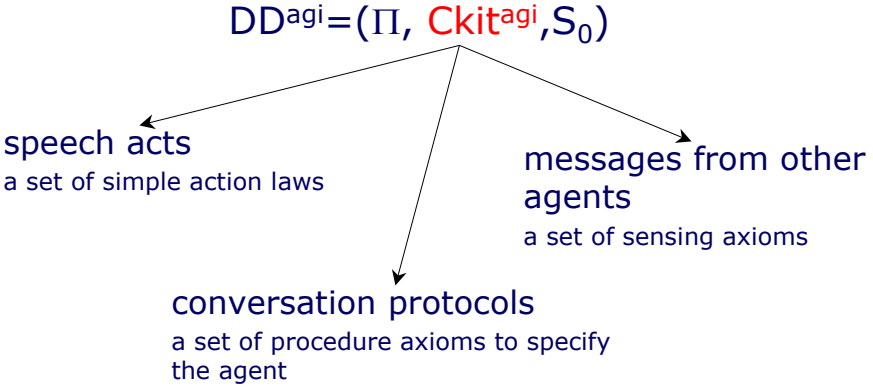
Domain description

it is used to describe the agent's behaviour by means of simple and complex actions, communicative actions and initial observations

DyLOG + CKit

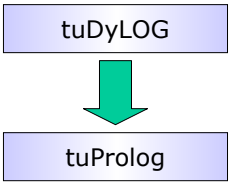


DyLOG + CKit



tuDyLOG: a DyLOG implementation

- Java package
- Built upon tuProlog
  - light-weight Prolog engine
  - it is possible to exploit some common mechanisms (like unification)

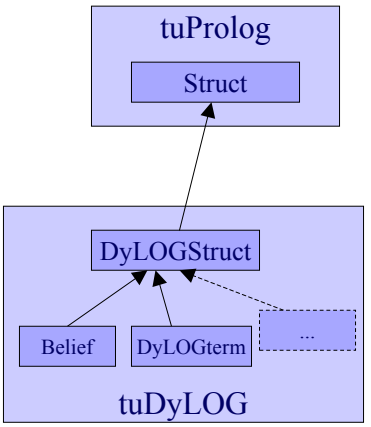


tuDyLOG main elements

- Each costruct is implemented by a class or a set of classes
- Class DomainDescription
  - its instance represents a DyLOG program
  - it contains instances of main program components, such as actions, communicative acts and initial observations

tuDyLOG main elements

- Class DyLOGStruct
  - is an extention of the tuProlog Struct class
  - is the connecting point between tuDyLOG and tuProlog
  - through DyLOGStruct a DyLOG costruct can be turned into a corresponding tuProlog structure



tuDyLOG main elements

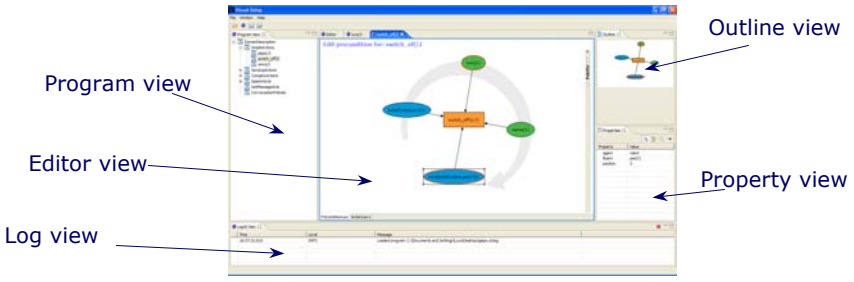
- Class Action
  - abstract class that is extended to represent costructs such as Simple, Complex and Sensing Actions (and Ckit elements).
- Class DyLOGLaw
  - abstract class that is extended to represent Precondition Law, Simple Action Law, etc

tuDyLOG: an example

```
DomainDescription dd = DomainDescription.getInstance();
SimpleAction turn_dial_1=new SimpleAction("turn_dial",1);
BeliefFluent b1=new BeliefFluent(new DyLOGStruct("ag"),new
   DyLOGStruct("in_front_of(I)"));
.....
//Precondition
turn_dial_1.getPreconditionLaw().getAction().setName("turn_dial");
turn_dial_1.getPreconditionLaw().getAction().setTerm(0,new
   Var("I"));
turn_dial_1.getPreconditionLaw().addPrecondition(b1);
....
//First action law
ActionLaw actionLaw = new ActionLaw(new
   DyLOGStruct(turn_dial_1.getName(), argomenti));
....
....
```

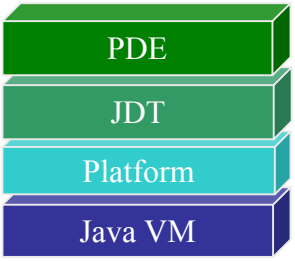

Visual DyLOG

- Visual environment to write DyLOG programs
- Written in Java using Eclipse platform
- Based on model-view-controller paradigm, where tuDyLOG package represents the model



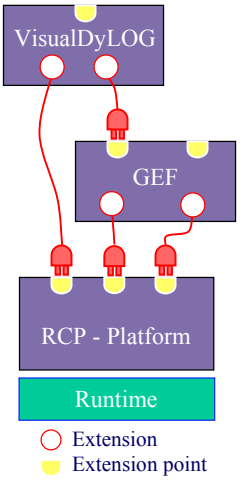
The Eclipse project

- Is a platform designed for building IDEs
- Proven, reliable and scalable technology upon which application can be:
  - designed
  - developed
  - deployed
- Written in Java
- Plug-in architecture



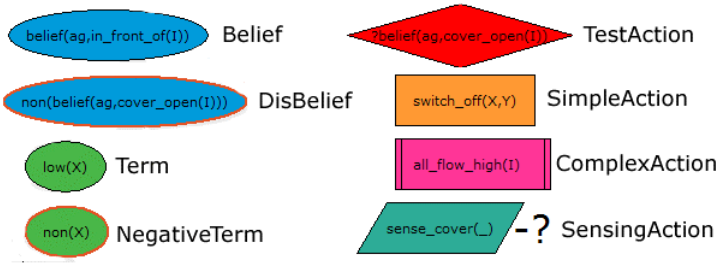
Eclipse Rich Client Platform

- Represents the smallest subset of Eclipse plug-ins that are necessary to build a generic platform application
- Used to deploy and distribute applications as stand-alone tools



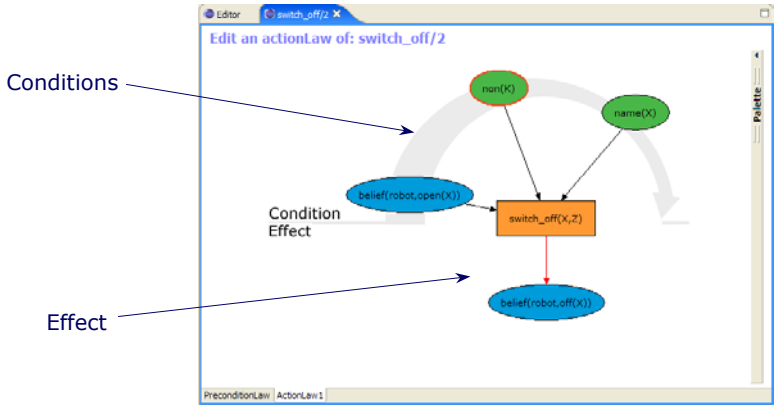
Visual DyLOG

- Aims to make more intuitive the design of DyLOG agent's behaviour by using:
  - graphical notation
  - used shapes recall flow-chart symbols
  - similar constructs are represented by same shapes



Visual DyLOG

- An example of an action law representation:



Visual DyLOG

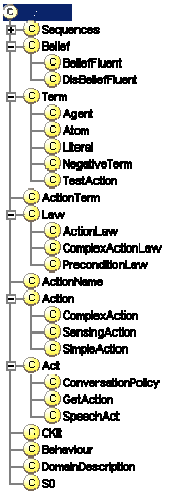
- Architecture is extensible
- It is possible to export/import a DyLOG program in/from many formats like
  - english-like language
  - OWL
  - other formats can be easily added

DyLOG ontology

- Written in OWL
- Allows a better integration of DyLOG agents in a Semantic Web scenario
- Syntactic constraints of the language can be specified by a proper ontology restriction
- An external reasoner can verify the correctness of the programs
- Used as a common interchange format
  - Visual DyLOG provides import/export features

DyLOG ontology main elements

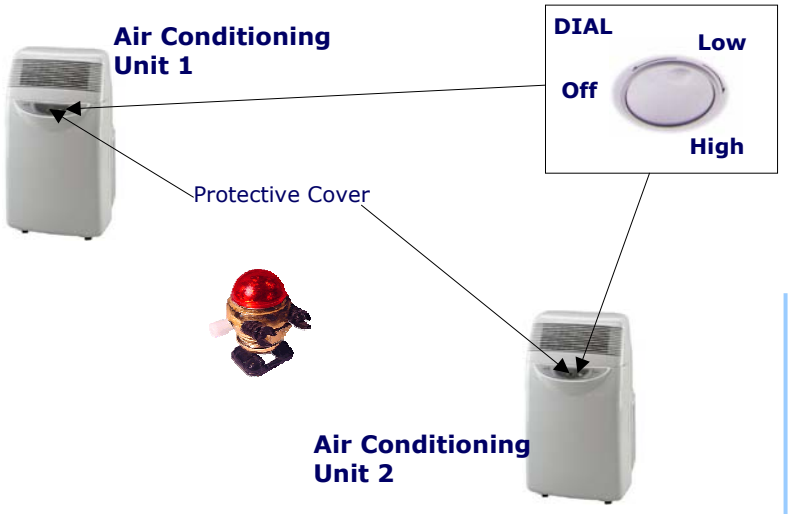
- It exploits the main characteristics of the OWL language
- A DyLOG program is represented by an instance of the DomainDescription class. It has 3 properties that specify
  - behaviour
  - communication policies
  - initial observations
- Each class specifies all the characteristics of the correponding DyLOG costruct



Conclusions and future work

- tuDyLOG allows
  - portability
  - a better integration with applications and frameworks that are already available
- Visual DyLOG allows to reduce the development effort
- Work in progress
  - tuDyLOG intepreter

Demo: an example



Design and development of a visual environment for writing DyLOG programs

C. Schifanella, L. Lusso, M. Baldoni, C. Baroglio
Dipartimento di Informatica, Univ. degli Studi di Torino
Corso Svizzera 185, 10149 Turin (Italy)
{schi, baldoni, baroglio}@di.unito.it,
lussoluca@tiscali.it

Turin, Italy, 12-01-2004