Verifiable Autonomy — how can you trust your robots?

Michael Fisher (*and Louise Dennis*)

Department of Computer Science and
Centre for Autonomous Systems Technology
University of Liverpool

Part I: *Programming and Requirements*

Overview

- *Autonomy and its Benefits*
  ... what do we mean by "*autonomy*"?
    ... why is it useful?

- *Hybrid Agent Architectures*
  ... architectures for autonomous systems
    ... the increasingly important role of *rational agents*.

- *Programming Rational Agents*
  ... *BDI* programming principles
    ... and simple examples

- *Requirements*
  ... we want to be precise, with *logical* requirements
    ... but which varieties of logic?

What is Autonomy?

> *the ability of a system to make its own decisions and to act on its own, and to do both without direct human intervention.*

Even within this, there are variations concerning decision-making:

Automatic: involves a number of fixed, and prescribed, activities; there may be options, but these are generally fixed in advance.

Adaptive: improves its performance/activity based on feedback from environment — typically developed using tight continuous control and optimisation, e.g. feedback control system.

Autonomous: decisions made based on system's (belief about its) current situation at the time of the decision — environment still taken into account, but internal motivations/beliefs are key.

Who is in Control?

Here is one particular categorisation (PACT) often used in aerospace scenarios:

0: "No Autonomy"
   → *Whole task done by human except for actual operation*
1: "Advice only if requested"
   → *Human asks system to suggest options and human selects*
2: "Advice" → *System suggests options to human*
3: "Advice, and if authorised, action"
   → *System suggests options and proposes one of them*
4: "Action unless revoked"
   - 4a: *System chooses action and performs it if human approves*
   - 4b: *System chooses action and performs it unless human disapproves*
5: "Full Autonomy"
   - 5a: *System chooses action, performs it and informs human*
   - 5b: *System does everything autonomously*

Why Autonomy?

Full autonomy is particularly useful when:

- systems must work in *remote* environments where direct human control is infeasible;
- systems must work in *dangerous* environments where humans cannot be nearby, and so cannot easily assess the possibilities;
- systems need to react *much* more quickly than humans can;
- while not being remote, there are just *too many* autonomous entities for any one human to keep track of;
  or
- it is actually *cheaper* to use autonomous systems!

Autonomy Everywhere!

Many applications both explicit (e.g. robots, driverless cars) and embedded (e.g. smart homes, communication networks).

And often without you realising....

We will briefly look at some of these uses of autonomy

N.B: but we ignore the obvious example of purely software

Example: Aerospace

In remote or dangerous environments it is impractical or costly to have direct human control, so *autonomy* is increasingly being built into the controlling software.

For example, *autonomous choices* are an important element in many *aerospace* applications, such as cooperative

formation flying satellites or *unmanned air vehicles*

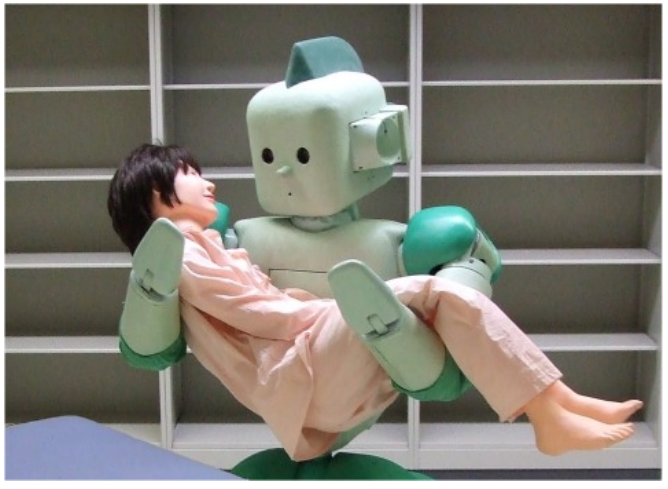
Example: Vehicle Coordination and Convoying

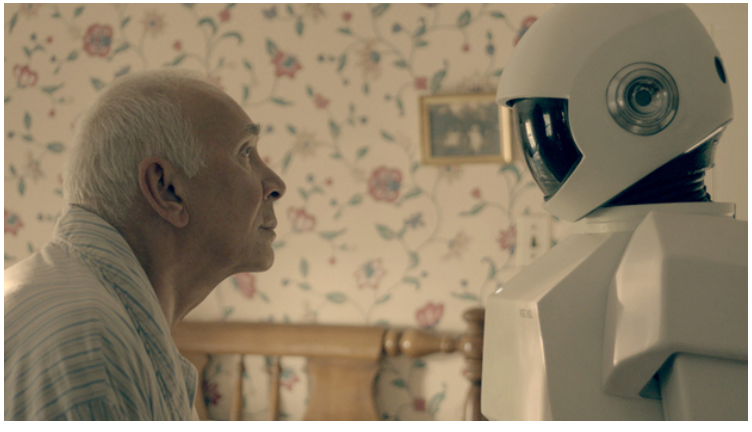
Which vehicles are autonomous?

Example: Robotic Assistants

Robotic Assistants are typically being designed to help the elderly or incapacitated.

RI-MAN, from Bio-mimetic Control Research Center, RIKEN, Japan: `http://rtc.nagoya.riken.jp`





[source: "Robot and Frank"]

[source: Univ. Hertfordshire]

How can we *Trust* a robot?

We want to know: 1. *the robot is safe.*

Now we have autonomy we also want to know that

2. the robot never **intends** *to harm us*

We are going show how we can prove (2) but never (1)!

Instead: 1'. *the robot always tries to be safe.*

Intuition: What do we want?

To be able to assess high-level properties, we want to know

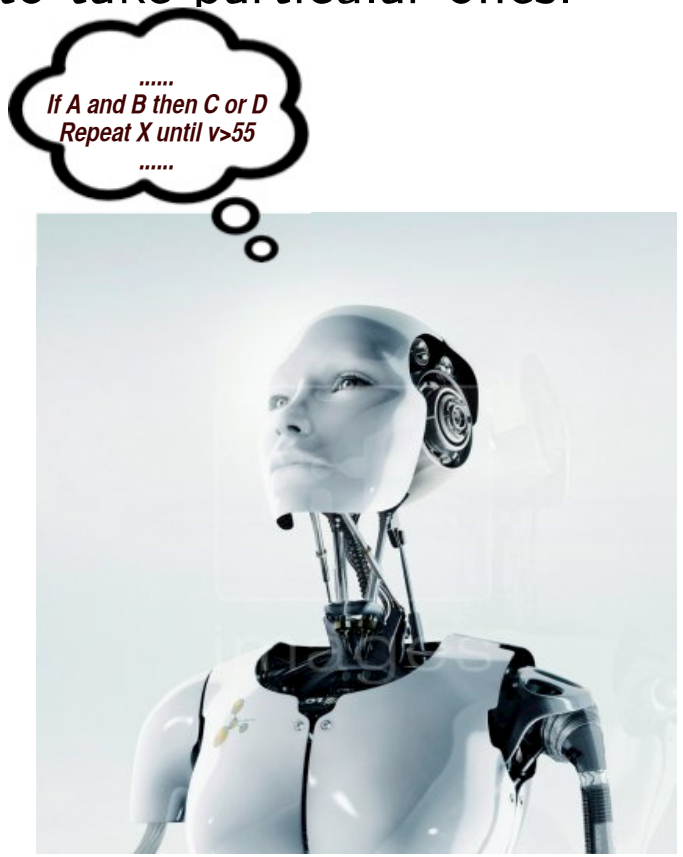
1. *what* a system is "thinking",
2. what *choices* it has, and
3. why it *decides* to take particular ones.

With humans all we can do is ask them questions:

No Psychiatrists for Robots?

But with an *autonomous system* we can (at least in principle) examine its internal programming and find out exactly

1. *what* it is "thinking",
2. what *choices* it has, and
3. why it *decides* to take particular ones.



Autonomous Systems Architectures

Many architectures used, for example

- *sense–plan–act architectures*
  - → originally used in intelligent systems, but often too slow with limited computational resources
- *subsumption architectures*
  - → more efficient, but often *opaque*.
- *hybrid architectures*
  - → distinct discrete control, but the reasons for decisions often very hard to discern.
- *three-layer architectures*
  - → but wide variations, and few common themes

So, recent approach, particularly in autonomous vehicles, is to use *hybrid agent architectures*.

RECAP: What is a Rational Agent?

An *Agent* captures the concept of autonomy, in that it is *able to make its own decisions without human intervention*.

But: this still isn't enough.

Systems controlled by neural networks, genetic algorithms, complex control systems, etc, can act autonomously yet the reasons for their actions are often *inflexible* and *opaque*.

Consequently, the concept of a "Rational Agent" is important.

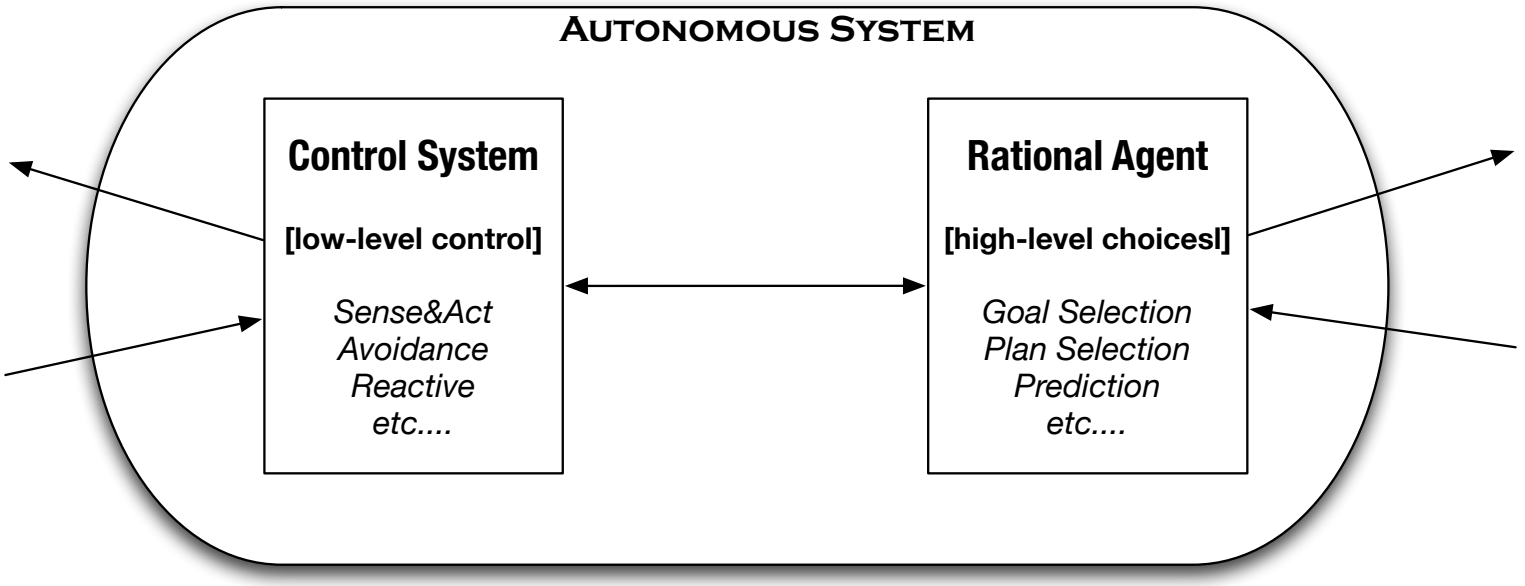
Rational Agent

> *must have explicit reasons for making the choices it does, and should be able to explain these if necessary*

Such agents are often programmed and analyzed by describing their *goals* and *knowledge/beliefs*, and how these change over time.

Hybrid Agent Architectures (1)

So, requirement for *reasoned* decisions and explanations has led on to *hybrid agent architectures*:



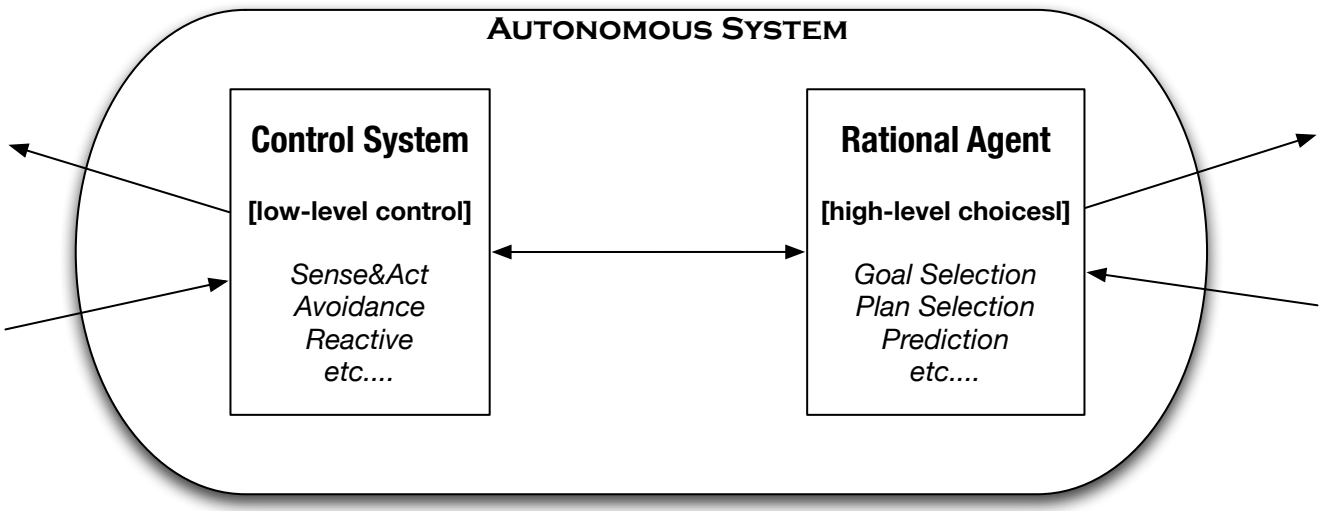
Importantly: Rational agents can adapt their autonomous behaviour to cater for the dynamic aspects of their environment, requirements, goals and knowledge.

Hybrid Agent Architectures (2)

Autonomous systems based on the hybrid combination of

1. *rational agent* for high-level autonomous decisions, and
2. traditional *control systems* for low-level activities,

have been shown to be easier to *understand*, *program*, *maintain* and, often, much more *flexible*.



We are developing rational agent languages for programming such hybrid autonomous systems; particularly in aerospace and robotics.

Hybrid Agent Architectures (3)

The *rational agent* has access to numerous components, including

- control systems,
- sensors,
- planners,
- learning systems, etc.

The agent might also have a description of how all these link together, and even have 'awareness' of how well they are functioning.

Intuition: from Pilot to Rational Agent

Autopilot can essentially fly an aircraft

- keeping on a particular path,
- keeping flight level/steady under environmental conditions,
- planning route around obstacles, etc.

Human pilot makes high-level decisions, such as

- where to go to,
- when to change route,
- what to do in an emergency, etc.

Rational Agent now makes the decisions the pilot used to make.

Example: Spacecraft Landing

Imagine a rational agent controlling a spacecraft that is attempting to land on a planet. The agent has:

control of dynamic activity

$\Rightarrow$ e.g., thrust, direction, etc;

information (i.e. 'knowledge'/'belief')

$\Rightarrow$ e.g., about the planet terrain and target landing sites;

motivations (i.e. 'goals')

$\Rightarrow$ e.g., to land soon, and to remain aloft until safe to land.

The rational agent must dynamically

- assess, and possibly revise, the information held
- generate new motivations or revise current ones
- decide what to do, i.e. *deliberate* over motivations/information

Logic Programming and Prolog

- Logic Programming grew out of attempts to automate formal human reasoning.
- It represents programs as a set of Horn Clauses:

$$T_1 \wedge T_2 \wedge \ldots \wedge T_n \rightarrow H$$

$$Z_1 \rightarrow T_1 \quad \ldots \quad Z_n \rightarrow T_n$$

$$Z_1 \quad \ldots \quad Z_n$$

```
h :- t1, t2, ..., tn.
t1 :- z1.
...
z1.
...
```

Declarative Programming

- Implementations of logic programming have built in search capabilities, allowing a programmer to focus on capturing the information describing the problem, rather than programming the search for the solution.
- It also allows the use of variables, representing the logical concept of "for all", and will instantiate the variables in a solution using *unification*.

$$\forall x.\, Q(x) \rightarrow P(x)$$

```
p(X) :- h(X).
p(X) :- q(X).
q(a).

X = a
```

Logic Programming for Agents: Events, Beliefs, Goals

- *Rational* agent programming languages seek to extend logic programming with the concepts of beliefs and goals and, in some cases, events.
- They are based on the Beliefs, Desires and Intentions paradigm.
- The extensions vary in many ways though there are some features that are common to many including:
  - Use of traditional Prolog to reason about beliefs (and goals),
  - Limiting search capabilities (especially where there are actions),
  - Using the head of the horn clause to react to events,
  - Using *guards* to restrict the applicability of clauses.
- In general the horn clauses of logic programming become *plans* in BDI agent programming.
- In most of these languages an agent can maintain several *intentions* and be working on separate plans in each intention.

Beliefs and Goals

- Nearly all rational agent language have concepts of beliefs and goals.
- Beliefs are propositions that the agent believes to be true and are stored in a *belief base*.
- Goals are states of the world the agent wants to bring about and are stored in a *goal base*.
- Beliefs and Goals may be added in the bodies of plans, and checked for in guards on plans.
- Beliefs may also be added because of perception.

Actions and Capabilities

- Actions are things the agent can do. In most cases we assume they have some effect on the external environment which the agent may need to check for.
- In languages which use them, actions are executed in from the bodies of plans.
- Capabilities are a way of capturing the behaviour of actions in a more formal fashion.
- Capabilities state the pre- and post-conditions for taking an action.
- In some languages they capture part of how the environment is modelled.

Plans

- Plans often look alot like *guarded horn clauses*. They provide a logical guard, expressed in terms of beliefs (and goals) and then provide (a list of) deeds to be done when the guard is true.
- Deeds can include actions, capabilities, belief and goal updates.
- In many languages plans are triggered by events.

The Reasoning Cycle

- Much as Prolog was driven by a built-in depth-first search mechanism, agent languages have a built-in *reasoning cycle*.
- The reasoning cycle controls when events are reacted to, plans are selected, executed and so on.
- Where an agent has an external environment the reasoning cycle also controls when perceptions and messages arrive.

The Environment

- Unlike many programming languages, by default we assume agent programs execute within some external environment which may also be a software entity.
- Researchers are beginning to look at languages for modelling these environments, particularly languages for describing the organisation of groups of agents.
- In this tutorial we assume that the environment is a software simulation of the real world written in Java.

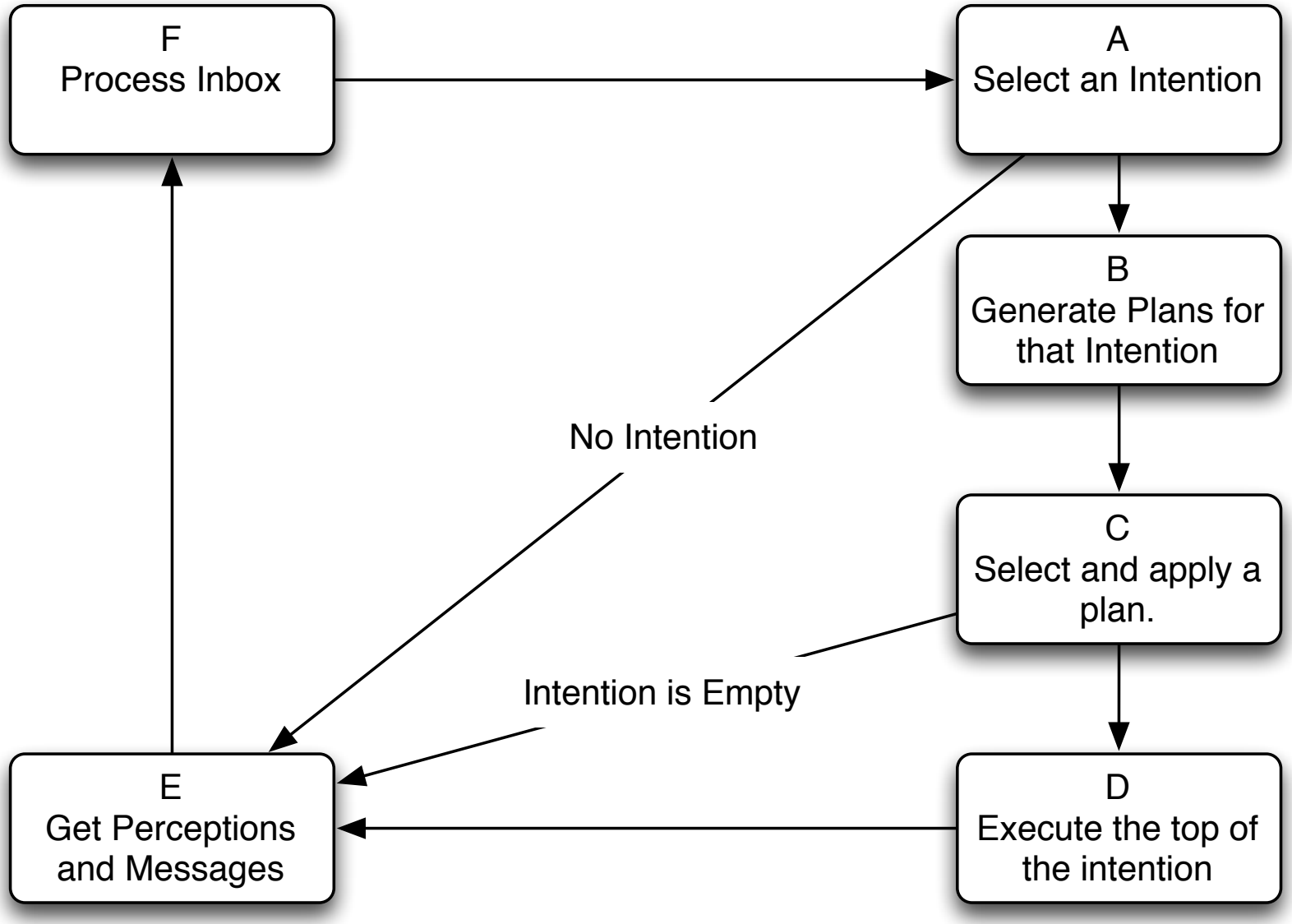
The Gwendolen Language

We give simple examples in the Gwendolen language.

Gwendolen is a BDI language specifically designed to be used with the AJPF model checker.

1. L. A. Dennis and B. Farwer. Gwendolen: A BDI Language for Verifiable Agents.
   In *Proc. AISB Workshop on Logic and the Simulation of Interaction and Reasoning*. AISB, 2008.
2. The MCAPL Source Code. `mcapl.sourceforge.net`.

The Gwendolen Reasoning Cycle



A Simple Example

```
:name: ag1                                                    1
                                                              2
:Initial Beliefs:                                             3
                                                              4
empty                                                         5
                                                              6
:Initial Goals:                                               7
                                                              8
pickup [achieve]                                              9
                                                              10
:Plans:                                                       11
                                                              12
+!pickup [achieve] : {B empty} <- +pickup,                    13
        -empty,                                               14
        print(done);                                          15
```

Robots Collaborating to Find Survivors

```
:Initial Beliefs:                                                    1
square(0, 0), square(0, 1), square(0, 2)                             2
square(1, 0), square(1, 1), square(1, 2)                             3
square(2, 0), square(2, 1), square(2, 2)                             4
                                                                     5
:Belief Rules:                                                       6
B area_empty :- ~((B square(Xc, Yc),                                 7
                   ~(B empty(Xc, Yc))));                             8
B unchecked(Xc, Yc) :- (B square(Xc, Yc),                            9
                        (~(B at(Xc, Yc)),                            10
                        ( ~(B empty(Xc, Yc)),                        11
                          ~(B human(Xc, Yc))))));                    12
                                                                     13
:Initial Goals:                                                      14
leave [achieve]                                                      15
```

Continued

```
+! leave [achieve]:                                                    1
     {~B at(X1, Y1), B unchecked(X, Y)}                                2
     <- +at(X, Y), move_to(X, Y);                                      3
+! leave [achieve]: {B at(X, Y), ~B human,                             4
      ~B area_empty, B unchecked(X1, Y1) } <-                          5
   +empty(X, Y), -at(X, Y),                                            6
   +at(X1, Y1), move_to(X1, Y1);                                       7
+! leave [achieve]: {B at(X, Y), ~ B human,                            8
      ~B area_empty, ~B unchecked(X1, Y1) } <-                         9
   +empty(X, Y), -at(X, Y);                                            10
+! leave [achieve]: {B at(X, Y), B human } <-                          11
   +found;                                                             12
+! leave [achieve]: {B area_empty} <- +leave;                         13
                                                                       14
+found : {B at(X, Y)} <-                                               15
    .send(lifter, :tell, human(X, Y)),                                 16
    +sent(lifter, human(X, Y)), +leave;                                17
```

The Lifting Robot

```
:Plans:                                                          1
+.received(:tell, Msg): {~ B Msg} ←                              2
      +Msg, +rec(msg);                                           3
                                                                 4
+human(X, Y): {⊤} ← +! free(human) [achieve];                    5
+! free(human) [achieve] : { B human(C, Y),                      6
              ~B at(C, Y), ~B have(human)} ←                     7
        move_to(C, Y), +at(C, Y);                                8
+! free(human) [achieve] : { B human(X, Y),                      9
             B at(X, Y), ~B have(human)} ←                       10
        lift(human), +have(human);                               11
+! free(human) [achieve] : {B have(human)} ←                     12
       +free(human);                                             13
```

Demo

```
Jun 12, 2013 5:56:17 PM ajpf.util.AJPFLogger info
INFO: loading property file: /Users/louiseadennis/E
....: Sleeping agent lifter
....: searcher done move_to(0,0)
....: Waking agent lifter
....: searcher done send(1:human(0,0), lifter)
....: searcher done send(1:human(0,0), lifter)
....: Sleeping agent searcher
....: lifter done move_to(0,0)
....: lifter done lift(human)
```

Rational Agent Requirements?

For describing requirements of our rational agent, we should choose an appropriate logic.

One that provides a level of abstraction close to the key concepts of the system. For example:

- dynamic communicating systems $\longrightarrow$ *temporal logics*
- systems managing information $\longrightarrow$ *logics of knowledge*
- autonomous systems $\longrightarrow$ *logics of motivation*
- situated systems $\longrightarrow$ *logics of belief, contextual logics*
- timed systems $\longrightarrow$ *real-time temporal logics*
- uncertain systems $\longrightarrow$ *probabilistic logics*
- cooperative systems $\longrightarrow$ *cooperation/coalition logics*

$\Rightarrow$ In realistic scenarios, we will need to *combine* several logics.

Logical Theories for Rational Agents

Logical theories for rational agents typically consist of

Dynamism: temporal or dynamic logic;

Information: modal/probabilistic logics of belief/knowledge;

Motivation: modal logics of goals, intentions, desires.

Again, this requires *combinations* of logics.

For example, the well known BDI approach comprises

- a (branching) temporal/dynamic logic,
- a KD45 modal logic of belief,
- a KD modal logic of desire, and
- a KD modal logic of intention.

Sample Logical Specification: Assisting Humans

"If a patient is in danger, then the patient believes that there is a probability of 95% that, within 2 minutes, a helper robot will want to assist the patient."

$B^{\geq 0.95}_{patient}$ ..... *patient believes with 95% probability*

$\Diamond^{\leq 2}$ ..... *within 2 minutes*

G_{helper} ..... *helper robot has a goal*

$$in\_danger(patient) \Rightarrow B^{\geq 0.95}_{patient} \Diamond^{\leq 2} G_{helper}\ assist(patient)$$

Sample Logical Specification: Pre-emptive Shopping

"If I believe, with over 75% probability that at some point in the future the shop assistant's goal will be to sell me some shoes, then I intend that within 5 seconds I will leave the shop."

$B_{me}^{>0.75}$ ..... *I believe with* $>$ 75% *probability*

$\Diamond$ ..... *at some point within the future*

$G_{assistant}$ ..... *shop assistant's goal*

I_{me} ..... *I intend*

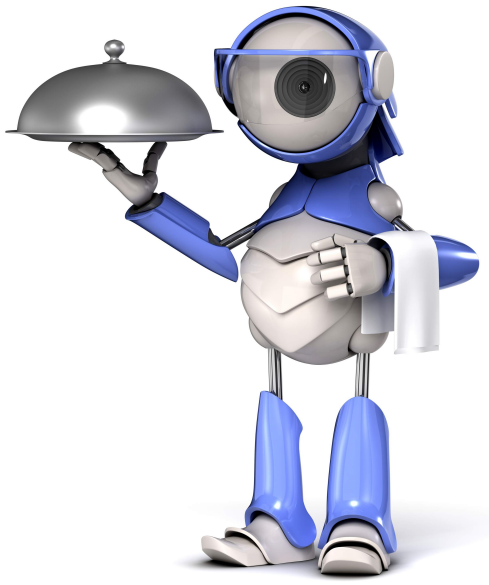
$\Diamond^{<5s}$ ..... *within 5 seconds*

$$B_{me}^{>0.75} \Diamond G_{assistant}\, sell\_shoes(me) \;\Rightarrow\; I_{me} \Diamond^{<5s} leave\_shop(me)$$

Typical Requirement [Assistive Systems]

To be more specific about requirements that we have concerning human-robot interaction, here is a typical statement we must *verify*:

if a robot "believes" that a human is directly in front of it, and it has a goal to get to a room beyond the human, then it should never deliberately choose a plan that brings it into close proximity with the human, unless there is no alternative, in which case the robot may even decide to wait for the person to move or instead may decide to drop or revise its goal.



Selected/Related References

- R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming Multi-agent Systems in AgentSpeak Using Jason*. Wiley, 2007.
- Dennis, L. A., Fisher, M., Aitken, J., Veres, S. M., Gao, Y., Shaukat, A., and Burroughes, G. Reconfigurable Autonomy. *KI — Kunstliche Intelligenz*, 2014
- Dennis, L. A., Fisher, M., and Webster, M. Verifying Autonomous Systems. ACM Communications, 2013
- Dennis, L. A., Fisher, M., Lisitsa, A., Lincoln, N., and Veres, S. M. Satellite Control Using Rational Agent Programming. IEEE Intelligent Systems 25(3):92-97, 2010.
- Dix, J. and Fisher, M. Where Logic and Agents Meet. Annals of Mathematics and Artificial Intelligence 61(1):15-28. Springer, 2011.
- S. Franklin and A. Graesser. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *LNCS 1193*, pp 21–35. 1996.
- Lincoln, N., Veres, S. M., Dennis, L. A., Fisher, M., and Lisitsa, A. Autonomous Asteroid Exploration by Rational Agents. *IEEE Computational Intelligence 8(4):25-38*, 2013.
- A. S. Rao. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In *LNCS 1038*, pp 42–55. Springer, 1996.
- A. S. Rao and M. Georgeff. BDI Agents: from Theory to Practice. In *Proc. 1st Int. Conf. Multi-Agent Systems (ICMAS)*, pp 312–319, San Francisco, USA, 1995.
- A. S. Rao and M. P. Georgeff. Modeling Agents within a BDI-Architecture. In *Proc. 2nd Int. Conf. Principles of Knowledge Representation and Reasoning (KR)*, pp 473–484. Morgan Kaufmann, 1991.
- A. S. Rao and M. P. Georgeff. An Abstract Architecture for Rational Agents. In *Proc. 3rd Int. Conf. Principles of Knowledge Representation and Reasoning (KR)*, pp 439–449. Morgan Kaufmann, 1992.
- P. Tabdada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.
- A. Tiwari. Abstractions for Hybrid Systems. *Formal Methods in Systems Design*, 32:57–83, 2008.
- M. Wooldridge and A. Rao, eds, *Foundations of Rational Agency*. Kluwer, 1999.