UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Agent Language Analysis: 3APL

Alvarez Napagao Sergio
Auffarth Benjamin
Salazar Ramirez Norman

Course: Multi-Agent Systems
Professor: Javier Vazquez

Introduction

- An Abstract Agent Programming Language or Artificial
- Autonomous Agents Programming Language
- a BDI approach
- implemented in Java at Utrecht

Features

- Different programming levels: base level and deliberation level
- Formal transition system semantics
- Practical reasoning rules for goal and plan revision

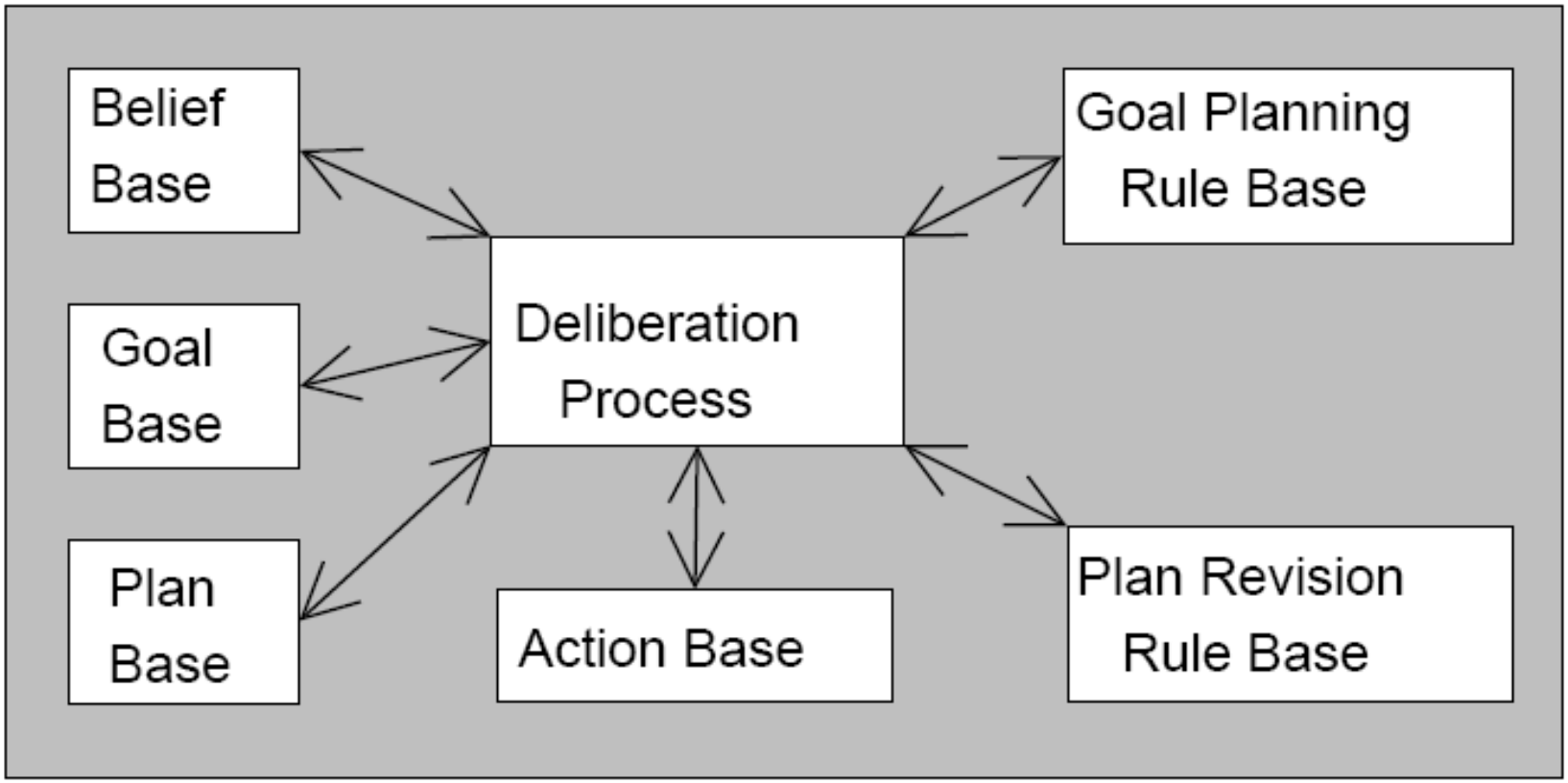
Paradigms in 3-APL

- Imperative programming: recursion, state based computation
- States (BDI): belief, goals, plans
- Logic Programming: querying the belief, goal and plan base

3-APL Agent

- Set of Beliefs (Belief-Base)
- Set of Goals (Goal-Base)
- Set of Plans (Plan-Base)
- Set of Practical Reasoning Rules (Goal Planning Rule-Base and Plan Revision Rule Base)
- Set of Basic Actions (Capabilities)

3-APL Agent


Belief Base
Goal Base
Plan Base
Deliberation Process
Action Base
Goal Planning Rule Base
Plan Revision Rule Base

Beliefs

- Agents has beliefs about its task and environment
- Prolog like formula (i.e. subset of first-order predicate language)
- Example:

  box(room1). pos(room2)

  door(r1,r2) :- door(r2,r1)

Actions

- The most primitive action an agent can perform is a basic action
- Manipulate the agent mental state and the environment
- There are 3 basic actions:
  - Mental Actions
  - Communication Actions
  - External Actions
- A Mental action is formed from a set of pre-execution beliefs and an action and a set of post-execution beliefs

Actions (cont.)

- Mental Action:

  {pos(X,Y)} GoRight() {not pos(X,Y),pos(X+1,Y)}
- Communication Action:

  Send(b,seller,query,computer(3000))
- External Action:

  Java(Genie,Speak("hello"),L)

Goals

- State of affaires desired by the agent
- Goal is a predicate or a conjunction of predicates
- Examples:

  clean()

  transport()

  on(a,b) and on(c,floor)

Plans

- Attitudes corresponding to plans of action to achieve. There are basic, abstract and composite plans
- Basic goals can be of 3 types:
  - A basic action to be executed. (mental, send, external, or skip)
  - A query to the Belief-Base to find if a belief is true or false; this also binds values to the variables in the goal
  - Adopt goal
- A composite plan can be constructed using basic goals
  - Sequence of plans
  - If plan(deterministic choice)
  - While plan (iterative plan)

Practical Reasoning Rules

- Give the agent the capability of constructing and revising plans
- Can also be called means-end reasoning
- Based on a belief that a plan is sufficient to achieve a goal the agent concludes it should adopt the plan

Practical Reasoning Rules (cont.)

- They can be divided in four classes:
  - Reactive
  - Plan
  - Failure (replan)
  - Optimization
- A practical reasoning consists in a:
  - Head which is a goal/plan (*g*)
  - Body which is a plan (*p*)
  - Guard which is a belief (*b*)
- Informally this means that when an agent has a goal *g* and finds it self in situation b it may consider replacing goal *b* with plan *p*

Planning Goal Rules

- Its purpose is to generate a plan that satisfies a specified goal
- It's of the reactive or plan class
- If there is no goal specified it's a reactive rule that would occur every time the guard condition is satisfied
- A PG-Rule is a practical reasoning rule with a goal (it can be empty) in its head
- Example:
- on(X,Z) <- on(X,Y) | move(X,Y,Z)

Plan-Revision Rules

- Its purpose is to adopt, revise or drop plans
- It's of the failure or optimization class
- A PR-Rule is a practical reasoning rule with a plan on its head
- Example:

move(X,Y,Z) <- on(U,Z) | move(U,Z,floor);move(X,Y,Z)

Agent Mental Attitude

- It's a data structure
- It's form from the beliefs, goals, plans and reasoning rules
- It can be modified by deliberation operations:
  - Applying a rule
  - Executing a plan
  - Etc.

Agent Mental Attitude

BDI Theory	3-APL
Beliefs	Beliefs
Desires	Goals(declarative)
Intentions	Plans (procedural)

Deliberation Process

- Its form by the deliberation operations
- It's also interpreter, since it decides the order in which the deliberation operations will be applied
- Can check if a goal still exists during the execution of a plan
- Garbage collection to remove left-over plans
- In case of parallel goals/plans it can decide if both can be implemented at the same time or it could chose between one of them

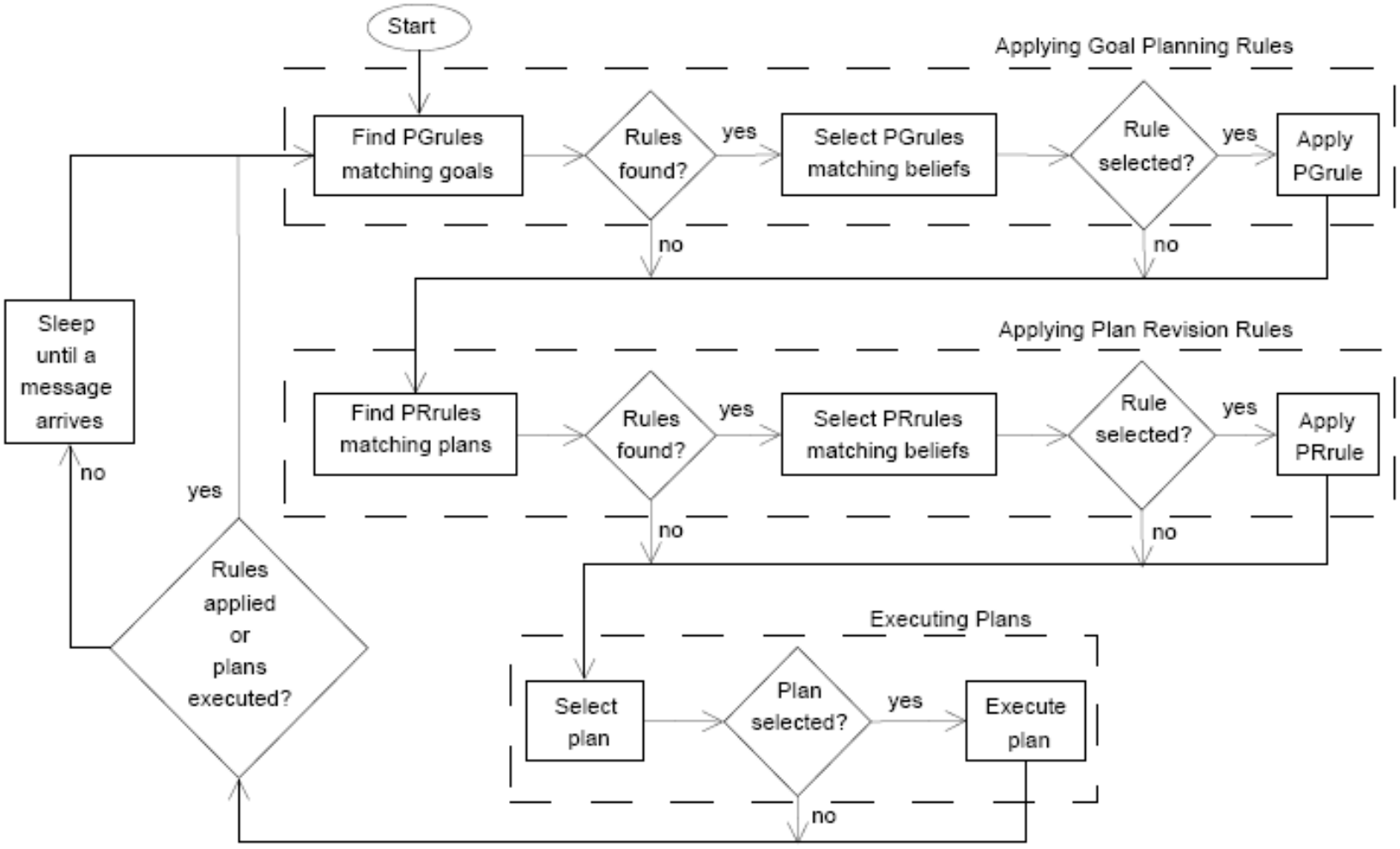
Deliberation Program

- It is meta-level program designed to execute a deliberation process
- It has operations such as:
  - SelectPGRule
  - SelectPlanRevisionRule
  - SelectPlan
  - ExecutePlan
- It can be implemented as a while loop
- Different order of execution of the operations can give the agents different behaviors

Deliberation Cycle

1. Find Plan Generation Rules that Match Goals
2. Remove Plan Generation Rules with atoms in head that exist in Belief Base
3. Find Plan Generation (PG) Rules that Match Beliefs
4. Select a Plan Generation (PG) Rule to Apply
5. Apply the Plan Generation (PG) Rule, thus adding a plan to the planbase
6. Find Plan Revision (PR) Rules that Match Plans
7. Find Plan Revision (PR) Rules that Match Beliefs
8. Select a Plan Revision (PR) Rule to Apply to a Plan
9. Apply the Plan Revision (PR) Rule to the Plan
10. Find Plans To Execute
11. Select a Plan To Execute
12. Execute the (first basic action of the) Plan

Deliberation Cycle (cont.)



Formal Semantics

- 3APL has a formal semantics
- It has been specified in:
  - Plotkin-style transitions
  - Z
- The implementation is the same as the formal semantics

Gaia Design

Gaia	3APL
The Environment Model	Java Class
The Roles Model –Responsibility –Permission	beliefs, goals, plans and actions –goals and plans –beliefs and actions
The Interaction Model	communication and external actions
Organizational structure	all components
–Topology structure	–communication structure, beliefs, goals, plans and actions
–Control structure	–algorithms and reasoning rules

Prometheus Design

Prometheus	3APL
Plan Descriptions:	
•Triggering event	Guards of reactive rules
•Plan steps	•Plan expressions
•Context of performing plans	•Guards of rules, test actions, action pre-conditions
•Data used/produced	•Beliefs, Java data, action post-condition, communication
Event Descriptions:	
•Event purpose	•Reasoning rules
•Data carried by event	•Substitutions in reasoning rules
Data Descriptions:	
•Data structures	•Terms, atoms, rules, Java data
•Methods manipulating data	•Actions

3APL Platform

- 3-APL is not only a language, but also an agent platform
- Provides:
  - A user interface to program, load and execute agents
  - Monitoring of message exchange
  - Monitoring mental states

3APL Platform

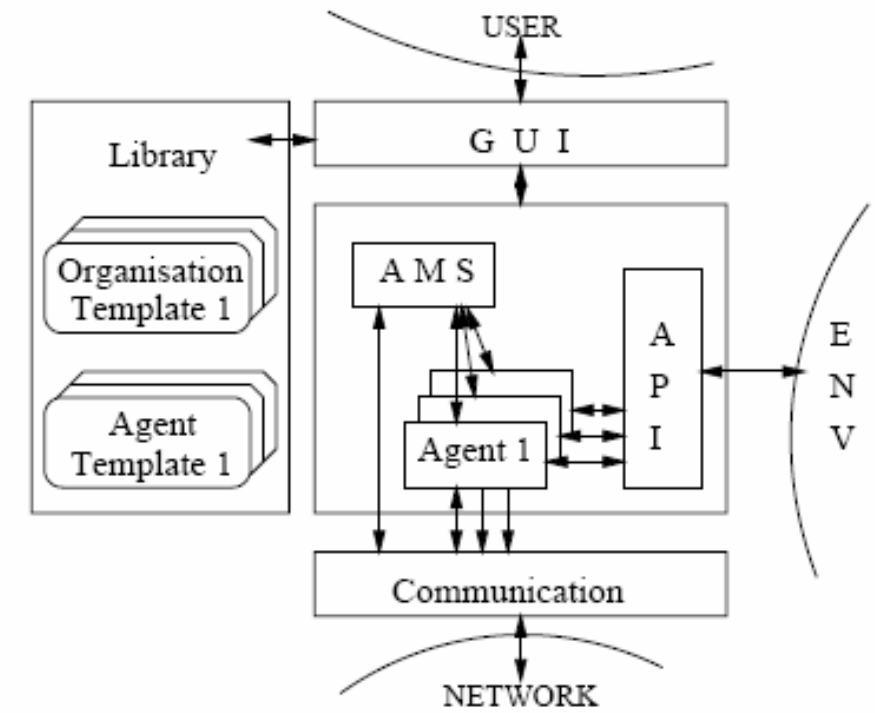


Figure 2. 3APL Development Platform

3APL Platform

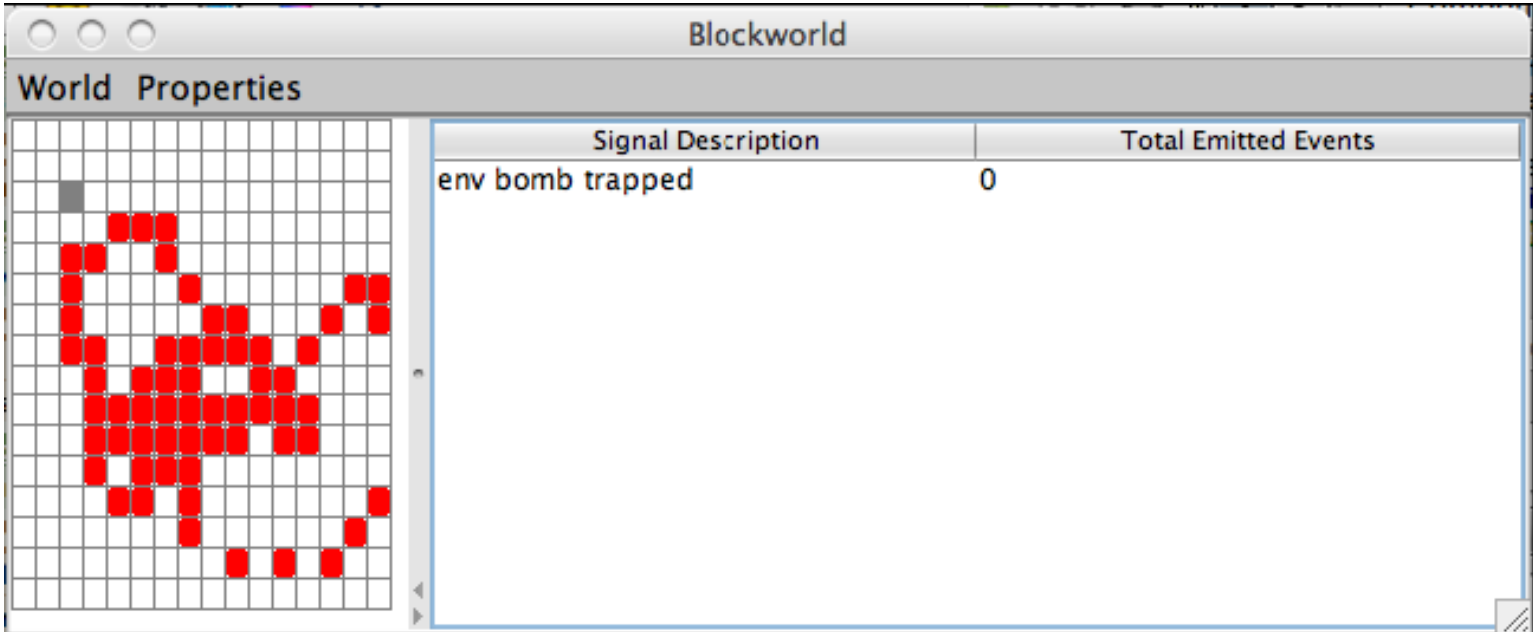
- Works on Java 1.5
- Easy to install, 850Kb jarfile with everything
- Each instance can be used as client or server

3APL Platform

- Easy connection with Java (”Plugins”)
- Example: Java("BlockWorld", east(), L);
- 3-APL provides a set of Java interfaces to implement the plugin
- Includes classes for graphical I/O

3APL Platform

- Graphical I/O: interaction between an external model and the agents!
- Example: BlockWorld included in the default library



- This allows us to code agents that make actions in a model without having to code the user interface

3APL Platform

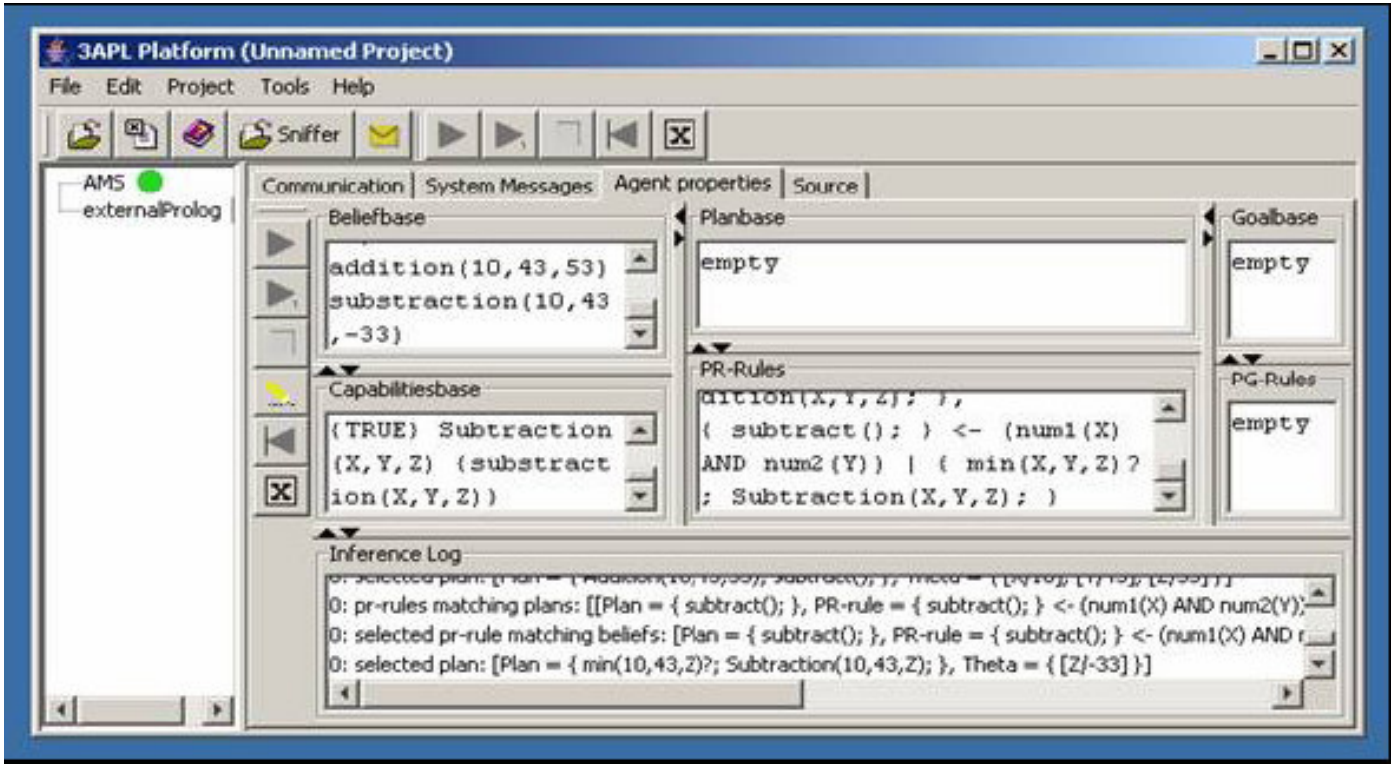
- Easy connection with Prolog (JIProlog)
- Example: LOAD ”cal.pl”
- Prolog clauses are added to the belief base
- Limitation: cannot be dynamic clauses

Communication

- Communication is quite simple, but also FIPA-compliant
- Interactions can be held between 3-APL and JADE agents
- Communication format: Send(Receiver, Performative, Content)
  - sent(Receiver, Performative, Content) is added to the sender belief base
  - received(Sender, Performative, Content) is added to the receiver belief base
- I BUT: Ontologies are not supported!

3APL Tools

- 3-APL default installation includes a visual interface



- Source code editing, running, monitoring and debugging
- 3-APL platform can work in network both as client and server
- Tested in Windows, MacOS and Linux

3APL Tools

- 3APL-M: version of 3-APL for mobile devices
- Binaries for J2SE and J2ME
- Little adaptation from 3-APL code has to be done

3APL Tools

- 3APL has been used in mobile robots
- There is a Haskell version of the 3-APL interpreter (with limitations)

Comparison

	Implementations	Formal Semantics	Industrial-strength applic.
PRS	UMPRS, PRS-CL, others	No	Yes
dMARS	In 1995, AAII implemented a C++ platform running on Unix; in 1997 dMARS was ported to Windows/NT	Operational	Yes
JACK	Java	No	Unmanned vehicle
JAM	Java	No	No
Jadex	Java	Operational	Yes
AS(L)	SIM Speak, AgentTalk, Jason	Operational	Virtual environments
3APL	Java and Prolog	Operational; meta-level	No
Dribble	No	Operational, dynamic logic-based	No
Coo-BDI	Coo-AgentSpeak	Operational	No

Comparison

	Basic components	Operation cycle	Ont	Dyn
PRS	Standard	Standard	No	No
dMARS	Standard	Standard	No	No
JACK	Standard + capabilities (that aggregate functional components) + views (to easily model data)	Standard	No	No
JAM	Standard + observer (user-specified declarative procedure that the agent interleaves between plan steps) + utility of plans	Utility-based	No	Yes
Jadex	Beliefs + goals + plans + capabilities (that aggregate functional components)	Standard	Yes	No
AS(L)	Standard	Standard; efficient	Yes	Yes
3APL	Beliefs, plans, practical reasoning rules, basic action specifications	Think-act	No	Yes
Dribble	Beliefs, plans, declarative goals, practical reasoning rules, goal rules, basic action specifications	Think-act	No	Yes
Coo-BDI	Standard + cooperation strategy (trusted agents + plan retrieval and acquisition policies) + plans' access specifiers	Perceive-cooperate-act	No	Yes

References

- Mehdi Dastani, 3APL Platform: User Guide, 19th January 2006, www.cs.uu.nl/3apl/download/java/userguide.pdf; accessed November 18, 2006. BNF specification of 3APL programming language
- M. Dastani, F. de Boer, F. Dignum, J.J. Meyer, *Programming Agent Deliberation: An Approach Illustrated Using the 3APL Language*. Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS'03), Melbourne, July 2003, ACM Press, 2003.
- M. d'Inverno, K. V. Hindriks, and M. Luck, "A formal architecture for the 3APL agent programming language," in Proc. of ZB'00, 2000, pp. 168–187.
- Marko Verbeek, 3APL as Programming Language for Cognitive Robots, Master Thesis.
- J.J. Meyer, (Cognitive) Agents Agent metaphor, http://www.siks.nl/act/jj-siksday-2005.pdf, accessed November 18, 2006.
- K. Hindriks, Mark d'Inverno and M. Luck, Architecture for Agent Programming Languages, Proceedings of the 14th European Conference on Artificial Intelligence W. Horn (ed.) , 363-367, IOS Press, 2000.
- Dastani, M., Hulstijn, J., Dignum, F. Meyer, J-J. Ch (2004) *Issues in Multiagent System Development*. Proceedings AAMAS'04, New York.
- Dastani, M.M., Riemsdijk, M.B. van, & Meyer, J-J.Ch. (2005). Programming Multi-Agents Systems in 3APL. In R. H. Bordini, M. Dastani, J. Dix, & A El Fallah Seghrouchni (Eds.), *Multi-Agent Programming (Languages, Platforms and Applications)* (pp. 39-67). New York: Springer Science.
- Mehdi Dastani, 3APL: A Programming Language for Multi-agent Systems (Syntax), http://www.cs.uu.nl/docs/vakken/map/slides/3apl-Syntax.pdf, accessed November 18, 2006