Intention and Plan Selection for BDI Agent Systems

Sebastian Sardina

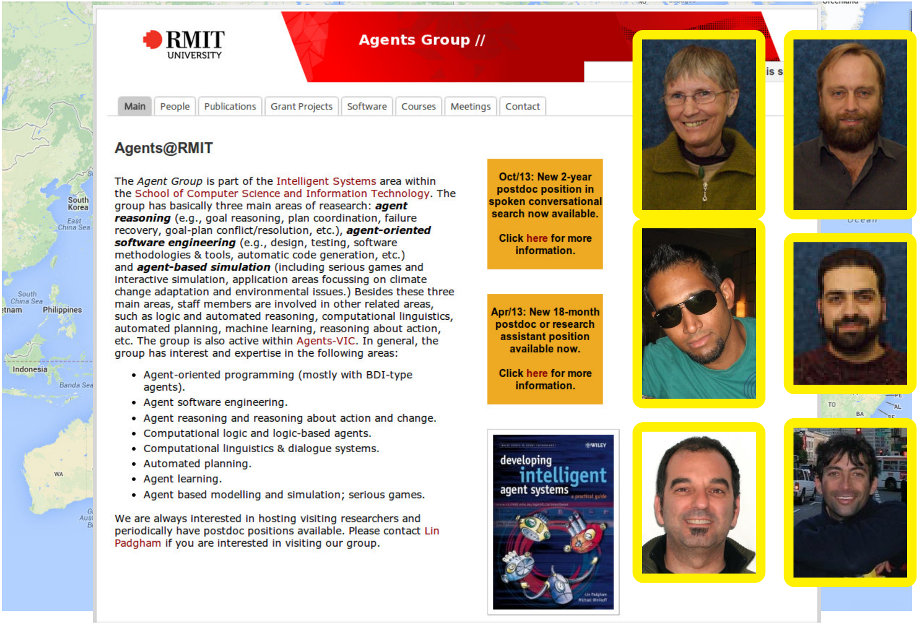
School of Computer Science and IT
RMIT University
Melbourne, Australia
sebastian.sardina@rmit.edu.au

RMIT UNIVERSITY

Joint work with Lin Padgham & others...

4TH WORKSHOP ON GOAL REASONING
IJCAI-2016
July 9th, 2016

RMIT Intelligent Agents Group



RMIT UNIVERSITY

Agents Group //

Main | People | Publications | Grant Projects | Software | Courses | Meetings | Contact

Agents@RMIT

The *Agent Group* is part of the Intelligent Systems area within the School of Computer Science and Information Technology. The group has basically three main areas of reasearch: ***agent reasoning*** (e.g., goal reasoning, plan coordination, failure recovery, goal-plan conflict/resolution, etc.), ***agent-oriented software engineering*** (e.g., design, testing, software methodologies & tools, automatic code generation, etc.) and ***agent-based simulation*** (including serious games and interactive simulation, application areas focussing on climate change adaptation and environmental issues.) Besides these three main areas, staff members are involved in other related areas, such as logic and automated reasoning, computational linguistics, automated planning, machine learning, reasoning about action, etc. The group is also active within Agents-VIC. In general, the group has interest and expertise in the following areas:

- Agent-oriented programming (mostly with BDI-type agents).
- Agent software engineering.
- Agent reasoning and reasoning about action and change.
- Computational logic and logic-based agents.
- Computational linguistics & dialogue systems.
- Automated planning.
- Agent learning.
- Agent based modelling and simulation; serious games.

We are always interested in hosting visiting researchers and periodically have postdoc positions available. Please contact Lin Padgham if you are interested in visiting our group.

Oct/13: New 2-year postdoc position in spoken conversational search now available.

Click here for more information.

Apr/13: New 18-month postdoc or research assistant position available now.

Click here for more information.

Intelligent Autonomous Behavior

1. **Behavior-based AI:** set of independent simple reactive modules.
   - intelligent behavior emerges "implicitly".
   - popular in robotics (since the '80).

Intelligent Autonomous Behavior

1. Behavior-based AI: set of independent simple reactive modules.
   - intelligent behavior emerges “implicitly”.
   - popular in robotics (since the ’80).
2. Agent-oriented programming: control specified by programmer.
   - BDI systems: JACK, JASON, 3APL, etc.
   - High-level languages: Golog-like languages, FLUX, etc.

Intelligent Autonomous Behavior

1. Behavior-based AI: set of independent simple reactive modules.
   - intelligent behavior emerges “implicitly”.
   - popular in robotics (since the ’80).
2. Agent-oriented programming: control specified by programmer.
   - BDI systems: JACK, JASON, 3APL, etc.
   - High-level languages: Golog-like languages, FLUX, etc.
3. Learning: learn how to act based on previous experience.
   - E.g., reinforcement learning.

Intelligent Autonomous Behavior

1. Behavior-based AI: set of independent simple reactive modules.
   - intelligent behavior emerges "implicitly".
   - popular in robotics (since the '80).
2. Agent-oriented programming: control specified by programmer.
   - BDI systems: JACK, JASON, 3APL, etc.
   - High-level languages: Golog-like languages, FLUX, etc.
3. Learning: learn how to act based on previous experience.
   - E.g., reinforcement learning.
4. Automated Planning: automatic synthesis of behavior from model.
   - Input: model of the world + initial state + goal to be achieved.
   - Output: plan or controller to achieve the goal in the world.

Intelligent Autonomous Behavior

2. Agent-oriented programming: control specified by programmer.
    - BDI systems: JACK, JASON, 3APL, etc.
    - High-level languages: Golog-like languages, FLUX, etc.

Here!

What are we after?

1. Understand what constitute **“rational behavior”** & an **“agent”**.
    - Theory of Practical Reasoning.
    - Informed by:
        - Philosophy of mind.
        - Phsycology.
        - Computer Science.

2. Find ways to **design** and **program** agent systems
    - Informed by what rational behavior is...
    - Two areas:
        - Agent-oriented Software Engineering.
        - Agent-oriented Programming.

What are we after?

2. Find ways to **design** and **program** agent systems
    - Informed by what rational behavior is...
    - Two areas:
        - Agent-oriented Software Engineering.
        - Agent-oriented Programming.

Agent Systems

An **intelligent agent** is an autonomous entity, existing over time in a dynamic environment, that is able to rationally balance pro-active and reactive behavior. It perceives through sensors and acts through effectors.

Agent Systems

An **intelligent agent** is an autonomous entity, existing over time in a dynamic environment, that is able to rationally balance pro-active and reactive behavior. It perceives through sensors and acts through effectors.

- **autonomy:** does not require continuous external control.
- **pro-activity:** pursues goals over time; goal directed behavior.
- **reactivity:** perceives the environment and responds to it.
- **situatedness:** observe & act in the environment.
- **flexibility:** achieve goals in several ways.
- **robustness:** will try hard to achieve goals.

And also: **modular scalability** & **adaptability!**

Agent Systems

An **intelligent agent** is an autonomous entity, existing over time in a dynamic environment, that is able to rationally balance pro-active and reactive behavior. It perceives through sensors and acts through effectors.

autonomy: does not require continuous external control.

pro-activity: pursues goals over time; goal directed b

reactivity: perceives the environment an

situatedness: observe & act in the enviro

flexibility: achieve goals in several ways.

robustness: will try hard to achieve goals.

goal-orientation!

And also: **modular scalability** & **adaptability!**

Technology Development

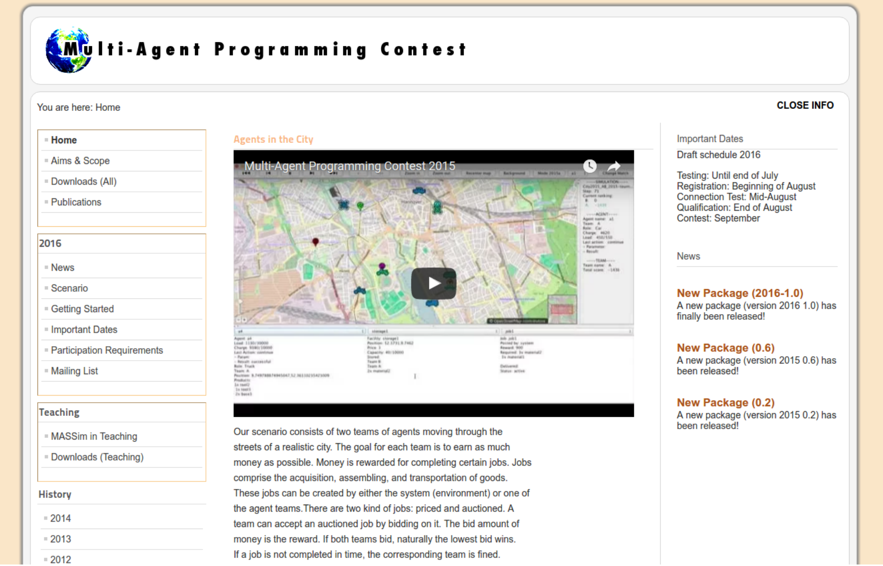
abstraction level
distribution
complexity of domain

Agent Oriented Programming (BDI systems)
Distributed Control - Multi-agent frameworks (JADE)

Object Oriented programming (C++, Java, Delphi)
Client/Server - Remote Procedure Call (CORBA)

Structured programming (FORTRAN, C)
Monolithic systems - Communication API (sockets)

International Multi-Agent Contest



https://multiagentcontest.org/

International Multi-Agent Contest

Multi-Agent Programming Contest

- stimulate research in multi-agent system **development and programming**;
- identifying key **problems**;
- collecting suitable **benchmarks**;
- gather **test cases**;
- test multi-agent prog. **languages, platforms, tools.**

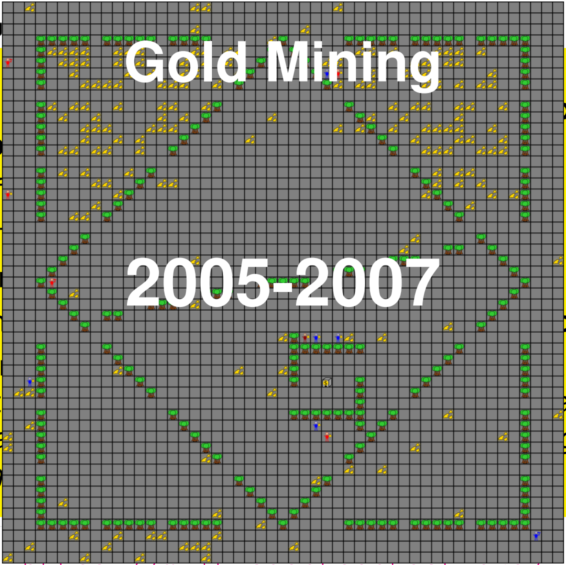
We encourage submissions that specify and design a multi-agent system in terms of **high-level concepts** such as *goals, beliefs, plans, roles, communication, coordination, negotiation, and dialogue* in order to ...

2012

If a job is not completed in time, the corresponding team is fined.

https://multiagentcontest.org/

International Multi-Agent Contest

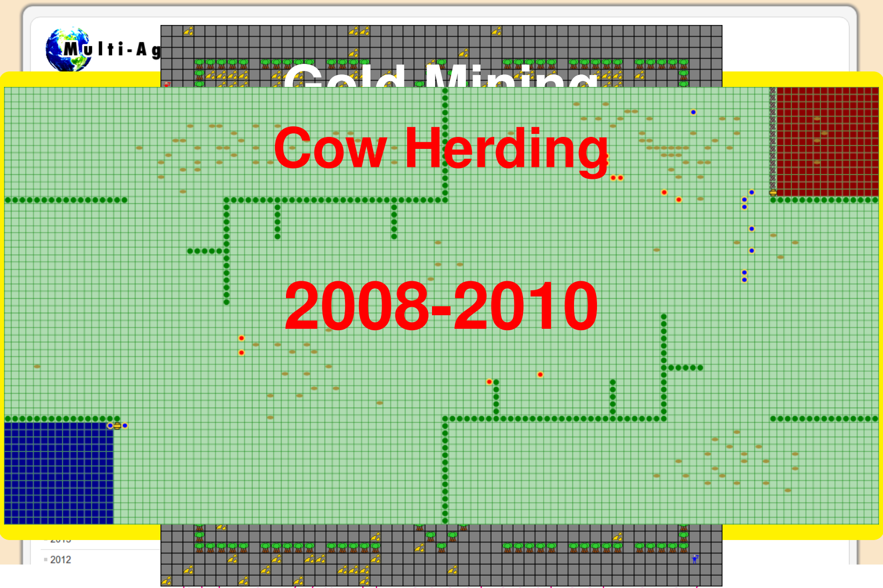


- stimul… …pment **and p…**
- identif…
- collec…
- gathe…
- test m… …**ols.**

We encou… multi- agent syst… …as *goals, beliefs, pla… …gotiation, and dialog…*

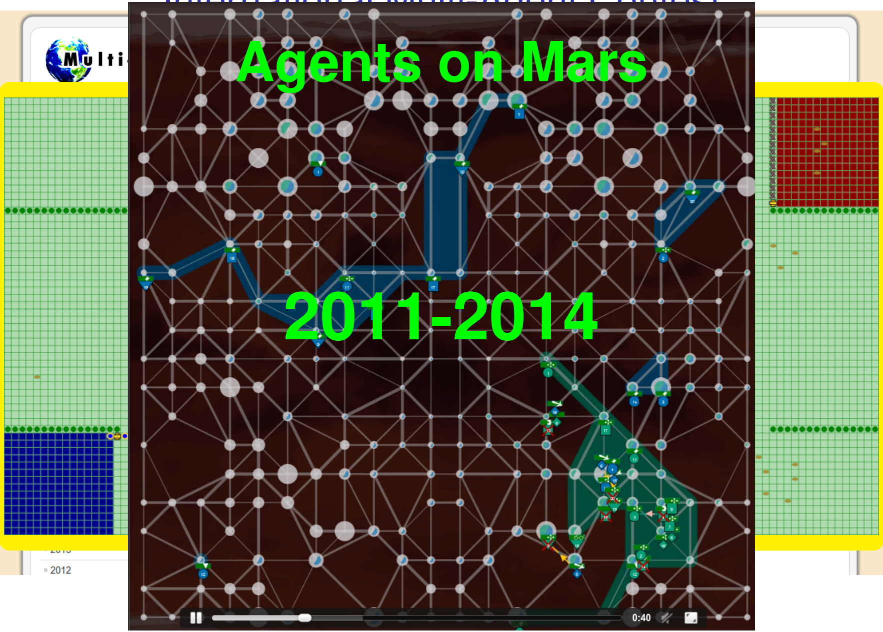
https://multiagentcontest.org/

International Multi-Agent Contest

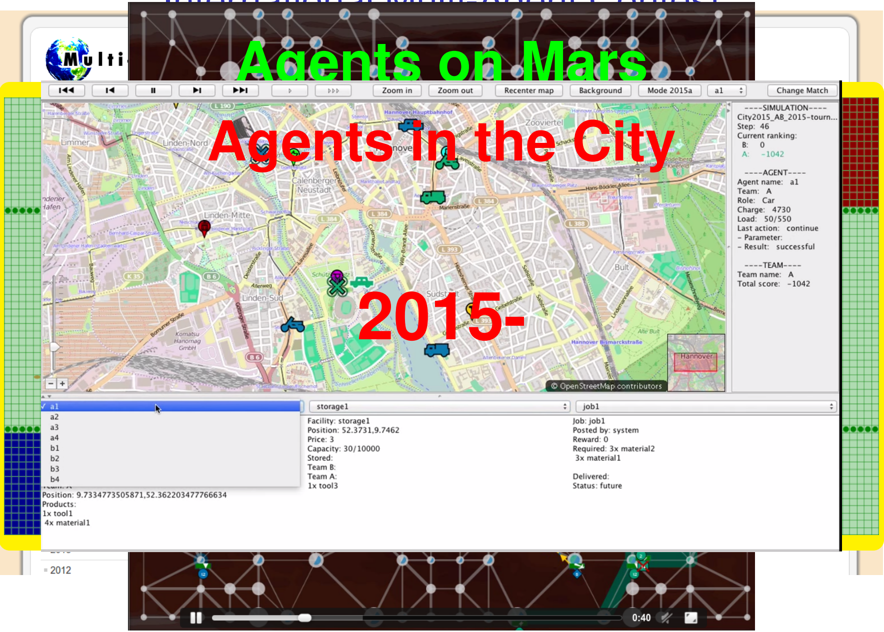


https://multiagentcontest.org/

International Multi-Agent Contest

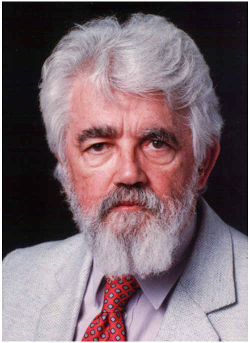


International Multi-Agent Contest



Intentional Stance for Computer Systems?

"To ascribe beliefs, free will, intentions, consciousness, abilities, or wants to a machine is legitimate when such an ascription expresses the same information about the machine that it expresses about a person. It is useful when the ascription helps us understand the structure of the machine, its past or future behavior, or how to repair or improve it. [...] "



John McCarthy

Question

How do we make all these ideas a concrete computational approach?

BDI Model: Thinking of and building "rational" systems

Plans and resource-bounded practical reasoning

MICHAEL E. BRATMAN
Department of Philosophy and Center for the Study of Language and Information, Stanford University, Stanford, CA 94305, U.S.A.

AND

DAVID J. ISRAEL AND MARTHA E. POLLACK
Artificial Intelligence Center and Center for the Study of Language and Information, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, U.S.A.

Received September 13, 1987

Revision accepted September 19, 1988

An architecture for a rational agent must allow for means-end reasoning, for the weighing of competing alternatives, and for interactions betwen these two forms of reasoning. Such an architecture must also address the problem of resource boundedness. We sketch a solution of the first problem that points the way to a solution of the second. In particular, we present a high-level specification of the practical-reasoning component of an architecture for a resource-bounded rational agent. In this architecture, a major role of the agent's plans is to constrain the amount of further practical reasoning she must perform.

Key words: planning, practical reasoning, resource bounds.

L'architecture d'un agent rationnel doit permettre le raisonnement procédant des fins aux moyens, le choix entre différentes actions possibles, et l'interaction entre ces deux modes de raisonnement. Elle doit aussi tenir compte des conséquences des limites de ressources disponibles. Nous esquissons ici une solution au premier problème qui indique comment on pourrait résoudre le second. Nous proposons, en particulier, une spécification abstraite d'un module de génération de plans pour un agent rationnel dont les ressources sont bornées. Dans cette architecture, le rôle principal des plans d'un agent est de limiter les ressources devant être consacrés au raisonnement.

Mots clés : planification, raisonnement pratique, limites de ressources.

Comput. Intell. **4**, 349–355 (1988)

IRMA Architecture

(Intelligent Resource-bounded Machine Architecture)

IRMA Architecture [Bratman, Israel, Pollack CI'88]

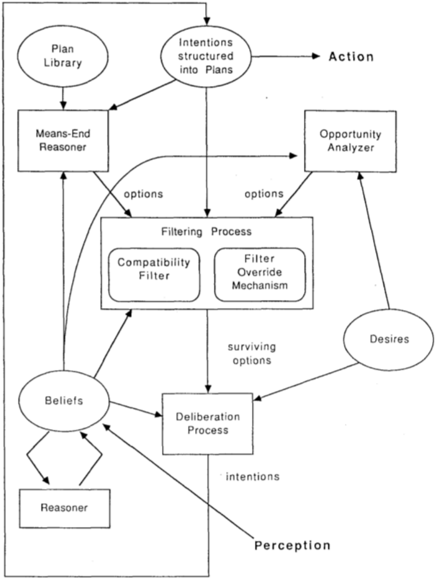
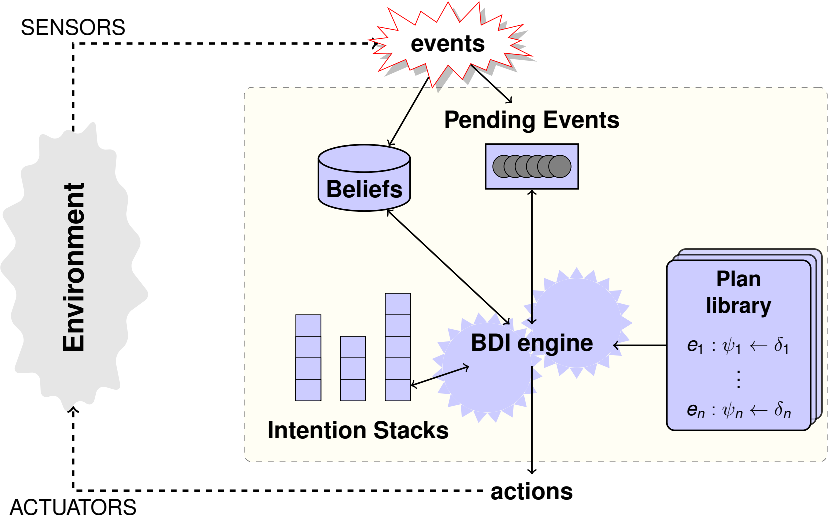
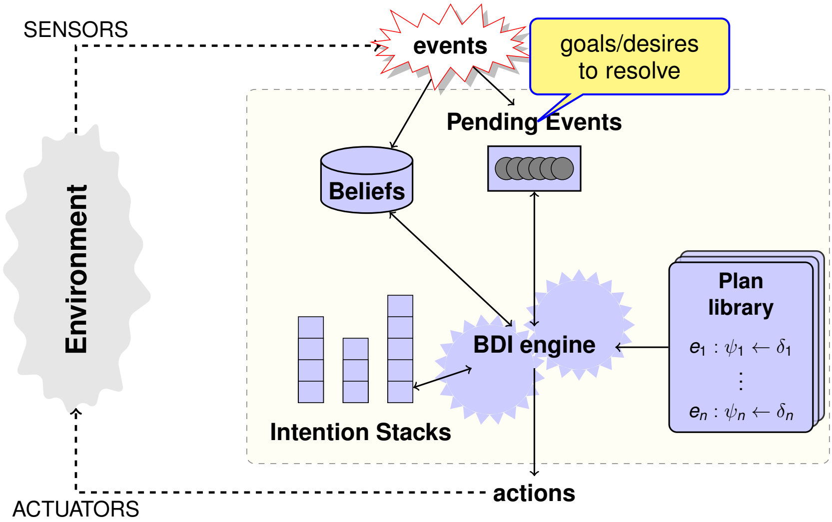


FIG. 1. An architecture for resource-bounded agents.

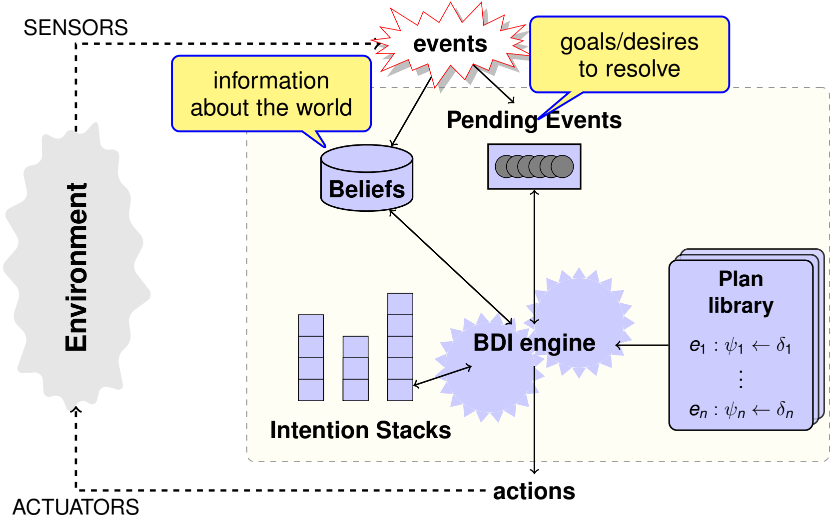
Detailed BDI Architecture



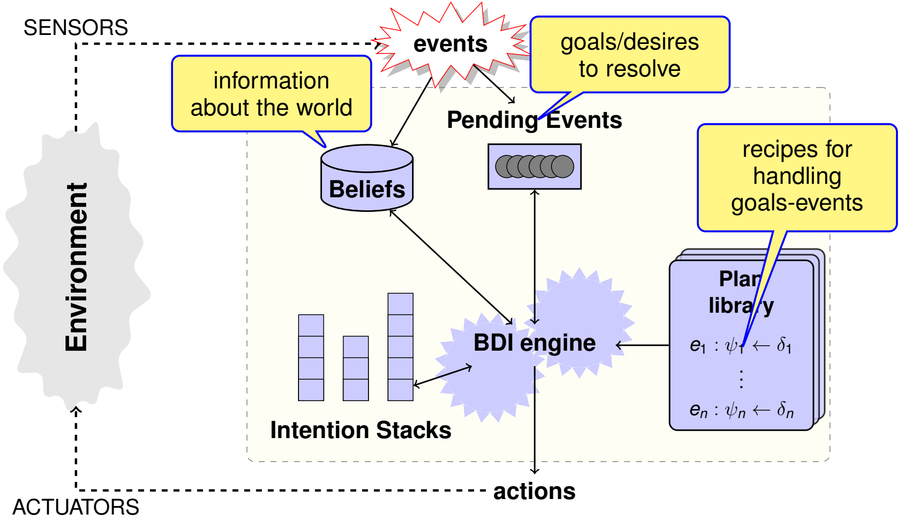
Detailed BDI Architecture



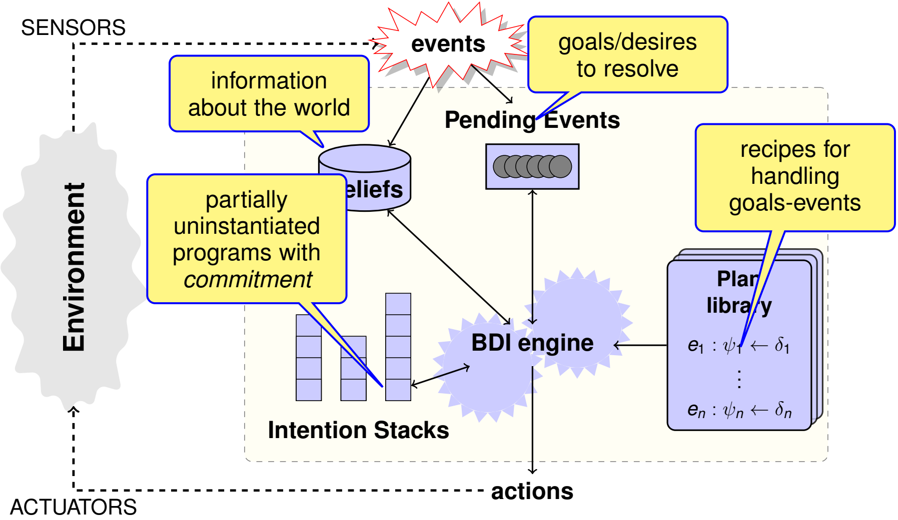
Detailed BDI Architecture



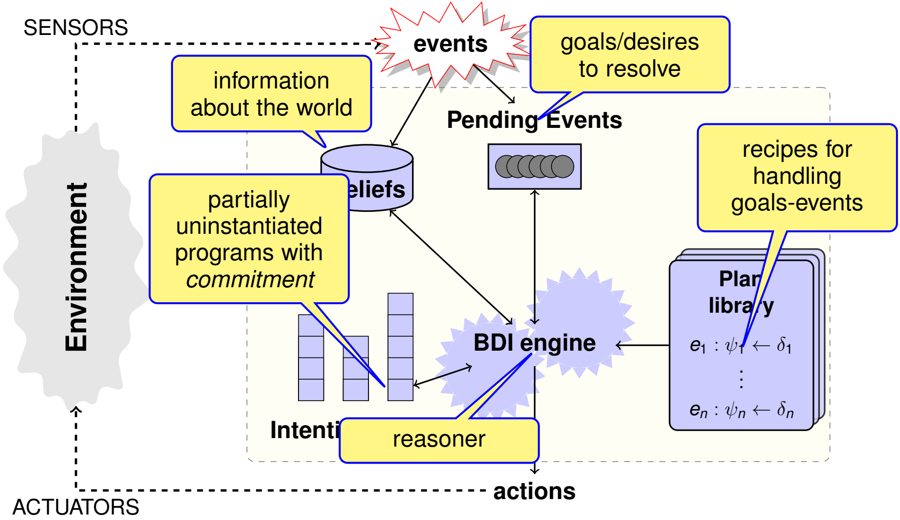
Detailed BDI Architecture



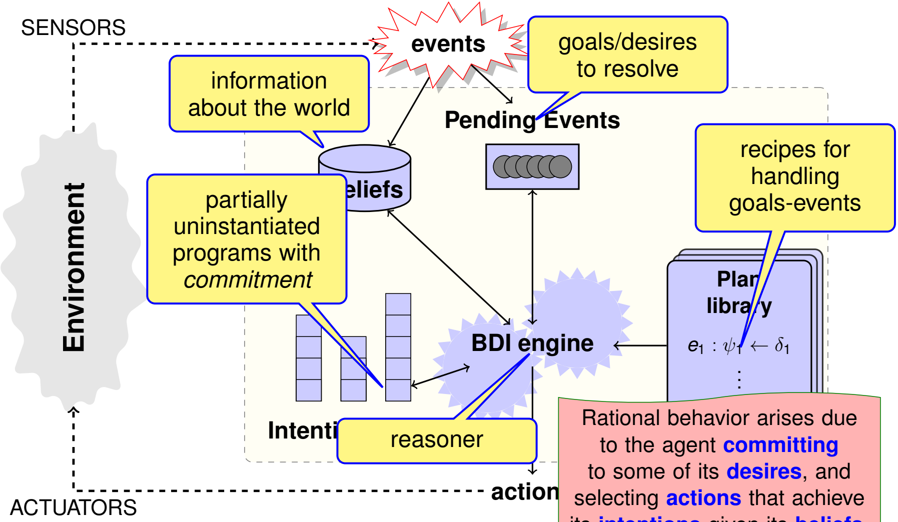
Detailed BDI Architecture



Detailed BDI Architecture



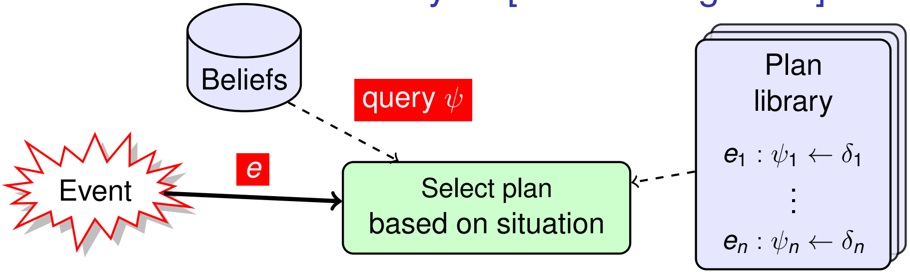
Detailed BDI Architecture



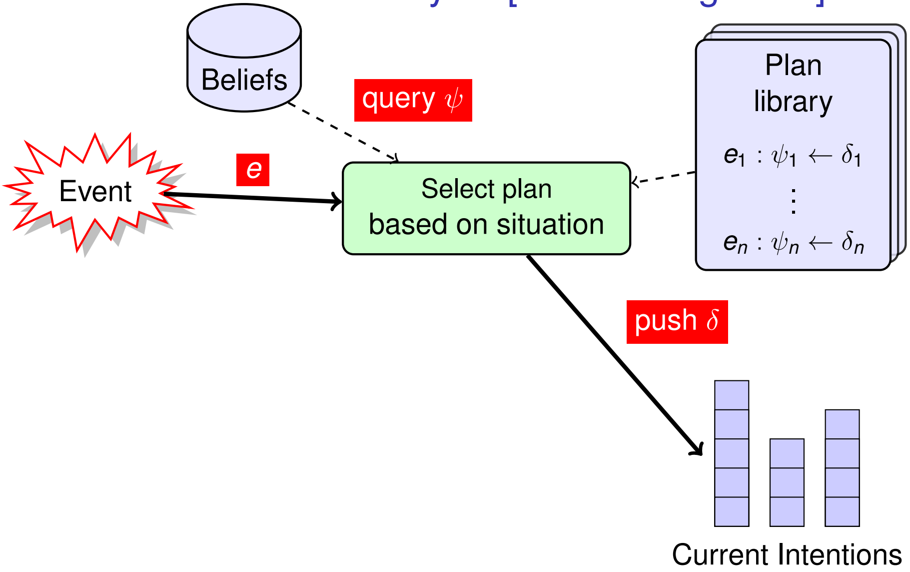
Rational behavior arises due to the agent **committing** to some of its **desires**, and selecting **actions** that achieve its **intentions** given its **beliefs**.

The BDI Execution Cycle [Rao&Georgeff 92]

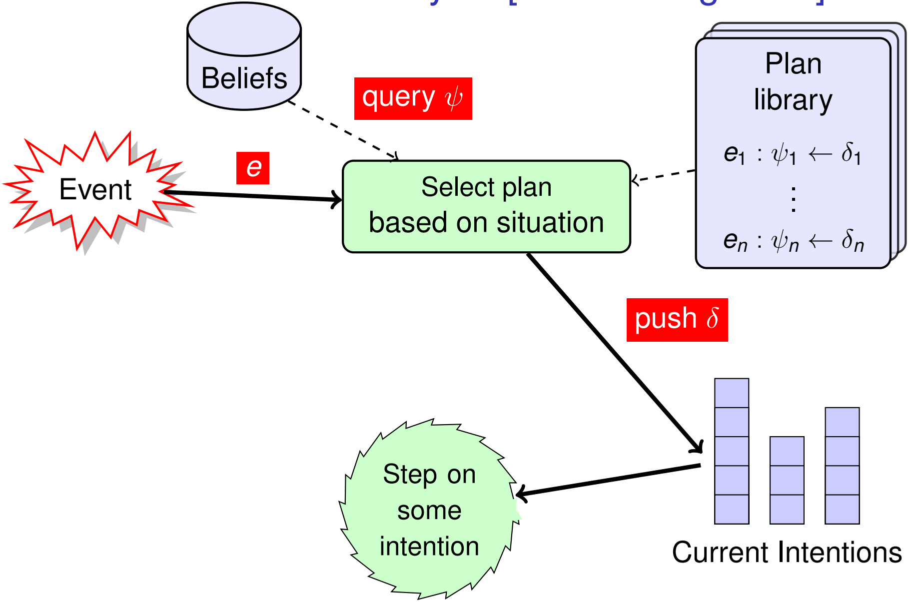
The BDI Execution Cycle [Rao&Georgeff 92]



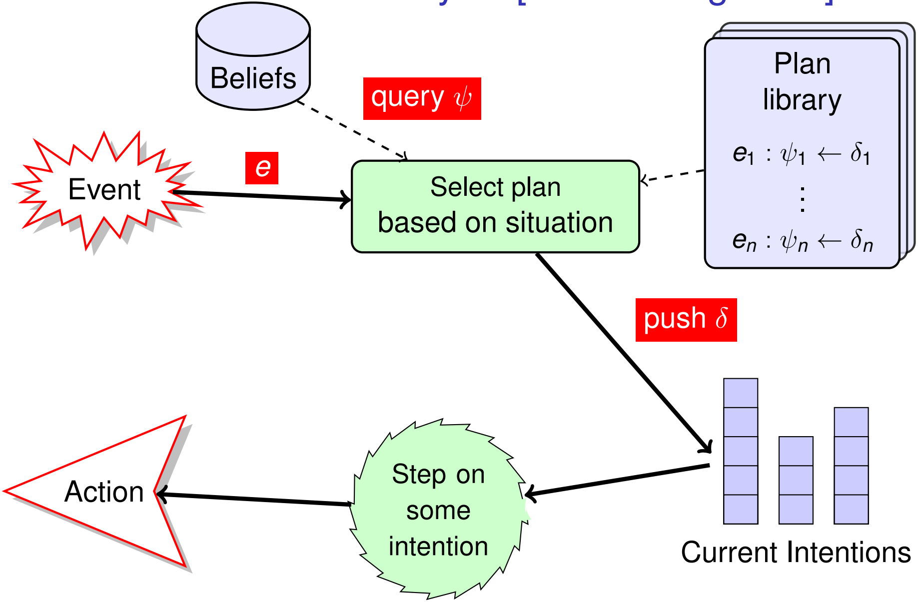
The BDI Execution Cycle [Rao&Georgeff 92]



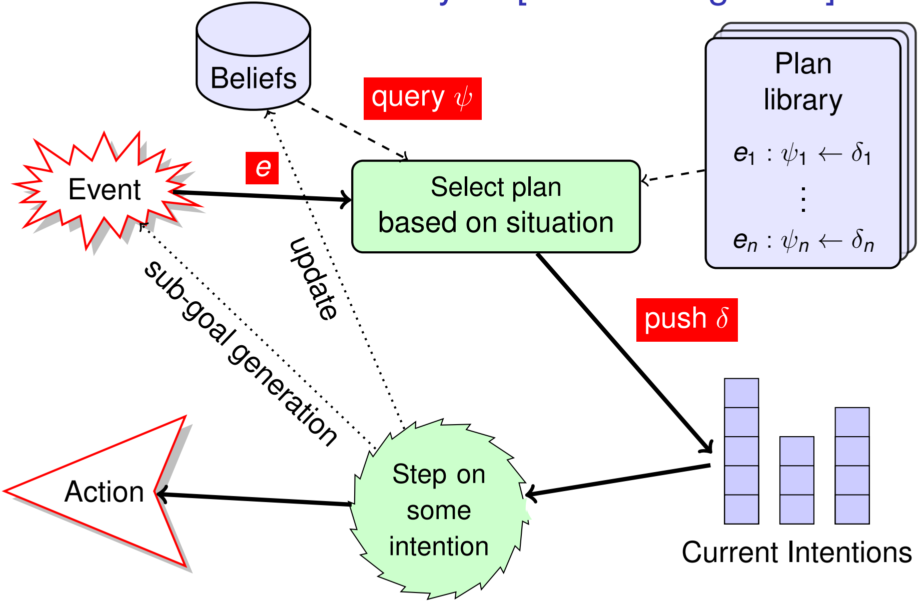
The BDI Execution Cycle [Rao&Georgeff 92]



The BDI Execution Cycle [Rao&Georgeff 92]



The BDI Execution Cycle [Rao&Georgeff 92]



Some BDI Agent-oriented Programming Languages

Some BDI programming language systems/platforms/architectures:

1. PRS & dMars
2. 3APL http://www.cs.uu.nl/3apl/
3. GOAL http://ii.tudelft.nl/trac/goal/
4. 2APL http://apapl.sourceforge.net/
5. JASON http://jason.sourceforge.net/wp/
6. JADEX http://sourceforge.net/projects/jadex/
7. SPARK http://www.ai.sri.com/~spark/
8. JACK http://aosgrp.com/products/jack/
9. SARL http://www.sarl.io/

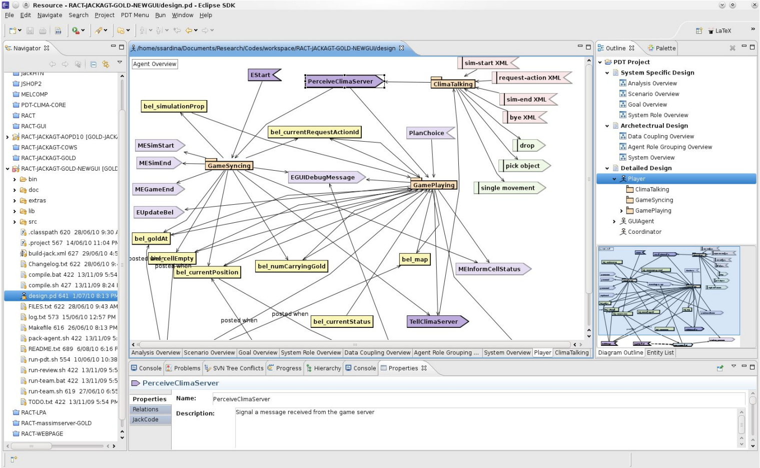
Defining Agents in JACK: `Player.agent`

Base class: `aos.jack.jak.agent.Agent`

```
public agent Player extends Agent {
    #has capability ClimaTalking cap;
    #handles event PerceiveClimaServer;
    #handles event EExecuteCLIMAaction;
    #handles event EAct;
    #posts event EExecuteCLIMAaction ev_executeAction;
    #sends event EInformLoc ev_informLoc;
     ...
    #uses plan MoveRandomly;
    #uses plan PickGold;
    #uses plan HandlePercept;
     ...
    #private data GoldAt bel_goldAt();
    #private data CurrentPosition bel_currPosition();
    #private data NumCarryingGold bel_noCarrGold();
    ....
}
```

Prometheus Design Tool (PDT)

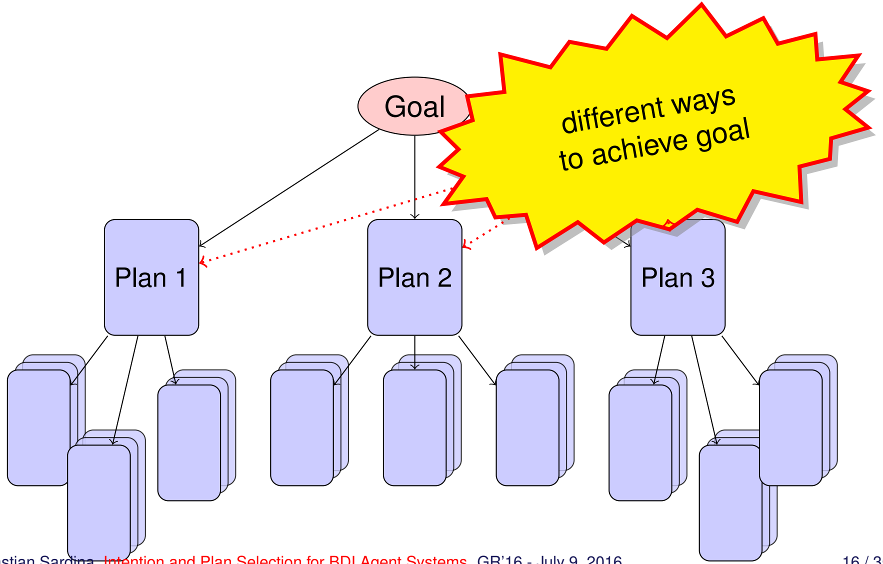
1. **Design:** of an agent system in 3 interrelated phases.
2. **Code generation:** skeleton code in JACK agent language.



www.cs.rmit.edu.au/agents/pdt/

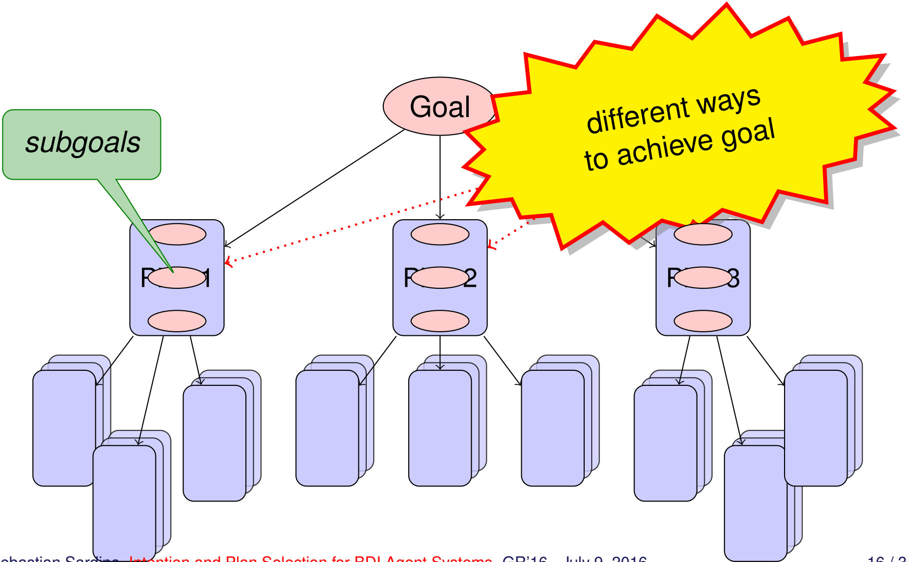
Possibility of Many Options

BDI execution = delayed context subgoal expansion + failure recovery



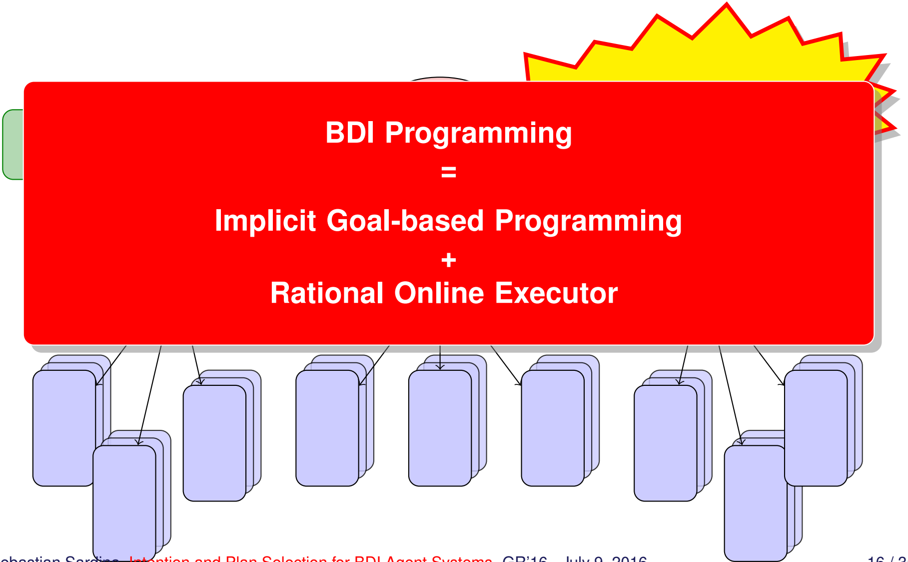
Possibility of Many Options

BDI execution = delayed context subgoal expansion + failure recovery



Possibility of Many Options

BDI execution = delayed context subgoal expansion + failure recovery



From David's talk: *Better future performance*

1. Avoid dead-ends with respect to current goals.
2. Avoid states to jeopardize goal achievement in future.
3. Take actions to maximize actions and goal in the future.

... or something on these lines. :-)

Two Core Deliberation Tasks

Standard Rational Executor/Reasoner
[Rao and Georgeff 1992, Bratman *et al.* 1988]

1. select pending event-goals to handle (*deliberation and filtering*).
2. select a plan to handle goal & commit to it (*means-end reas.*).
3. select intention and execute part of it (*execution*).

Two Core Deliberation Tasks

Standard Rational Executor/Reasoner [Rao and Georgeff 1992, Bratman *et al.* 1988]

2. select a plan to handle goal & commit to it (*means-end reas.*).
3. select intention and execute part of it (*execution*).

Two Core Deliberation Tasks

Standard Rational Executor/Reasoner
[Rao and Georgeff 1992, Bratman *et al.* 1988]

2. select a plan to handle goal & commit to it (*means-end reas.*).
3. select intention and execute part of it (*execution*).

Standard approaches:

- Let the BDI user program both selections
  - meta-reasoning plans, deliberation cycle programming, preferences.
- Select from several built-in schemes.
  - random, top-down, round-robin, FIFO.
- Select based on additional domain information.
  - priorities, deadlines, reward and cost, etc.

Towards Better Plan & Intention Selection

Smarter plan & intention selection under 3 constraints:

- Domain-independent: no extra domain information required.
- No major overhead on BDI executor.
- Easily incorporated into existing BDI platforms.

General approach

Towards Better Plan & Intention Selection

Smarter plan & intention selection under 3 constraints:

- Domain-independent: no extra domain information required.
- No major overhead on BDI executor.
- Easily incorporated into existing BDI platforms.

General approach

Plan Selection: *Prefer plans that have succeeded in similar situations.*

- Learn/improve plans' context conditions.
- Induce decision trees for plans' based on executions.
- Rely on WEKA package.
- [AAMAS'10, JRAS'10, IJCAI'11]

Towards Better Plan & Intention Selection

Smarter plan & intention selection under 3 constraints:

- Domain-independent: no extra domain information required.
- No major overhead on BDI executor.
- Easily incorporated into existing BDI platforms.

General approach

Plan Selection: *Prefer plans that have succeeded in similar situations.*

- Learn/improve plans' context conditions.
- Induce decision trees for plans' based on executions.
- Rely on WEKA package.
- [AAMAS'10, JRAS'10, IJCAI'11]

Intention Selection: *Prefer most "vulnerable" applicable intention.*

- How much *know-how* is available for a goal-event?
- Reason on plan/goal coverage using model counting.
- [AAMAS'12, AAMAS'14, JAAMAS'15]

Towards Better Plan & Intention Selection

Smarter plan & intention selection under 3 constraints:

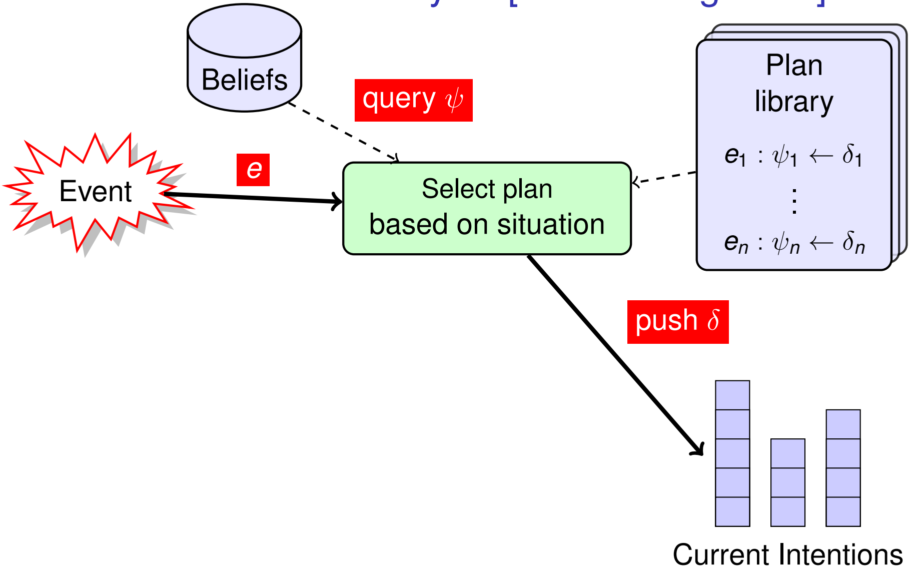
- Domain-independent: no extra domain information required.
- No major overhead on BDI executor.
- Easily incorporated into existing BDI platforms.

General approach

Plan Selection: *Prefer plans that have succeeded in similar situations.*

- Learn/improve plans' context conditions.
- Induce decision trees for plans' based on executions.
- Rely on WEKA package.
- [AAMAS'10, JRAS'10, IJCAI'11]

The BDI Execution Cycle [Rao&Georgeff 92]

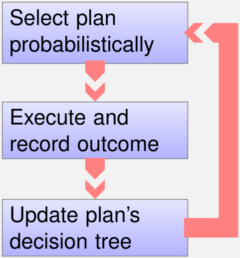


Recap. Plan Selection

- Problem: Context conditions hard to craft & environment changes.

Recap. Plan Selection

- Problem: Context conditions hard to craft & environment changes.
- Approach: Attach decision tress to plans + confidence measure.
  - Easy to incorporate by using learning packages (eg., WEKA)

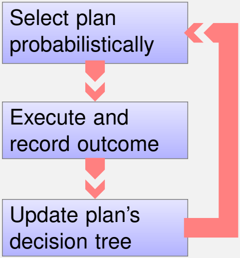


Plan Selection relative to confidence level:

- Plan success rate (as per DT).
- Plan Stability.
- World novelty rate.

Recap. Plan Selection

- Problem: Context conditions hard to craft & environment changes.
- Approach: Attach decision tress to plans + confidence measure.
  - Easy to incorporate by using learning packages (eg., WEKA)

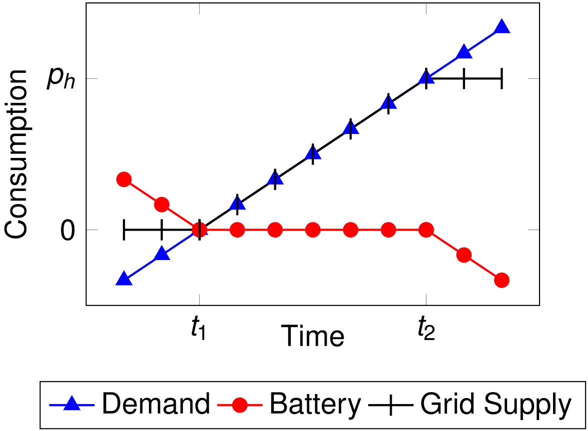


Plan Selection relative to confidence level:

- Plan success rate (as per DT).
- Plan Stability.
- World novelty rate.

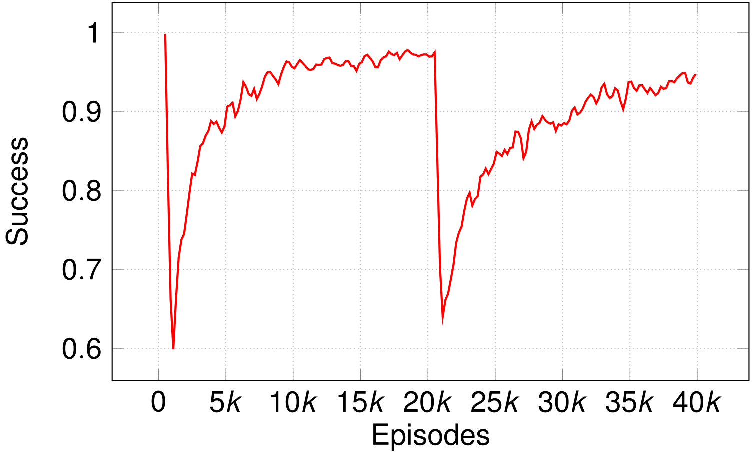
- Empirical Evaluation: agent learns to succeed & adapts
  - Synthetic programs of various shapes [AAMAS'10]
  - Hanoi Tower [JRAS'10]
  - Battery Controller [IJCAI'11]

A Battery Storage Application [IJCAI'11]



Given net building demand, calculate an appropriate battery response in order to maintain grid power consumption within range $[0, p_h]$.

Experiment: Partial Failure with Restoration



Recovery from temporary module failures during $[0, 20k]$, $[20k, 40k]$ episodes.

Towards Better Plan & Intention Selection

Smarter plan & intention selection under 3 constraints:

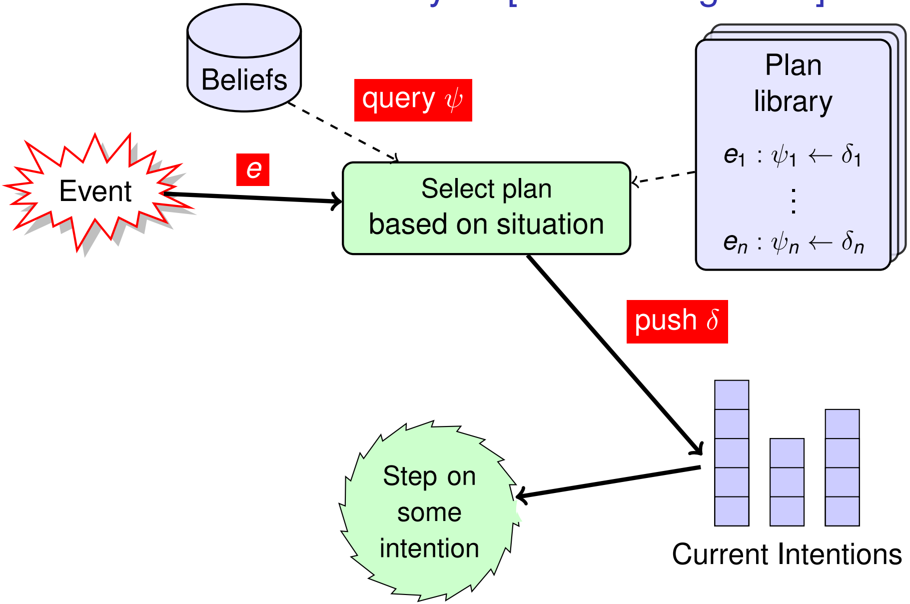
- Domain-independent: no extra domain information required.
- No major overhead on BDI executor.
- Easily incorporated into existing BDI platforms.

General approach

Intention Selection: *Prefer most "vulnerable" applicable intention.*

- How much *know-how* is available for a goal-event?
- Reason on plan/goal coverage using model counting.
- [AAMAS'12, AAMAS'14, JAAMAS'15]

The BDI Execution Cycle [Rao&Georgeff 92]



Intention Selection

How to choose which intention to progress next?

Issues intention interference + dynamic environment + incomplete know-how

Objective maximize successfully executed intentions

Intention Selection

How to choose which intention to progress next?

Issues intention interference + dynamic environment + incomplete know-how

Objective maximize successfully executed intentions

Standard approaches:

Simple first-in-first-out (FIFO) and round-robin (RR)

Meta-level programming deliberation cycle, call-back hooks, etc.

Domain info priorities, deadlines, value, dependencies, etc.

Intention Selection

How to choose which intention to progress next?

Issues intention interference + dynamic environment + incomplete know-how

Objective maximize successfully executed intentions

Standard approaches:

Simple first-in-first-out (FIFO) and round-robin (RR)

Meta-level programming deliberation cycle, call-back hooks, etc.

Domain info priorities, deadlines, value, dependencies, etc.

Challenge: intelligent, domain-independent intention selection

- improves intention success;
- improves focus of attention;
- low over-head;
- easy to implement.

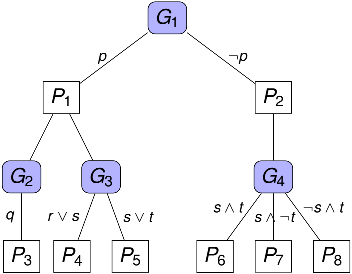
Low Coverage Prioritization [AAMAS'12]

Idea: Opportunistically execute the most "vulnerable" intention

Intentions contain unresolved goals...

How much "know-how" is available?

Less know how, more vulnerable...



Low Coverage Prioritization [AAMAS'12]

Idea: Opportunistically execute the most "vulnerable" intention

Intentions contain unresolved goals...

How much "know-how" is available?

Less know how, more vulnerable...

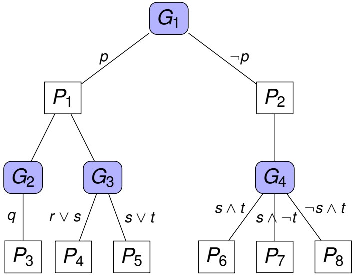
$\Downarrow$

Plan coverage = % states applicable

Goal coverage = % states with app. plans

Aggregated coverage = considers know-how below hierarchy

Lower the coverage, more vulnerable intention



Low Coverage Prioritization [AAMAS'12]

Idea: Opportunistically execute the most "vulnerable" intention

Intentions contain unresolved goals...

How much "know-how" is available?

Less know how, more vulnerable...

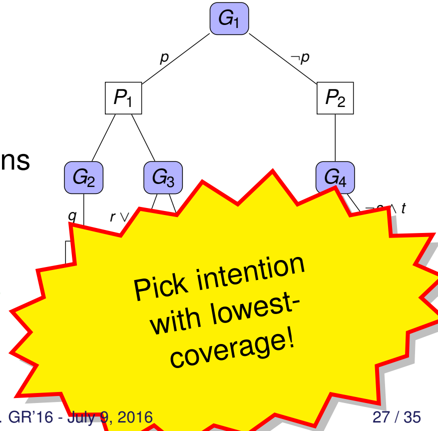
⇓

Plan coverage = % states applicable

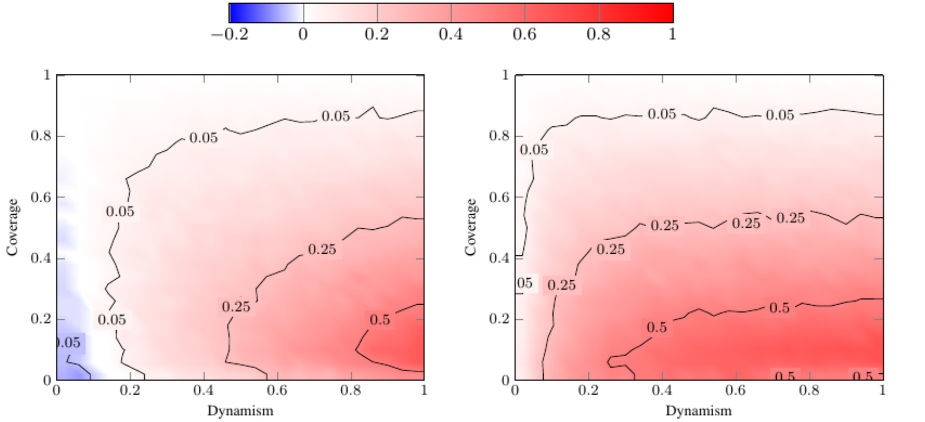
Goal coverage = % states with app. plans

Aggregated coverage = considers know-how below hierarchy

Lower the coverage, more vulnerable intention



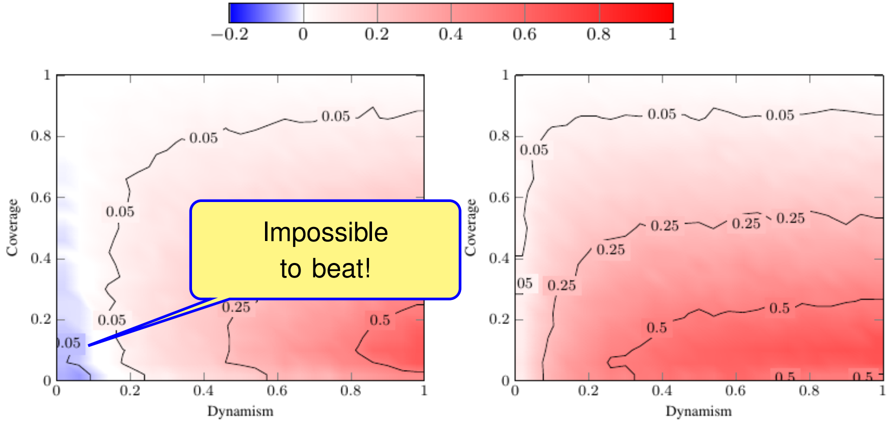
Experimental Results: Impact on Intention Success



(a) $\mathcal{C}$ – FIFO (b) $\mathcal{C}$ – RR

Improves success consistently!

Experimental Results: Impact on Intention Success



(a) $\mathcal{C}$ – FIFO (b) $\mathcal{C}$ – RR

Improves success consistently!

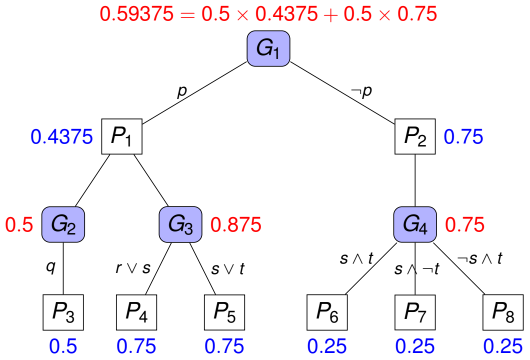
Coverage of Plans and Goals

Coverage of a plan = % of states where plan is applicable

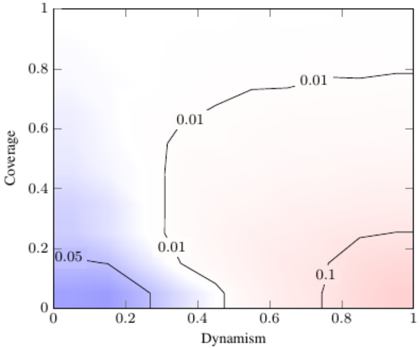
Coverage of a goal = % of states with plans available

Overlap of plans = plans applicable simultaneously

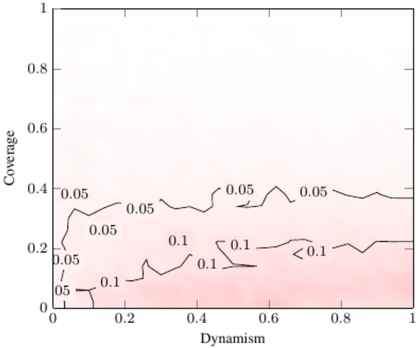
Aggregated coverage = know-how below hierarchy + overlap



Enablement Contribution



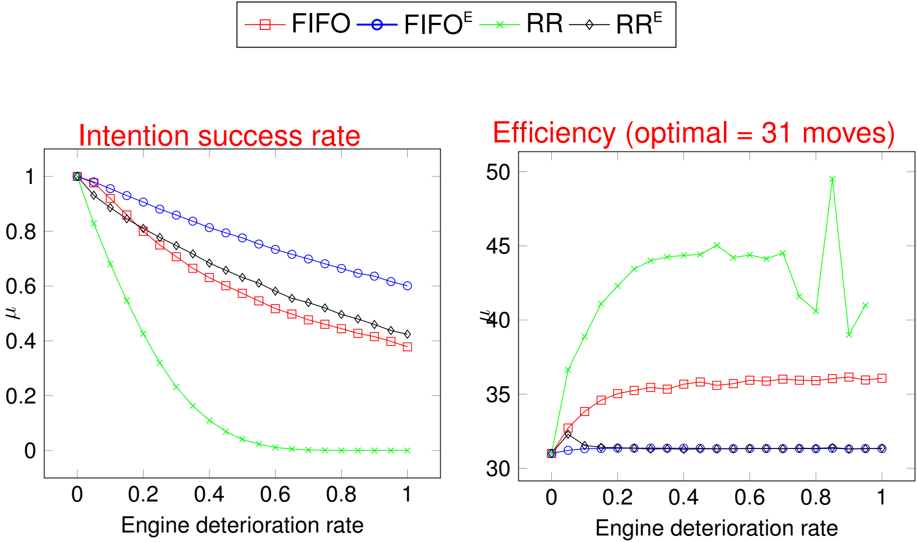
(c) $\mathcal{C}$ – FIFO$^{\mathsf{E}}$



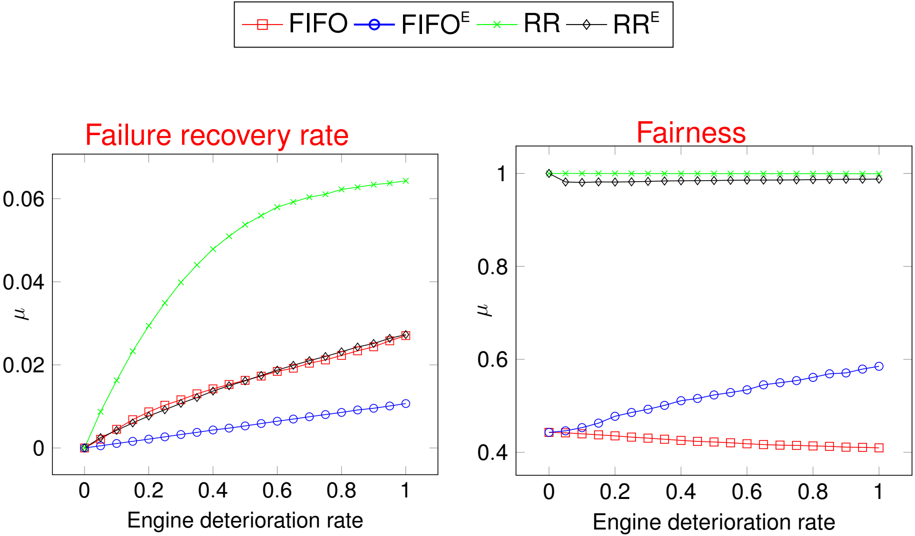
(d) $\mathcal{C}$ – RR$^{\mathsf{E}}$

Major component is goal enablement checking!

Enablement Addition to FIFO and RR on Hanoi



Enablement Addition to FIFO and RR on Hanoi II



Summary

- Intention & Plan selection at the core of BDI “intelligence”
  - ... but almost not addressed (in domain-independent way)!

Summary

- Intention & Plan selection at the core of BDI "intelligence"
  - ... but almost not addressed (in domain-independent way)!
- Proposed learning-based plan selection:
  - Attach a decision tree as a "learnt" context-condition.
  - Use of confidence measure to balance exploit/explore.
  - **Experimental Results:**
    - Converges to optimal
    - Adapts to changes.

Summary

- Intention & Plan selection at the core of BDI "intelligence"
  - ... but almost not addressed (in domain-independent way)!
- Proposed learning-based plan selection:
  - Attach a decision tree as a "learnt" context-condition.
  - Use of confidence measure to balance exploit/explore.
  - **Experimental Results:**
    - Converges to optimal
    - Adapts to changes.

- Proposed low-coverage prioritization:
  - Pick most "know-how vulnerable" intention: via coverage
  - **Experimental Results:**
    - Increases the success rate (almost always).
    - Better in low-coverage + highly dynamic situations.
    - Improves RR significantly with little loss on fairness.
    - Improves efficiency & decreases failure recovery.

Summary

- Intention & Plan selection at the core of BDI "intelligence"
  - ... but almost not addressed (in domain-independent way)!
- Proposed learning-based plan selection:
  - Attach a decision tree as a "learnt" context-condition.
  - Use of confidence measure to balance exploit/explore.
  - **Experimental Results:**
    - Converges to optimal
    - Adapts to changes.

- Proposed low-coverage prioritization:
  - Pick most "know-how vulnerable" intention: via coverage
  - **Experimental Results:**
    - Increases the success rate (almost always).
    - Better in low-coverage + highly dynamic situations.
    - Improves RR significantly with little loss on fairness.
    - Improves efficiency & decreases failure recovery.
- Both approaches domain-independent & implementable.

Challenges?

Need more advanced built-in infrastructure support for goal reasoning:

1. domain-independent;
2. automatic;
3. not based on new programming constructs;
4. based on knowledge representation & learning!

Challenges?

Need more advanced built-in infrastructure support for goal reasoning:

1. domain-independent;
2. automatic;
3. not based on new programming constructs;
4. based on knowledge representation & learning!

Promising challenges:

1. Plan & intention selection: *key places of deliberation!*
2. Goals and plan integration: *representation & reasoning*
3. Goal conflict & synergies.
4. Goal generation/creation: *basic motivations/desires?*
5. Verification & debugging techniques/tools.

Thank you for your attention!

. . . and thanks to those who contributed to this work:

Lin Padgham

John Tangarajah

Dhirendra Singh

Max Waters

References: Agents

Michael E. Bratman, David J. Israel, and Martha E. Pollack.
Plans and resource-bounded practical reasoning.
Computational Intelligence, 4(3):349–355, 1988.

Anand S. Rao and Michael P. Georgeff.
An abstract architecture for rational agents.
In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 438–449, San Mateo, CA, 1992.

Anand S. Rao.
Agentspeak(L): BDI agents speak out in a logical computable language.
In *Proceedings of the European Workshop on Modeling Autonomous Agents in a Multi-Agent World (Agents Breaking Away)*, volume 1038 of *Lecture Notes in Computer Science (LNCS)*, pages 42–55. Springer, 1996.

Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge.
Programming Multi-agent Systems in AgentSpeak Using Jason.
Wiley Series in Agent Technology. Wiley, 2007.

References: Plan Selection

Dhirendra Singh, Sebastian Sardina, and Lin Padgham.
Extending BDI plan selection to incorporate learning from experience.
Journal of Robotics and Autonomous Systems, 58:1067–1075, 2010.

Dhirendra Singh, Sebastian Sardina, Lin Padgham, and Stéphane Airiau.
Learning context conditions for BDI plan selection.
In van der Hoek, Kaminka, Lespérance, Luck, and Sen, editors, *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 325–332, Toronto, Canada, May 2010. IFAAMAS.

Dhirendra Singh, Sebastian Sardina, Lin Padgham, and Geoff James.
Integrating learning into a BDI agent for environments with changing dynamics.
In Craig Knoblock Toby Walsh and Carles Sierra, editors, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2525–2530, Barcelona, Spain, August 2011. AAAI Press.

References: Intention Selection

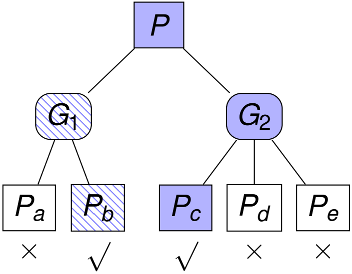
John Thangarajah, Sebastian Sardina, and Lin Padgham.
Measuring plan coverage and overlap for agent reasoning.
In Conitzer, Winikoff, Padgham, and van der Hoek, editors, *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1049–1056, Valencia, Spain, June 2012. IFAAMAS.

Max Waters, Lin Padgham, and Sebastian Sardina.
Evaluating coverage based intention selection.
In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 957–964, Paris, France, May 2014. IFAAMAS.
Nominated for Jodi Best Student Paper award.

Max Waters, Lin Padgham, and Sebastian Sardina.
Improving domain-independent intention selection in BDI systems.
Autonomous Agents and Multi-Agent Systems, 29(4):683–717, 2015.

Raj Jain, Dah-Ming Chiu, and William R Hawe.
A quantitative measure of fairness and discrimination for resource allocation in shared computer system.
Eastern Research Laboratory, Digital Equipment Corporation, 1984.

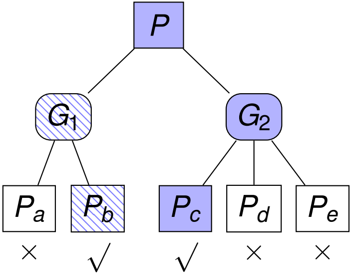
A Dynamic Confidence Measure



Observe & record on averaging window *n*:

- rate of plan success.
- plan local stability: success rate $> \epsilon$?
- plan global stability: ratio of stable plans below in the goal-tree.
- rate of new worlds seen.

A Dynamic Confidence Measure



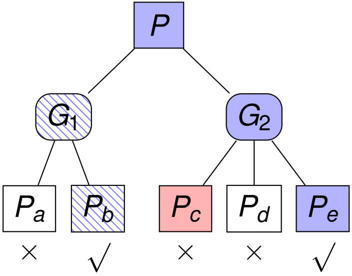
Observe & record on averaging window n:

- rate of plan success.
- plan local stability: success rate $> \epsilon$?
- plan global stability: ratio of stable plans below in the goal-tree.
- rate of new worlds seen.

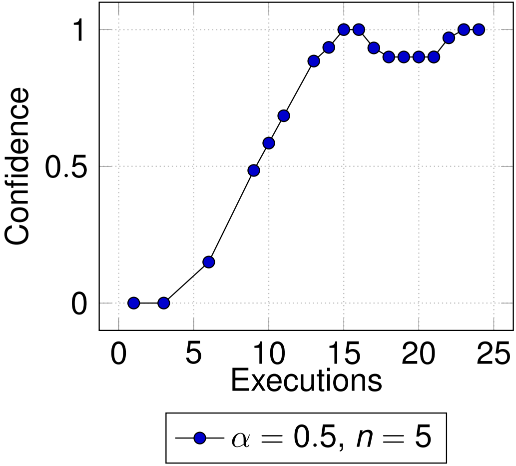
Confidence Measure for Plans

- Stability measure $\mathcal{C}_s(P, w, n)$:
  - how well-informed the last n executions of plan P in world w were?
- World metric $\mathcal{C}_w(P, n)$:
  - how much we have seen the “interesting” worlds for P?
- Aggregated confidence measure $\mathcal{C}(P, w, n)$:
  - $\boxed{\mathcal{C}(P, w, n) = \alpha \mathcal{C}_s(P, w, n) + (1 - \alpha)\mathcal{C}_w(P, n)}$

Example: Dynamic Confidence Measure



After E=15, P_c starts to fail. The confidence drops, promoting new exploration and re-learning.



Plan Selection via Plan Weighting

Given P's confidence measure $\mathcal{C}(P, w, n)$ & DT estimation $\mathcal{P}(P, w)$:

Plan Weight

Using predicted likelihood of success & confidence measure:

$$\Omega(P, w, n) = 0.5 + [\mathcal{C}(P, w, n) \times (\mathcal{P}(P, w) - 0.5)],$$

- When $\mathcal{C}(P, w, n) = 1$, then $\Omega(P, w, n) = \mathcal{P}(P, w)$
- When $\mathcal{C}(P, w, n) = 0$, then $\Omega(P, w, n) = .5$ (default weight)

BDI Plan selection: probabilistically+proportionally to plans' weights.