Multi-Agent System Design: Analysis of the 2APL alanguage

Alex Aushev Salvador Medina Carles Miralles Alejandro Suárez

Universitat Politècnica de Catalunya

March 2017

Overview

Introduction

- BDI-based agent-oriented programming language.
- Provides **programming constructions** to express the concepts in existing agent-oriented methodologies: *Goals and Beliefs, Actions, Reasoning rules and Plans.*
- Agent **deliberation cycle** to process agent constructions.
- Has **formal semantics** that allows verify whether agent programs satisfy their specification.
- Integrates **declarative and imperative** programming paradigms.

Beliefs and Goals

Beliefs

Definition

Information available to agent, expressed as facts and rules in Prolog. All facts are assumed to be ground.

Syntax

Bealiefs:

```
pos(1,1).
hasGold(0).
trash(2,3).
clean(blockworld) :- not trash(_,_).
```

Goals

Definition

List of logical formulas each of one denotes a situation the agent wants to realize (not necessary all at once).

Syntax

Goals:

hasGold(5) and clean(blockworld), hasGold(10)

(G1) agent gets 5 units of gold and the blockworld is clean
(G2) the agent gets 10 units of gold (regardless the blockworld state)

Actions

Actions (I)

- **Capabilities** of an agent
- **Passive** entities, **activated** in reasoning rules and plans
- There are 5 kinds of:
  - Update beliefs
  - Goals management
  - Agent communication
  - Environment interaction
  - Beliefs and Goals queries

Actions (II)

Update beliefs: {Pre} Action {Post}

BeliefUpdates:

{not carry(gold)} **PickUp()** {carry(gold)}

{trash(X,Y) and pos(X,Y)} **RemoveTrash()** {not trash(X,Y)}

Goals dynamics: adopt/drop

adopta(ϕ), adoptz(ϕ) : Add the goal ϕ at the begining/end.

dropgoal(ϕ), dropsubgoal(ϕ), dropsupergoal(ϕ): drop goal ϕ, all subgoals of ϕ and all supergoals of ϕ.

Agent communication: send

send(Receiver,Performative, Language,Ontology,Content)
send(Receiver,Performative,Content). Simplified when is assumed a Language & Ontology between agents.

Actions (III)

Environment interaction: @env(ActionName,Return)

env: Java class.
ActionName: method that modifies the environment.
Return: List of result values or empty list

@blockworld(east(),L): go one step east in the blockworld environment.

Test beliefs and goals: B()/G()

To check whether an agent has a belief/goal.
Beliefs: p(a)
Goals: q(b)

B(p(X)) & G(q(X)) **Fails** (not unifier), block the execution of the plan
B(p(X) & G(q(Y)) **Success** {X/a, Y/b}, instantiate variables in subsequent actions in the plan.

Plans and Reasoning Rules

Plans

- **Imperative** constructions to achieve the goals of an agent
- Sequence of actions composed by sequence/conditional/loop operator
- Initial plan: how to start
- Goal Plans: How to achieve goals
- Recovery Plans: backup plans
- Procedure Plans: plans for external events or input messages or abstract actions

Plans

Initial Plan

Plans:

$[@blockworld(enter(5,5,red),L); ChPos(5,5)]$
$send(admin, request, register(me))$

Updates my beliefs base with initial position 5,5 after entering into the blockworld ([]Atomic).
Request the administrator to register him.

Planning goal rules (PG-Rules)

Syntax

$[< goalquery >] \leftarrow < belquery > | < plan >$

- Generates a plan if agent has goals in goalquery and fulfill belief conditions.
- Goal is optional which means that a plan can be generated only based on beliefs conditions.

Example

PG-Rules:

$$clean(R) \leftarrow pos(X1, Y1)\ and\ trash(X2, Y2) \mid \{[goto(X1, Y1, X2, Y2); RemoveTrash()]\}$$

- If agent has the clean goal it beliefs that there are trash in X2,Y2 then a plan is generated for going from X1,Y1 to X2,Y2 and remove the trash
- goto() is an abstract action (next section)

Procedure call Rules (PC-Rules)

Definition

It allows generate plans as response to:

- **messages** sent by other agents
- **events** generated by external environment
- **the execution** of abstract actions

Example

PC-rules:

```
message(A,inform,La,On,goldAt(X2,Y2)) ← not carry(gold) |
                { getAndStoreGold(X2,Y2) }
event(gold(X2,Y2),blockworld) ← not carry(gold) |
                { getAndStoreGold(X2,Y2) }
getAndStoreGold(X,Y) ← pos(X1,Y1) | {
        [ goTo(X1,Y1,X,Y);@blockworld(pickup(),_); PickUp();
        goTo(X,Y,s3,s3);@blockworld(drop(),_);StoreGold() ] }
```

Plan repair Rules (PR-Rules)

Definition

$< plan > \leftarrow < belquery > \mid < plan >$

A plan fails if:

- The precondition of a belief update action is not entailed by the belief base
- If there is not applicable procedure rule for an abstract action
- If an external actions throws an ExternalActionFailedException
- The agent doesn't have access to an external environment
- An external action is not defined for an enviroment
- A test expression is not entailed by the belief and goal bases
- A goal being adopted that is already in the belief base or the goal is not ground
- An atomic plan if one action of the plan fails

Plan repair Rules (PR-Rules)

Example

PR-Rules:

$$@blockworld(east(),\_);@blockworld(east(),\_);X \leftarrow true \mid \{@blockworld(north(),);@blockworld(east(),);@blockworld(east(),);@blockworld(south(),);X\}$$

A **failing plan** starting with $@blockworld(east(),\_);@blockworld(east(),\_);$ generates this repair plan (X is whatever other action)

2APL Semantics

2APL Configuration (I)

Individual Agent

The configuration of an individual 2APL agent is defied as $A_\iota = \{\iota, \sigma, \gamma, \pi, \theta, \epsilon\}$.

Where:

- ι is the Agent's Identifier
- σ is a set of belief expressions
- γ is a list of goal expressions
- π is a set of plans in (π, r, p) where π is the plan, r are the PG_rules and p is the identifier
- θ is a ground substitution (binds variables)
- $\epsilon = E, I, M$ where E,I and M are external events, failed plans and messages respectively

2APL Configuration (II)

Individual Agent

The configuration of a 2APL multi-agents system is defined as $\{A_1, ..., A_n, \chi\}$.

Where:

- A_i is the configuration of Agent i
- χ is a set of external shared environments
- In the initial state, $\theta = \emptyset$ and $\epsilon = \{\emptyset, \emptyset, \emptyset\}$ for all agents
- The configuration is updated at each iteration by means of the transition rules

Multi-Agent transition rules

Multi-Agent transition rules (I)

Basic Actions

$\frac{A_i \rightarrow A_i'}{\{A_1,...,A_i,..,A_n,\chi\} \rightarrow \{A_1,...,A_i',..,A_n,\chi\}}$, this is, basic actions of agent i do not affect the environment nor other agents

External Actions (I)

$\frac{F^{env}_{\iota,\alpha}(t_1,...,t_n,\chi)=(t,\chi')}{\chi \xrightarrow{env(\iota,\alpha(t_1,...,t_n),t)} \chi'}$, this is, when an external action is broadcasted, the environment can make a transition according to it

External Actions (II)

$\frac{A_i \xrightarrow{env(\iota,\alpha(t_1,...,t_n),t)} A_i' \&\& \chi \xrightarrow{env(\iota,\alpha(t_1,...,t_n),t)} \chi'}{\{A_1,...,A_i,..,A_n,\chi\} \rightarrow \{A_1,...,A_i',..,A_n,\chi'\}}$, this is, if a external action by agent A_i changes the environment, it can be the only one noticing it

Multi-Agent transition rules (II)

Internal environment dynamics

$\frac{env \in \chi \& env \xrightarrow{(\theta,(k,\ldots,l))!} env'}{\{A_1,\ldots,A_i,..,A_n,\chi\} \rightarrow \{A'_1,\ldots,A'_i,..,A'_n,\chi'\}}$, this is, which agents receives an event is the decision of the environment designer

Message passing

$\frac{A_i \xrightarrow{\varphi!} A'_i \& A_j \xrightarrow{\varphi?} A'_j}{\{A_1,\ldots,A_i,A_j,\ldots,A_n,\chi\} \rightarrow \{A'_1,\ldots,A'_i,A'_j,\ldots,A'_n,\chi'\}}$, this is, when an agent sends a message to another agent, both transitions (send and receive) are done synchronously

Execution of Multi-Agent Systems

Semantics of Computational Run

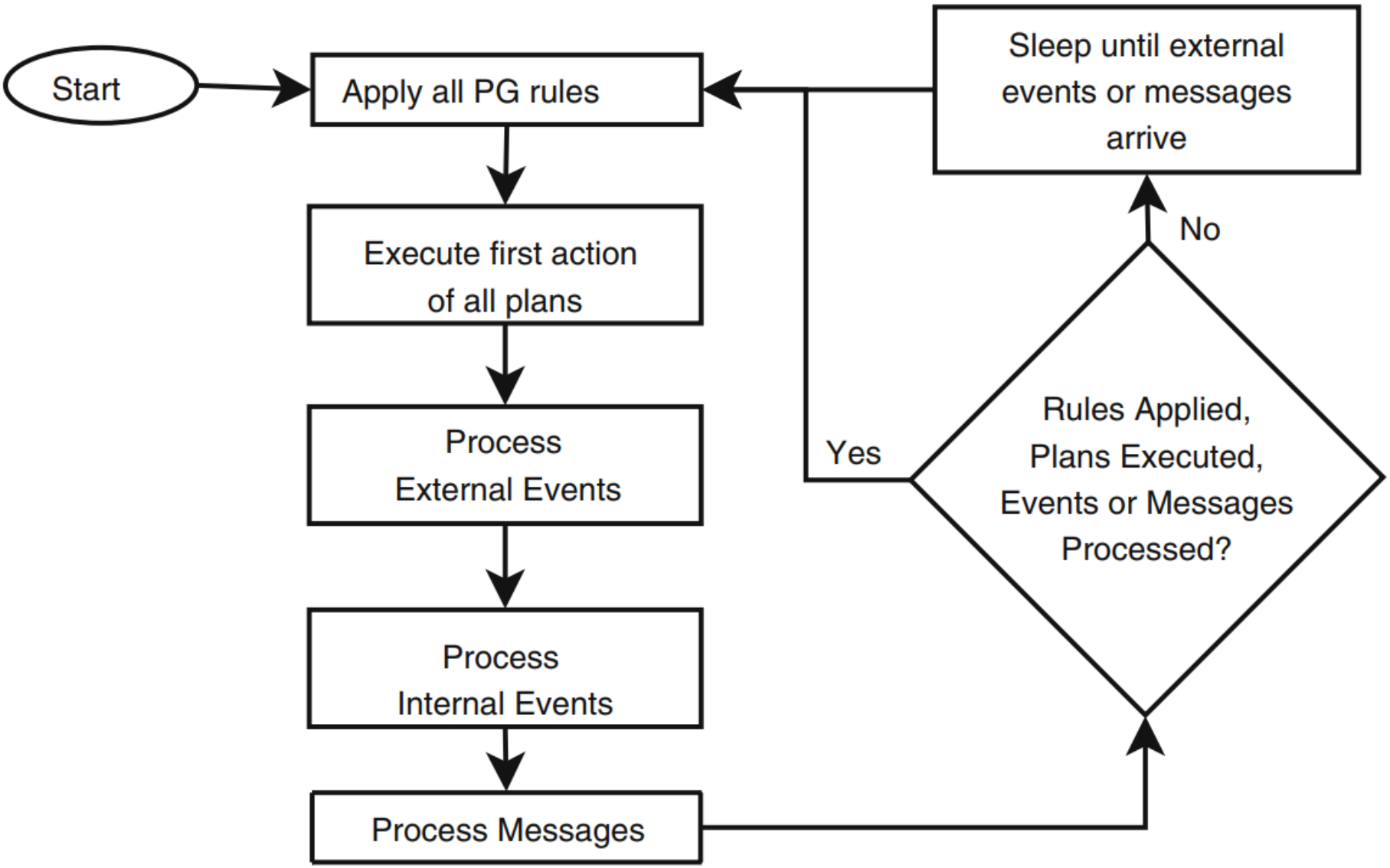
Definition

The execution of a 2APL multi-agent system with initial configuration $A_1, ..., A_n, \chi$ is the set of computation runs $CR(A_1, ..., A_n, \chi)$ of the 2APL transition system.

Definition

A computation run $CR(s0)$ is a sequence $s_0, ...$ where s_i is a configuration, and $\forall i > 0 : s_{i1} \rightarrow s_i$ is a transition in the transition system.

Single-Agent Deliberation Cycle



Characteristics of the Deliberation Cycle (I)

- The interpreter of 2APL executes individual agents and the environment in parallel in an interleaving mode.
- Each cycle starts by applying all applicable PG-rules, each rule only one time.
- The DC proceeds by executing only the first actions of all plans. In order to allow all plans to get a chance to be executed.
- Internal and external events and messages are processed by applying the first applicable PC-rule.

Characteristics of the Deliberation Cycle (II)

- If the execution of a plan fails, then the plan will either be repaired in the same deliberation cycle or get re-executed in the next deliberation cycle.
- If the first action of a failed plan is a belief update, test, adopt goal, abstract or external action and there is no plan repair rule to repair it, then the failed plan may be successfully executed in the next deliberation cycle.

Blocks of Highlighted Text

Block 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Block 2

Pellentesque sed tellus purus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Vestibulum quis magna at risus dictum tempor eu vitae velit.

Block 3

Suspendisse tincidunt sagittis gravida. Curabitur condimentum, enim sed venenatis rutrum, ipsum neque consectetur orci, sed blandit justo nisi ac lacus.

Multiple Columns

Heading

1. Statement
2. Explanation
3. Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Table

Treatments	Response 1	Response 2
Treatment 1	0.0003262	0.562
Treatment 2	0.0015681	0.910
Treatment 3	0.0009271	0.296

Table: Table caption

Theorem

Theorem (Mass–energy equivalence)

$E = mc^2$

Verbatim

Theorem (Aasdf)

```
asdadsf
asdfasdf
     asdfasdf
```

Figure

Uncomment the code on this slide to include your own image from the same directory as the template .TeX file.

Citation

An example of the `\cite` command to cite within the presentation:

This statement requires citation [Smith, 2012].

References

John Smith (2012)

Title of the publication

Journal Name 12(3), 45 – 678.

The End