ENCAPSULATING REACTING BEHAVIOUR IN GOAL-BASED PLANS FOR PROGRAMMING BDI AGENTS

Rafael H. **Bordini**
School of Technology, PUCRS
Porto Alegre, RS, Brazil
rafael.bordini@pucrs.br

Rem **Collier**
University College of Dublin
Dublin, Ireland
rem.collier@ucd.ie

Jomi F. **Hübner**
DAS, Fed. Univ. of Santa Catarina
Florianópolis, SC, Brasil
jomi.hubner@ufsc.br

Alessandro **Ricci**
DISI, University of Bologna
Cesena, Italy
a.ricci@unibo.it

LAMAS@AAMAS 2020

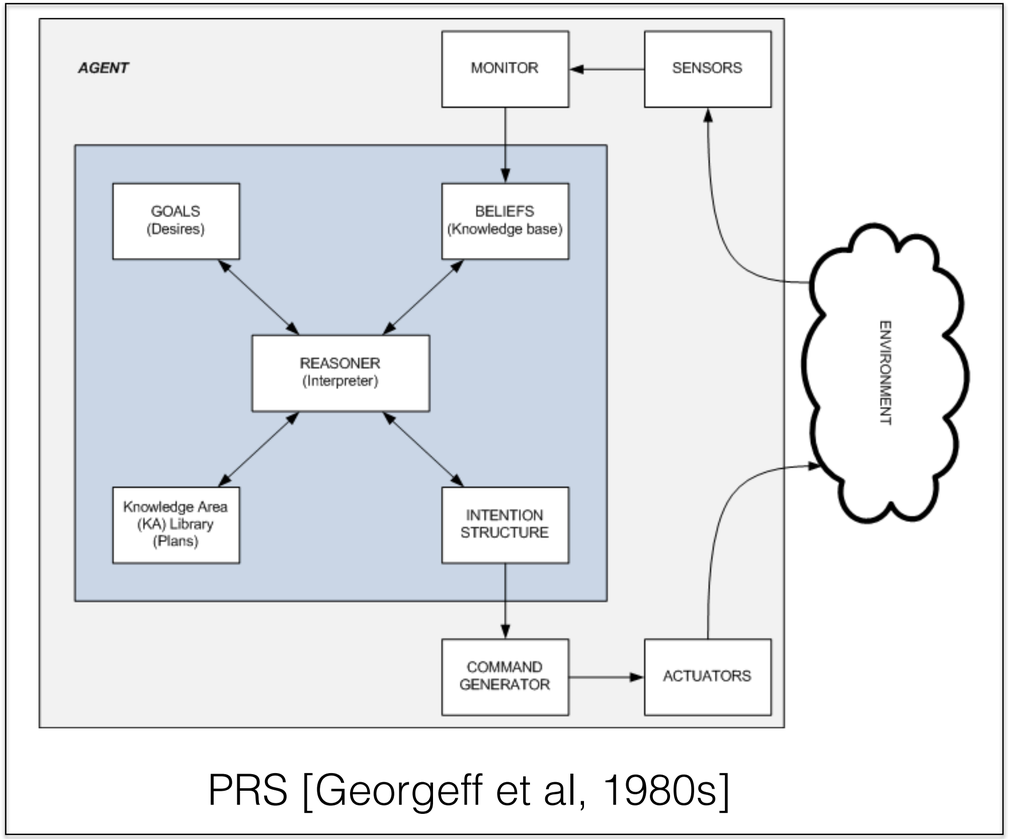
FIRST PART - OVERVIEW

- Context
  - BDI agent programming context
- Problem
  - weak encapsulation in plans
- Contribution
  - extended plan model
  - implementation in Jason and ASTRA

[BACKGROUND]

PLANS IN BDI AGENT PROGRAMMING

- Belief Desire Intention (BDI) model
- Plans and Intentions



PRS [Georgeff et al, 1980s]

[BACKGROUND]

PLANS IN BDI AGENT PROGRAMMING

- Belief Desire Intention (BDI) model
- Plans and Intentions

BDI Platforms, Frameworks, Languages
- dMARS, JAM, JACK, SPARK,…
- 3APL/2APL, GOAL, Jason, ASTRA,…

Abstract formal languages
- AgentSpeak(L), CAN

[BACKGROUND]

PLANS IN BDI AGENT PROGRAMMING

- Belief Desire Intention (BDI) model
- Plans and Intentions

plans
how to bring about a state of affairs

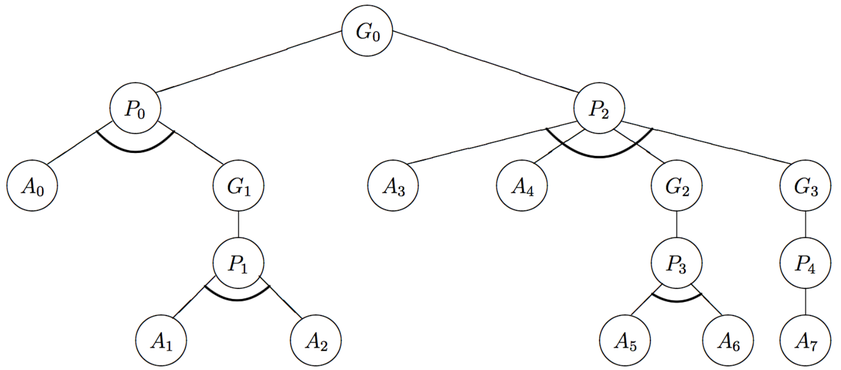
specifying the course of action to achieve such states of affairs

intentions
the activity used to achieve that state of affairs (runtime concept)

[PROBLEM]
WEAK ENCAPSULATION

- Plan encapsulation
- Weak encapsulation
- An example in Jason
- Drawbacks

plan specification should include (*encapsulate)*
- the **state of affairs** to achieve
- the **strategy** to bring about it



in the Goal-Plan Tree model (**GPT)**
- plan *p* and a parent goal *g*
- plan *p* and children nodes (strategy)

[PROBLEM]

WEAK ENCAPSULATION

- Plan encapsulation
- Weak encapsulation
- An example in Jason
- Drawbacks

Current BDI models and implementations:

- allow for specifying **plans with *no explicit state of affairs***

  in GPT => plan *p* with no parent goal *g*

- impossibility to encapsulate *reactive* behaviour in the strategy of the plan

  in GPT => reactive behaviour ?

➡ ***drawbacks***
- in the practice of agent programming
- agent reasoning at runtime

[PROBLEM]
WEAK ENCAPSULATION

- Plan encapsulation
- Weak encapsulation
- An example in Jason
- Drawbacks

```
+!cnp(I,Task)
  <- !announce_cfp(I,Task);
     !bids(I).

+!announce_cfp(I,Task) <- ...

+!bids(I)
  <- .wait(4000);
     !contract(I).

+propose(I,_) : all_ans(I) <- !contract(I).
+refuse(I)    : all_ans(I) <- !contract(I).

+!contract(I) : not .intend(contract(I)) <- ...
```

Contract Net Protocol sketch

[PROBLEM]
WEAK ENCAPSULATION

- Plan encapsulation
- Weak encapsulation
- An example in Jason
- Drawbacks

```
+propose(I,_) : all_ans(I) <- !contract(I).
```

reactive plans => goal-less intentions
the goal is in developer's mind
but not in the agent mind

reactive behaviour not encapsulated in the plan strategy
implemented as unrelated plans

=> hand-managed beliefs as a workaround

[PROPOSAL]
PLAN MODEL EXTENSION

- Revisiting the plan model
- The example revisited
- Formalisation & implementation

enforce goal/task specification

every plan has always a state of affairs to be achieved

in GPT => plan *p* has always a parent goal *g*

allow for encapsulating reactive behaviour in plan strategy

from reactive plans to reactive rules inside a plan

in GPT => (?)

[PROPOSAL]
PLAN MODEL EXTENSION

- Idea
- The example revisited (Jason-ER)
- Formalisation & implementation

```
+!cnp(I,Task) {
    <- !announce_cfp(I,Task);
       !bids(I);
       !contract(I).

    +!bids(I) {
        <- .wait(4000); .done.

        // reaction rules
        +propose(I,_) : all_ans(I) <- .done.
        +refuse(I)    : all_ans(I) <- .done.
    }

    +!announce_cfp(I,Task) <- ...
    +!contract(I) <- ...
}
```

[PROPOSAL]
PLAN MODEL EXTENSION

- Idea
- The example revisited
- Formalisation & implementation

- **abstract formal language** capturing the model
- **semantics:** extension of the reasoning cycle
- first implementations:
  - based on ***Jason*** and ***ASTRA***
  - available on github

[PROPOSAL]

CONCLUDING REMARKS

- Results so far
- Ongoing & Future work

idea evaluated using a selected set of programming examples

expected advantages brought by strong encapsulation
modularity, reusability, readability

no performance penalties

[PROPOSAL]

CONCLUDING REMARKS

- Results so far
- Ongoing & Future work

validating the approach with **more complex** agent/MAS programs and projects

feedbacks for improving & refining the approach by using it in practice

GPT-based formalisation

- *understanding behavioural properties*
- *agent reasoning at runtime*