Agents’ architectures

Autonomous Software Agents

A.A. 2022-2023

Prof. Paolo Giorgini

Dr. Marco Robol

UNIVERSITY OF TRENTO - Italy

Department of Information and Communication Technology

Internal and Social Architectures

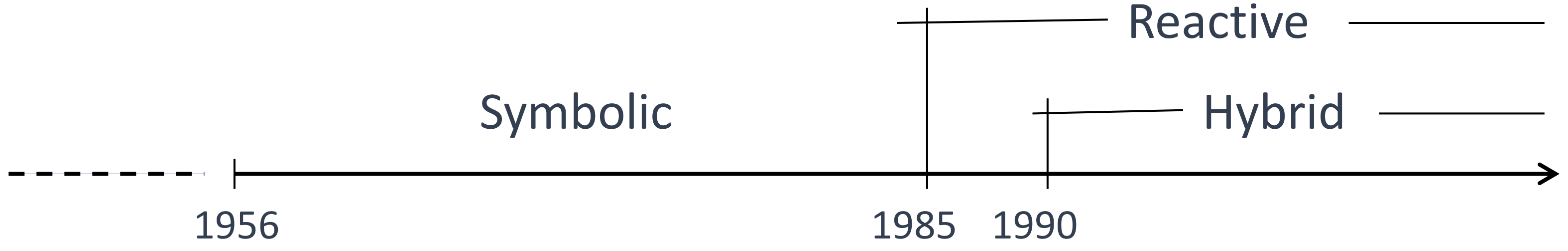
- The internal agent architecture determines the kinds of components an agent consists of and additionally define how these components interact
    - implement a reasoning process that ultimately leads to agent actions
- The social agent architectures have been devised for describing group structures and behavior
    - provide concepts on an organizational level, which allow the description of structures and behavior similar to how work is organized within human organizations

Internal Agent architectures

- Fundamental mechanism underlying the autonomous components that support behaviour in real-world, dynamic and open environments
  - Logic-based (symbolic)
  - Reactive
  - Layered (hybrid)
  - BDI (deliberative)
- Most used
  - Reactive -- hybrid -- deliberative

Agent Architectures

- Originally (1956-1985), pretty much all agents designed within AI were symbolic reasoning agents
- Its purest expression proposes that agents use explicit logical reasoning in order to decide what to do
- Problems with symbolic reasoning led to a reaction against this — the so-called reactive agents' movement, 1985–present
- From 1990-present, a number of alternatives proposed: hybrid architectures, which attempt to combine the best of reasoning and reactive architectures



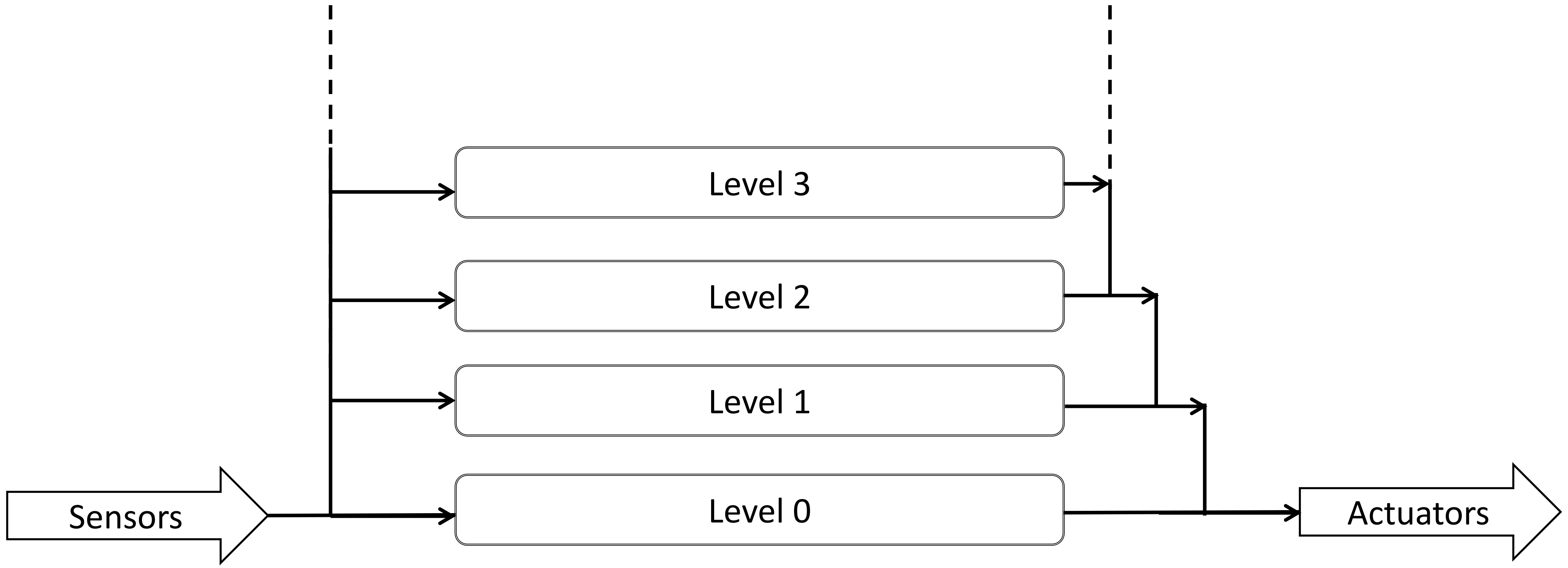
Reactive architectures

- Underpin the importance of fast reactions to changes within highly dynamic environments
  - In specific application domains fast reactions outweigh correct behavior, which is generated too slowly and might be no longer applicable
  - Reactive agents do not possess a symbolic representation of the world and build their decisions on the received percepts

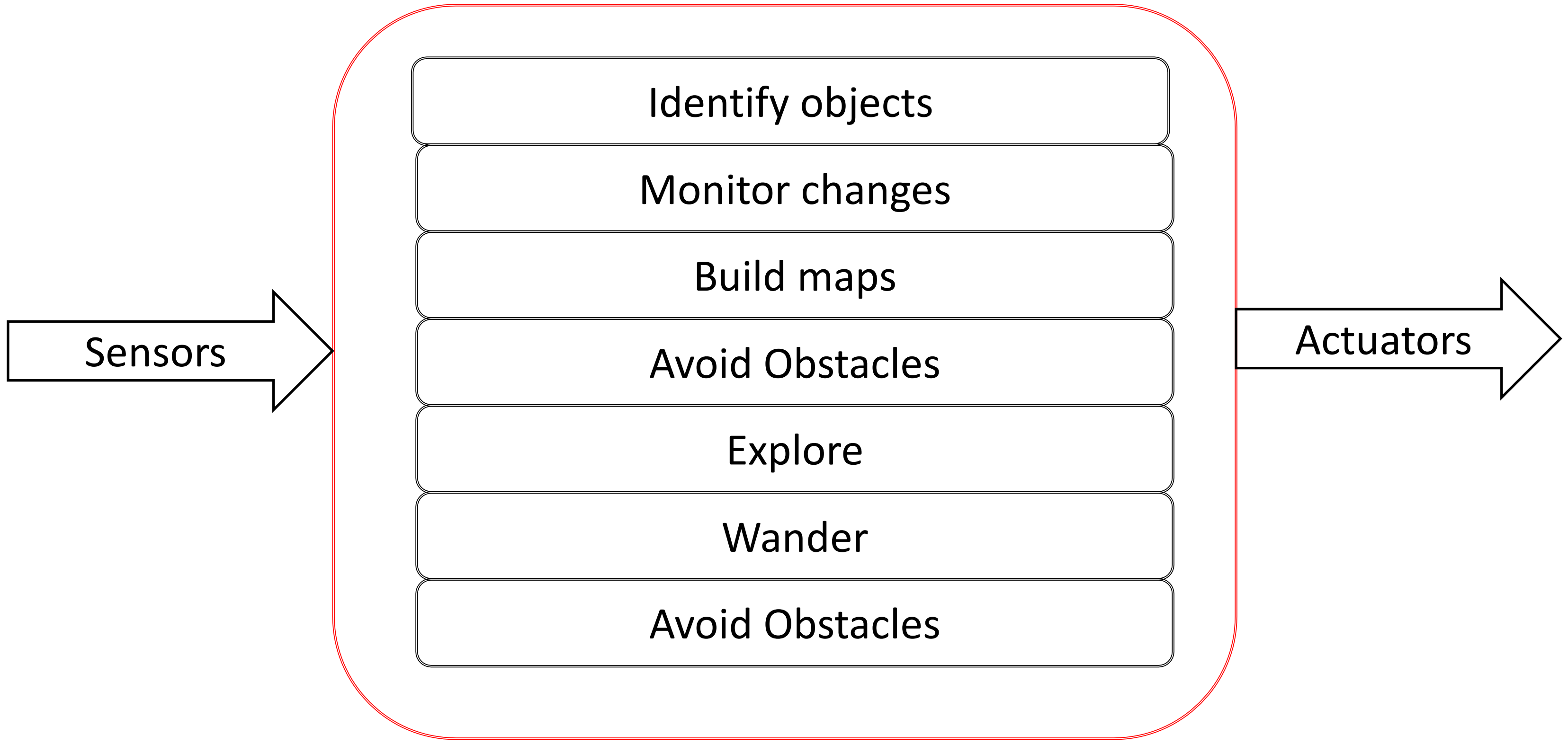
Reactive Architecture - (Brooks, 1991)

- Brooks has put forward three theses:
  - Intelligent behavior can be generated without explicit representations
  - Intelligent behavior can be generated without explicit abstract reasoning
  - Intelligence is an emergent property of certain complex systems
- Subsumption architecture
  - Hierarchy of task-accomplishing behaviors
  - Each behavior is a rather simple rule-like structure
  - Each behavior ‘competes’ with others to exercise control
  - Lower layers represent more primitive kinds of behavior (such as avoiding obstacles), and have precedence over layers further up the hierarchy
  - The resulting systems are extremely simple

Reactive Architecture – subsumption



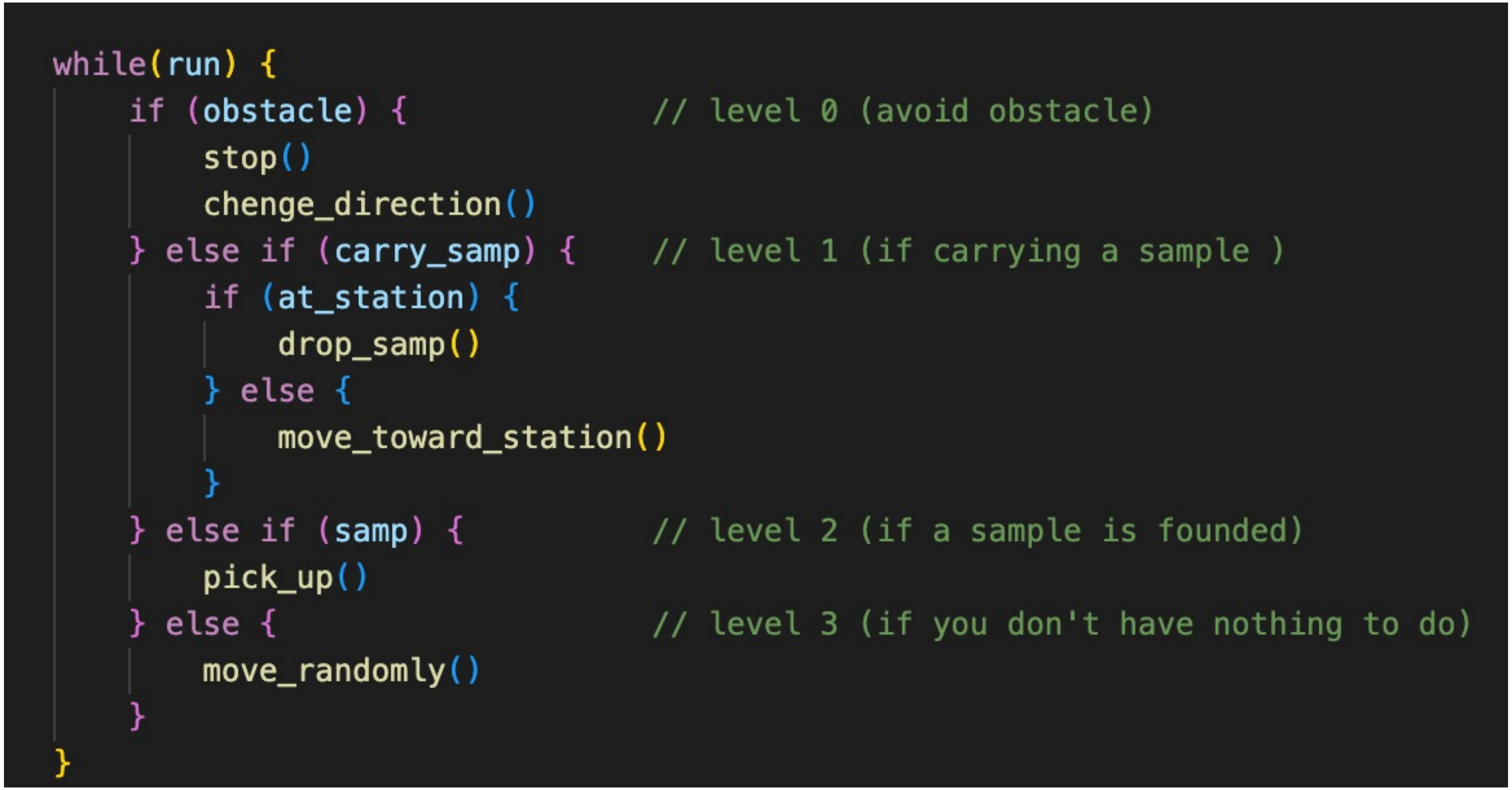
Reactive Architecture (1)



From Brooks, “A Robust Layered Control System for a Mobile Robot”, 1985

Explorer Rules

- The lowest-level behavior is obstacle avoidance:
  - (1) **if detect an obstacle then change direction**
- Any samples carried by agents are dropped back at the mother-ship:
  - (2) **if carrying samples and at the base then drop samples**
- Agents carrying samples will return to the mother-ship:
  - (3) **if carrying samples and not at the base then travel up gradient**
- Agents will collect samples they find:
  - (4) **if detect a sample then pick sample up**
- An agent with "nothing better to do" will explore randomly:
  - (5) **if true then move randomly**



```
while(run) {
    if (obstacle) {                     // level 0 (avoid obstacle)
        stop()
        chenge_direction()
    } else if (carry_samp) {            // level 1 (if carrying a sample )
        if (at_station) {
            drop_samp()
        } else {
            move_toward_station()
        }
    } else if (samp) {                  // level 2 (if a sample is founded)
        pick_up()
    } else {                            // level 3 (if you don't have nothing to do)
        move_randomly()
    }
}
```

What about with a multithread implementation?

Advantages and Limitations of RAs

- Advantages
  - Simplicity, Economy, Computational tractability, Robustness against failure
- Limitations
  - Agents without environment models must have sufficient information available from local environment
  - If decisions are based on local environment, how does it take into account non-local information - i.e., it has a "short-term" view
  - How to make reactive agents that learn ?
  - Since behavior emerges from component interactions plus environment, it is hard to see how to engineer specific agents (no principled methodology exists)
  - It is hard to engineer agents with large numbers of behaviors (dynamics of interactions become too complex to understand)

Logic-based architectures

- Emphasize a symbol-based reasoning process, which requires an agent to posses a local worldview
- Intelligent behaviors can be generated by a symbolic representation of the environment and syntactically manipulating this representation
  - Logical formulae (symbolic representation)
  - Logical deduction or theorem proving (syntactic manipulation)
- A theory can explain how the agent should behave
  - how goals are generated so as to satisfy its delegated objectives
  - how agent interleaves goal-directed and reactive behavior in order to achieve these goals
  - The theory (specification) is refined through a series of progressively more concrete stages until finally an implementation is reached

Symbolic Reasoning Agents

- The classical approach to building agents is to view them as a particular type of knowledge-based system
- This paradigm is known as symbolic AI
- We define a deliberative agent to be one that:
  - contains an explicitly represented, symbolic model of the world
  - makes decisions (for example about what actions to perform) via symbolic reasoning

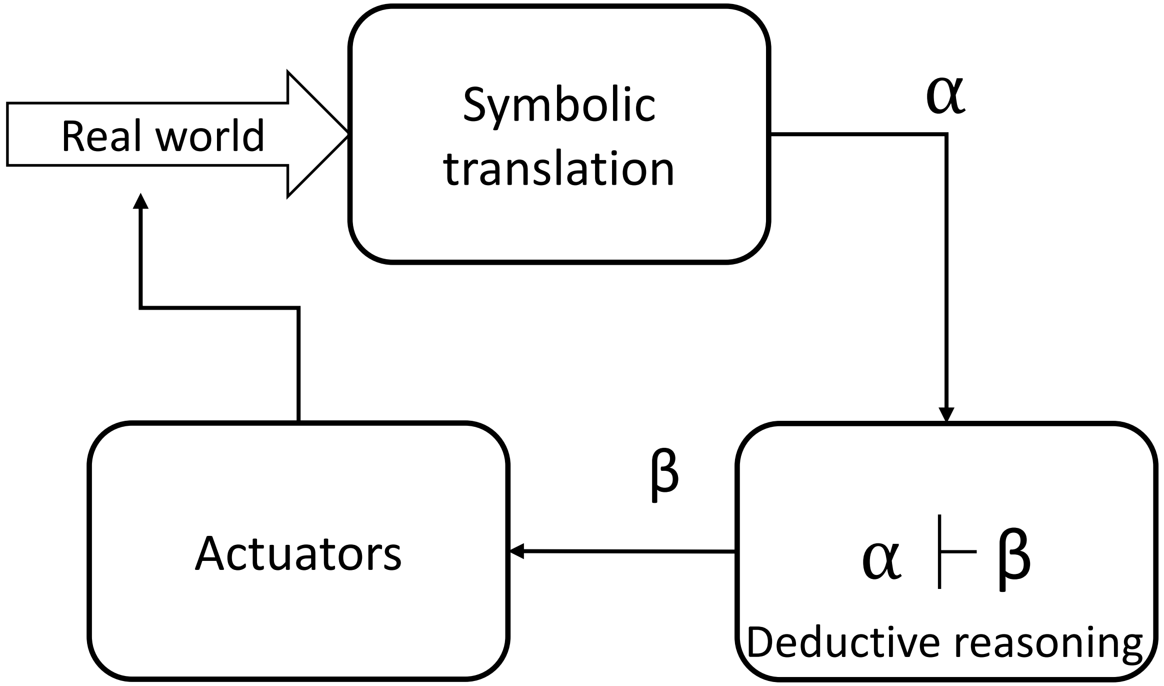
Symbolic Reasoning Agents

To build an agent in this way, there are two key problems:

- The transduction problem
  - real world into an accurate, adequate symbolic description
  - in time for that description to be useful…vision, speech understanding, learning
- The representation/reasoning problem:
  - symbolically represent information about complex real-world entities and processes, and how to get agents to reason with this information in time for the results to be useful…knowledge representation, automated reasoning, automatic planning

Underlying problem lies with the complexity of symbol manipulation algorithms

- many (most) search-based symbol manipulation algorithms of interest are highly intractable


Real world
Symbolic translation
α
β
α ⊢ β
Deductive reasoning
Actuators

Deductive Reasoning Agents

- How can an agent decide what to do?
- Basic idea is to use logic to encode a theory stating the *best* action to perform in any given situation
- Let:
  - ρ be this theory (typically a set of rules)
  - Δ be a logical database that describes the current state of the world
  - Ac be the set of actions the agent can perform
  - $\Delta \vdash_{\rho} \phi$ means that ϕ can be proved from Δ using ρ

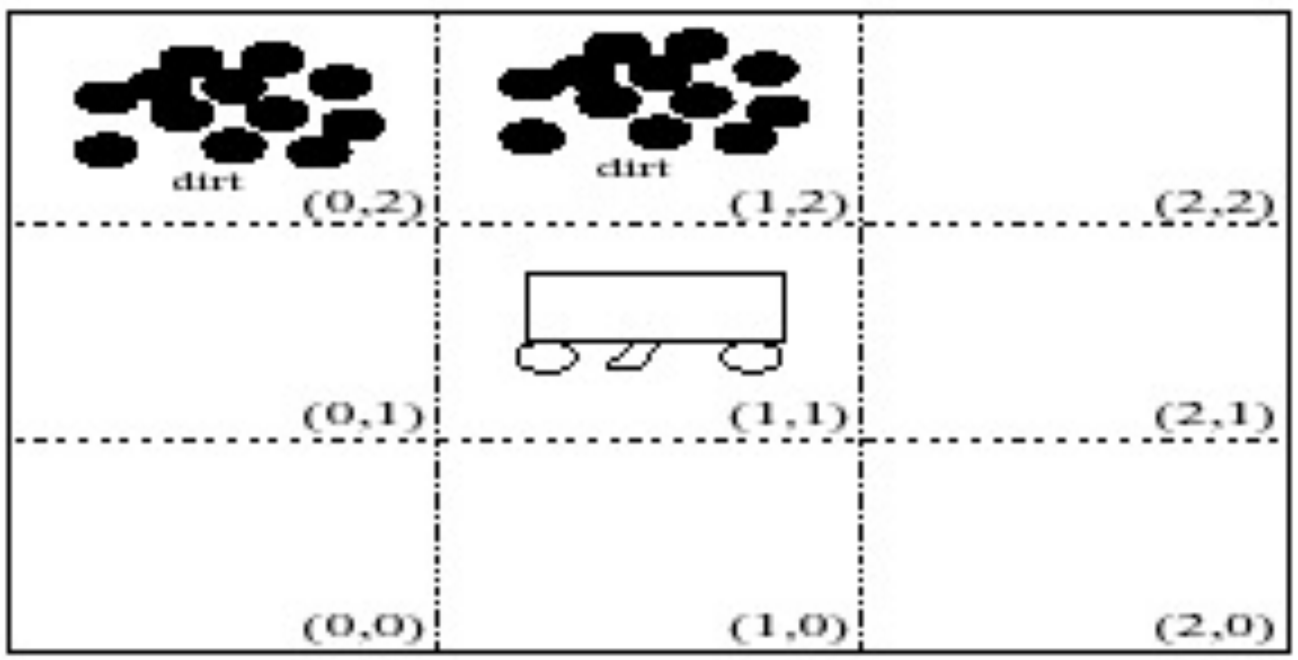
Deductive Reasoning Agents: action selection

```
/* try to find an action explicitly prescribed */
for each $a \in Ac$ do
        if $\Delta \vdash_{\rho} Do(a)$ then
                return $a$
        end-if
end-for

/* try to find an action not excluded */
for each $a \in Ac$ do
        if $\Delta \nvdash_{\rho} \neg Do(a)$ then
                return $a$
        end-if
end-for
return $null$ /* no action found */
```

Deductive Reasoning Agents

- An example: The Vacuum World
  - Goal is for the robot to clear up all dirt



Deductive Reasoning Agents

- Use 3 *domain predicates* to solve problem:

$In(x, y)$	agent is at (x, y)
$Dirt(x, y)$	there is dirt at (x, y)
$Facing(d)$	the agent is facing direction d

- Possible actions:

$$Ac = \{turn, forward, suck\}$$

… *turn* means “turn right”

Deductive Reasoning Agents

- Rules ρ that govern the agent's behavior:
  - φ(…) -> ψ(…)
  - φ and ψ are predicates over some arbitrary list of constants and variables
  - If φ matches against the agent's database, then ψ can be concluded, with any variables in ψ instantiated
- Rules
  - Cleaning action rule: `In(x,y)^Dirt(x,y)->Do(suck)`
  - This action will take priority over all other possible behaviors of the agent (such as navigation)
  - In case the rule does not match against the database, the basic action of the agent will be to traverse the world

Deductive Reasoning Agents

In an agent’s database we have facts like

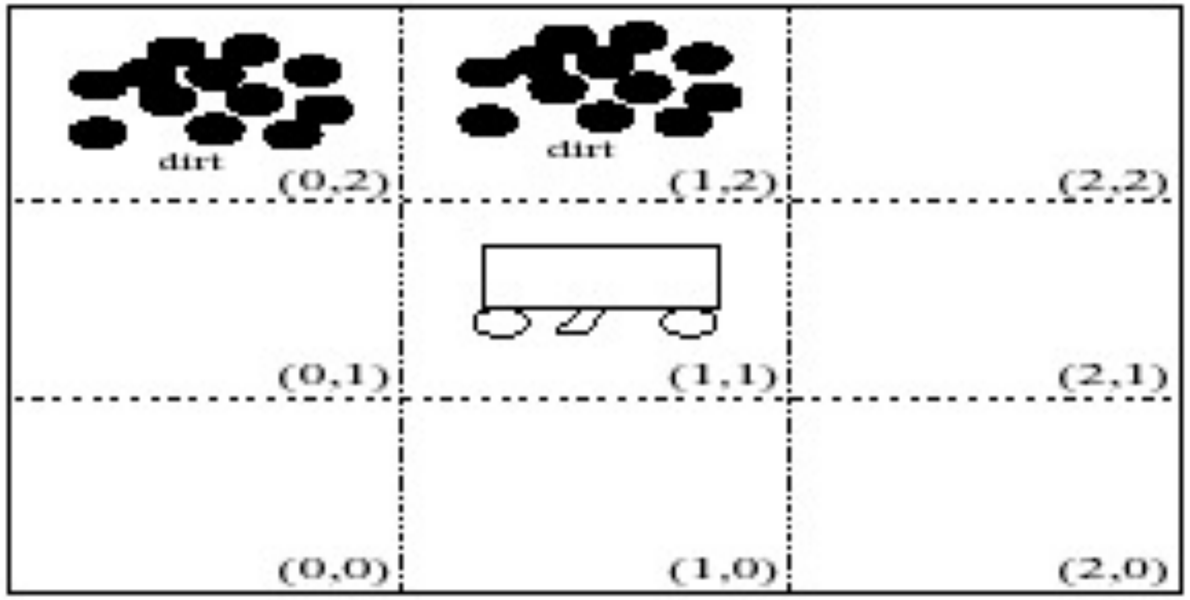
```
In(1,1) Dirt(0,2) Dirt(1,2) Facing(West)
```

while formulas like

```
In(x,y) ∧ Dirt(x,y) → Do(suck)
```

are part of the reasoning framework and not in the database (compare: program vs. interpreter)

Example



Sates

- `In(x,y)`
- `Facing(d): d={n,s,e,w}`
- `Dirt(x,y)`

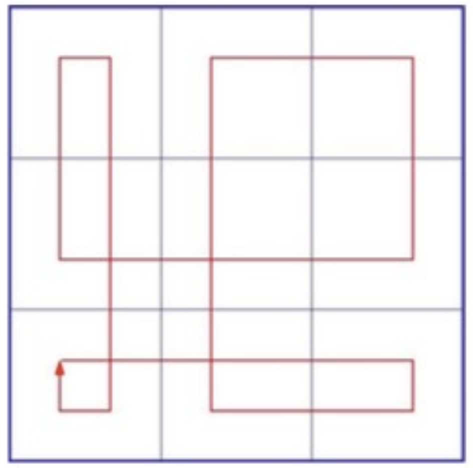
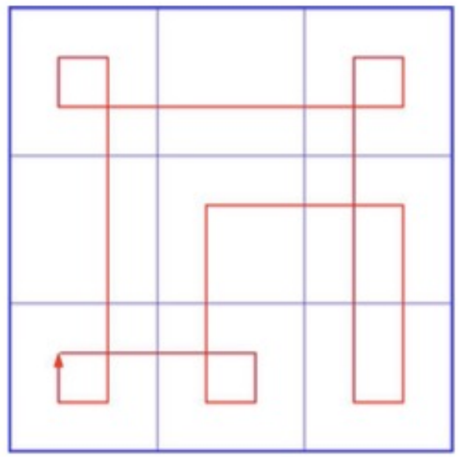
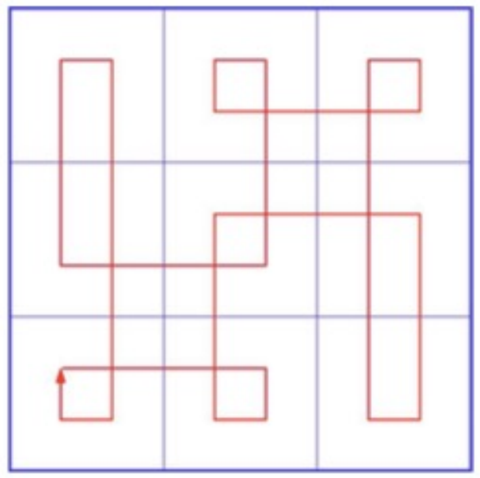
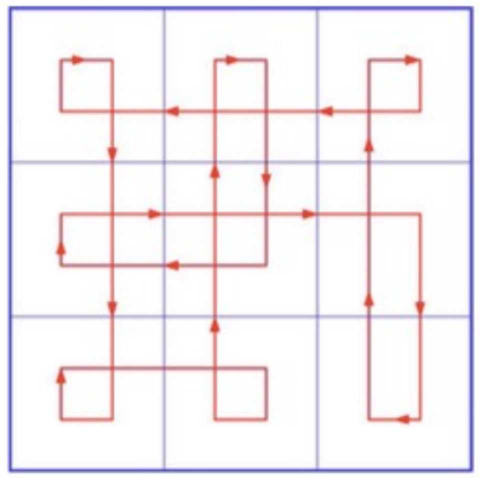
Actions

- `DO(a): a={suck, rotate, forward}`
  - “rotate” means rotate clockwise the direction of moving

Solution in two steps

- Find a cyclic path that allows the robot to go through all positions
- Write the navigations rules
  - The order will define the priority of the rules

Possible solutions

Rules

$In(x,y) \wedge Dirt(x,y) \rightarrow DO(suck)$

$[In(0,0) \vee In(1,2)] \wedge \neg Facing(E) \rightarrow DO(turn)$

$[In(2,0) \vee In(2,1)] \wedge \neg Facing(W) \rightarrow DO(turn)$

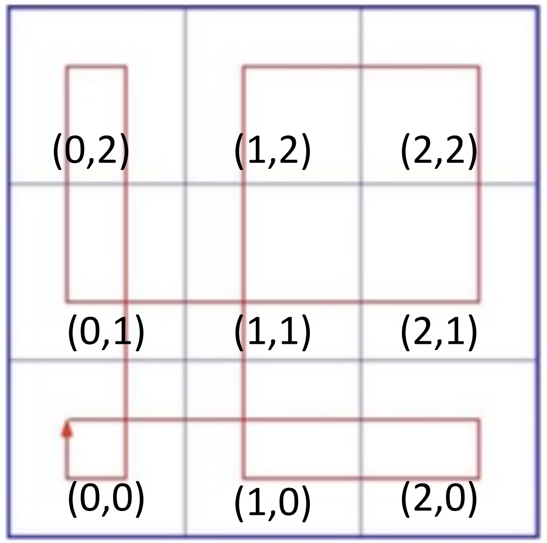
$[In(0,2) \vee In(2,2)] \wedge \neg Facing(S) \rightarrow DO(turn)$

$In(1,0) \wedge [Facing(W) \vee Facing(S)] \rightarrow DO(turn)$

$In(1,1) \wedge [Facing(E) \vee Facing(S)] \rightarrow DO(turn)$

$In(0,1) \wedge [Facing(W) \vee Facing(E)] \rightarrow DO(turn)$

$DO(forward)$

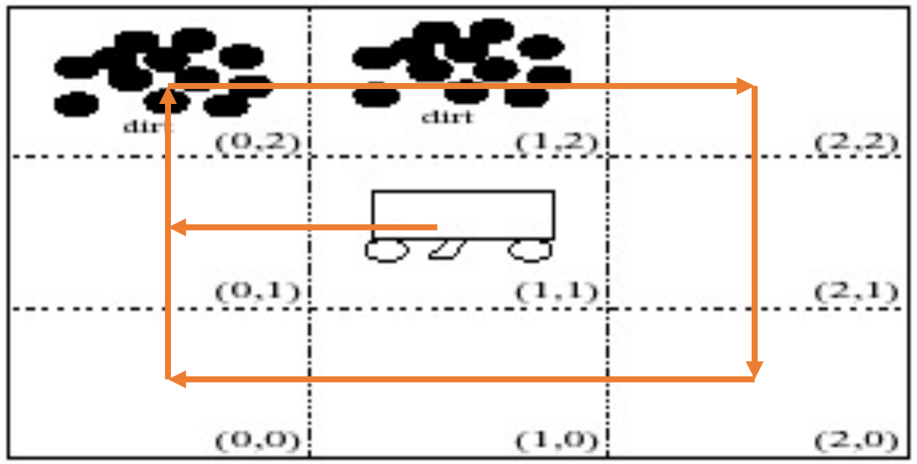


Extended

- A new predicate `free(x,y,d)`

```
In(x,y)^Dirt(x,y)-> Do(suck)
free(x,y,d)-> Do(forward)
Do(rotate)
```

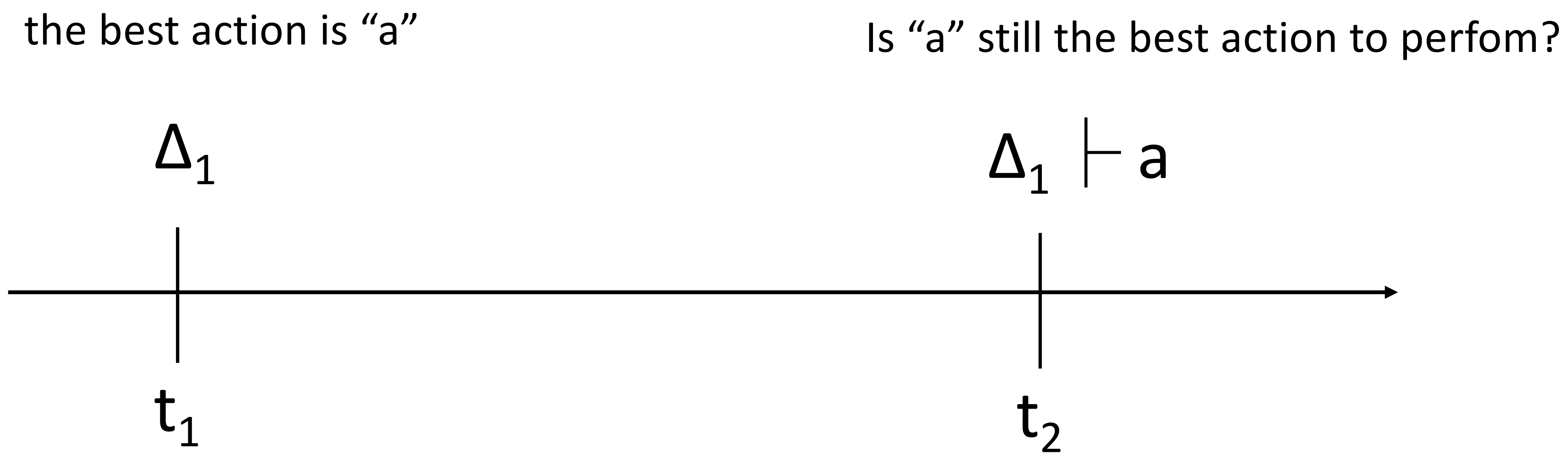
- Try with memory



Problems with DRA

- If we can prove `Do(a)`, then `a` is an optimal action
  - namely, the best action that could be performed when the environment is as described in the database
- Suppose at time t_1 the agent's database is Δ_1 and `a` is the optimal action to be performed. At time t_2, the agent deduces `a` (it took some time to do it) so the agent start to do `a`
- What happened between t_1 and t_2?
  - If $t_2 - t_1$ is infinitesimal, no problem (decision making was instantaneous)
  - Logic-based reasoning is not instantaneous !

Calculative rationality



Calculative rationality

- An agent is said to enjoy the property of calculative rationality if and only if its decision –making apparatus will suggest an action that was optimal when the agent decision-making process began.
- Calculative rationality is clearly not acceptable in environment that change faster than the agent make decisions
- Moving away from strictly logical representation languages and complete sets of deduction rules, one can build agents that enjoy respectable performance
    - Losing obviously advantages like simplicity and logical semantics

Other problems

The problem of “translating” raw data provided by the agent’s sensors into an internal symbolic form

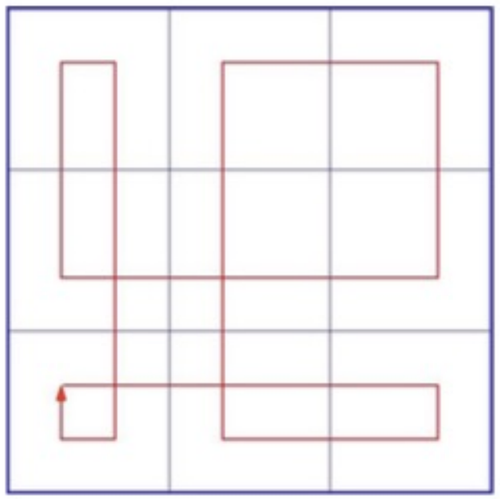
- How to convert video camera input to $Dirt(0, 1)$?
- Representing properties of dynamic, real-world is extremely hard (e.g., represent and reason about temporal information)

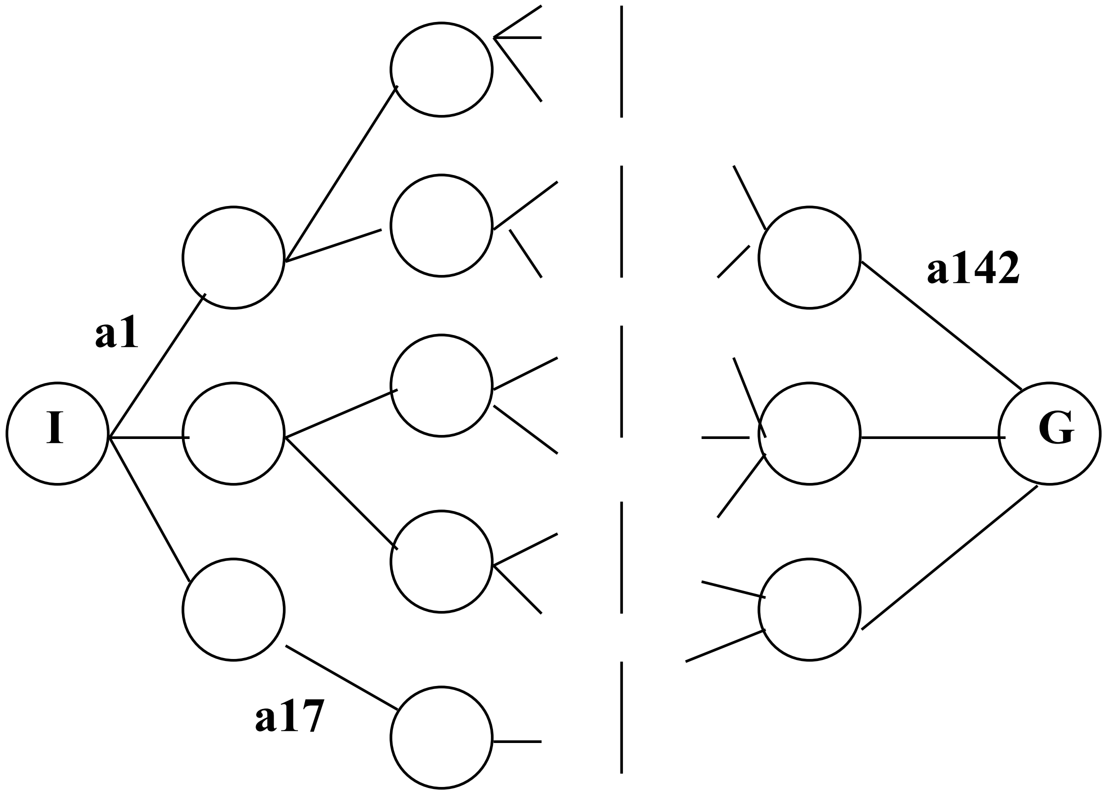
- Decision making using first-order logic is *undecidable*!
  - Even where we use *propositional* logic, decision making in the worst case means solving NP-complete problems

- Typical solutions:
  - weaken the logic
  - use symbolic, non-logical representations
  - shift the emphasis of reasoning from *run time* to *design time*

Planning Systems (in general)

- A reasoning agent is one that is sable to create a plan of actions
  - find a sequence of actions that transforms an initial state into a goal state





Planning

- Planning involves issues of both Search and Knowledge Representation
  - Eg., Robot Planning - Planning of speech acts

The Blocks World consists of equal sized blocks on a table

- A robot arm can manipulate the blocks using the actions:

```
UNSTACK(a,b)
STACK(a,b)
PICKUP(a)
PUTDOWN(a)
```

The Blocks World

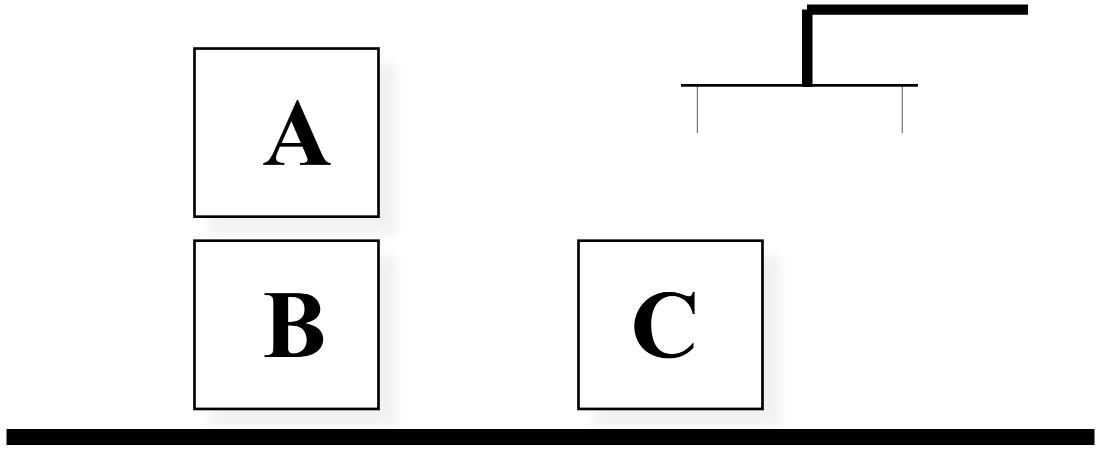
- The Blocks World (today) consists of equal sized blocks on a table
- A robot arm can manipulate the blocks using the actions:

```
UNSTACK(a,b)
STACK(a,b)
PICKUP(a)
PUTDOWN(a)
```

The Blocks World

- We also use predicates to describe the world:

```
ON(A,B)
ONTABLE(B)
ONTABLE(C)
CLEAR(A)
CLEAR(C)
ARMEMPTY
```



Formulas to describe facts always True

we can write general logical truths relating the predicates:

$\exists x\ \mathrm{HOLDING}(x) \rightarrow \neg\mathrm{ARMEMPTY}$

$\forall x(\mathrm{ONTABLE}(x) \rightarrow \neg\exists y(\mathrm{ON}(x, y)))$

$\forall x(\neg\exists y(\mathrm{ON}(y, x)) \rightarrow \mathrm{CLEAR}(x))$

So…how do we use theorem-proving techniques to construct plans?

Green’s Method

- Add state variables to the predicates, and use a function `DO` that maps actions and states into new states

$$\texttt{DO: AxS} \rightarrow \texttt{S}$$

- Example:

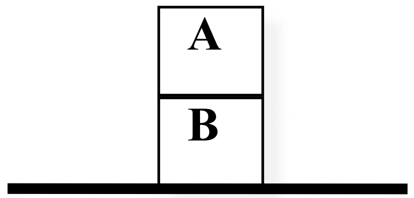
`DO(UNSTACK(x,y),s)` is a new state

UNSTACK

- To characterize the action UNSTACK we could write:

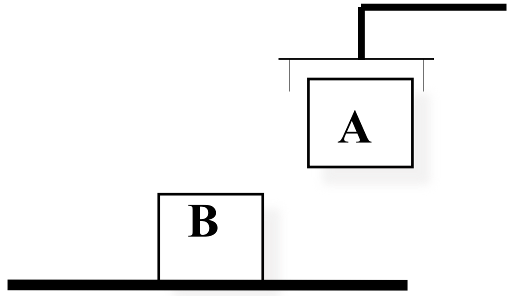
$$\mathrm{CLEAR}(x,s) \land \mathrm{ON}(x,y,s) \rightarrow$$
$$(\mathrm{HOLDING}(x,\mathrm{DO}(\mathrm{UNSTACK}(x,y),s)) \land$$
$$\mathrm{CLEAR}(y,\mathrm{DO}(\mathrm{UNSTACK}(x,y),s)))$$

- We can prove that if S_0 is



$$\mathrm{ON}(A,B,S_0) \land \mathrm{CLEAR}(A,S_0)$$ then

$$\mathrm{HOLDING}(A,\overbrace{\mathrm{DO}(\mathrm{UNSTACK}(A,B),S_0)}^{s_1}) \land$$
$$\mathrm{CLEAR}(B,\mathrm{DO}(\underbrace{\mathrm{UNSTACK}(A,B),S_0}_{s_1}))$$



$$\mathrm{CLEAR}(B,s1) \land \mathrm{HOLDING}(A,s1)$$

More Proving

- The proof could proceed further; if we characterize PUTDOWN:

  HOLDING(x,s)→ONTABLE(x,DO(PUTDOWN(x),s))

- Then we could prove:

$$\mathtt{ONTABLE(A,}\ \underbrace{\mathtt{DO(PUTDOWN(A),}\ \underbrace{\mathtt{DO(UNSTACK(A,B),\ S0)}}_{s_1}\mathtt{)}}_{s_2}\mathtt{)}$$

- The nested actions in this constructive proof give you the plan:
  1. UNSTACK(A,B); 2. PUTDOWN(A)

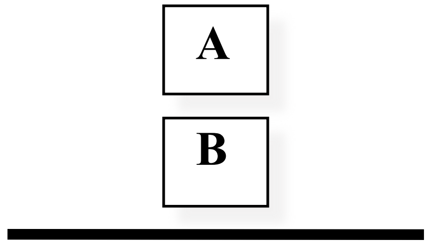
More Proving

- if we have in our database:

$\mathrm{ON}(A,B,S_0) \wedge \mathrm{ONTABLE}(B,S_0) \wedge \mathrm{CLEAR}(A,S_0)$

and our goal is

$\exists s: \mathrm{ONTABLE}(A,s)$



we could use theorem proving to find the plan

But how?

$$\mathrm{ONTABLE}(B, \mathrm{DO}(\mathrm{PUTDOWN}(A), \mathrm{DO}(\mathrm{UNSTACK}(A,B), S_0)))$$

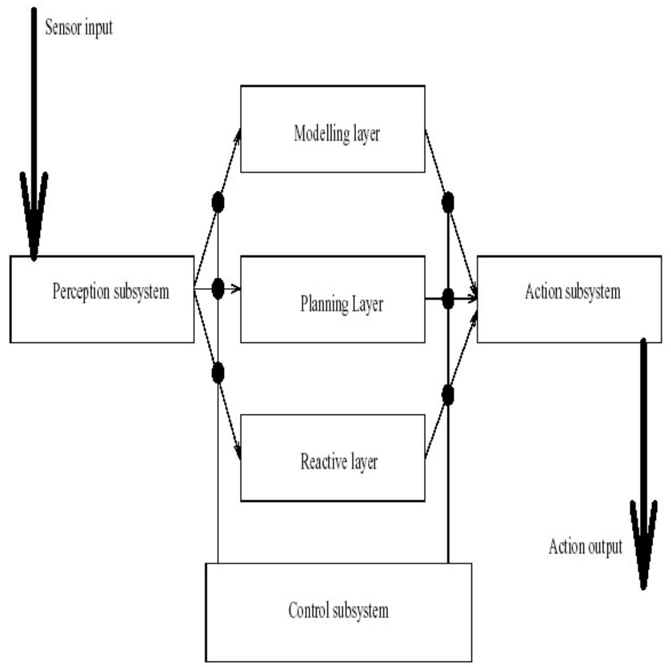
Hybrid Architectures

- A hybrid architecture is based on two (or more) parts:
  - deliberative, containing a symbolic world model, which develops plans and makes decisions in the way proposed by symbolic AI
  - reactive, which is capable of reacting to events without complex reasoning

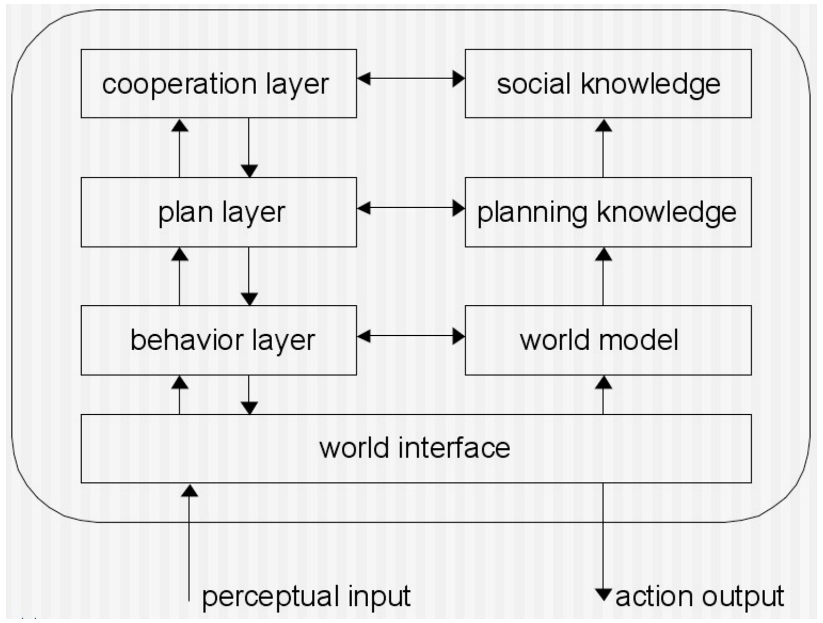
    (often, the reactive component is given some kind of precedence over the deliberative one)
- This kind of structuring leads naturally to the idea of a layered architecture -- the agent's control subsystems are arranged into a hierarchy, with higher layers dealing with information at increasing levels of abstraction
  - Horizontal layering: each layer is directly connected to the sensory input and action output. Each layer itself acts like an agent, producing suggestions as to what action to perform.
  - Vertical layering: sensory input and action output are each dealt with by at most one layer each

Two examples

Ferguson – TOURINGMACHINES



Müller –InteRRaP



BDI architecture

- The most popular and used in agent community (Rao and Georgeff, 1995)
- Many implementations
  - PRS (1987), dMARS (1998), JAM (1999), Jack (2001), JADEX (2005)
- A model of practical reasoning (1980s) where:
  - Environment can evolve in many (unpredictable) directions
  - Many potential courses of actions available at any point
  - Many potential objectives at any point
  - Environment can only be sensed locally
  - Environment dynamic - implies resource bound in reasoning!
- Additional assumption: plans!
  - "Plan" used in the sense of a known recipe ("know-how") rather than a chosen course of action (i.e., not "here" is the plan")

BDI: Belief

- Information about the world, the past, …
- Why?
  - World is dynamic, so the agent needs to remember the past
  - Agent has a local view, so it needs to remember
  - Agent resource bounded, so store information rather than re-compute
- Since stored information is imperfect, semantics should be beliefs, not knowledge
  - Knowledge (in the context of logics of knowledge of belief) has the property that is agent knows P then P must be true

- Note: in addition to caching beliefs, also cache plans (recipes) for similar reasons

BDI: (Goal)Desire

- Desired end state (e.g., “you desire to graduate”)
- Captures why a particular piece of code is/was executing
  - … useful for recovering from failure (cf. execution cycle)
  - … (potentially) allows for reasoning about goal interaction

BDI: Intention

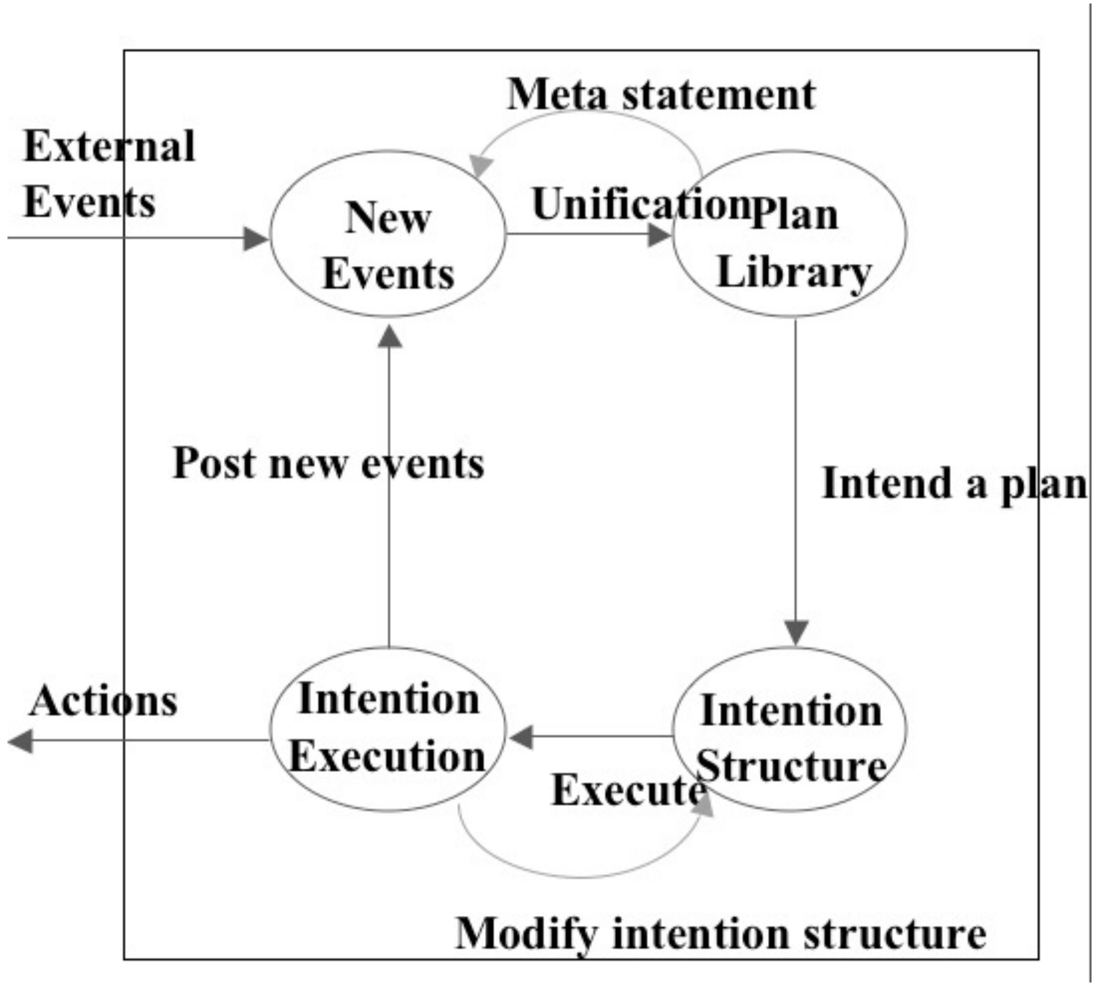
- Question: Are beliefs and desires enough?
- Answer: No, also need intentions
- Intention = selected course of action (plan instances)
- Why?
  - Resource bounded agents in dynamic environment need to plan bit-by-bit while acting
  - Intentions capture the partial plan (in the sense of "course of action")
  - Intentions can also be used to coordinate agents
- Properties:
  - Intentions lead to action
  - Intentions are persistent by default
  - Intentions should be internally consistent
  - Intentions should be fleshed out by the time they need to be "executed"

Commitment, re-planning, and options

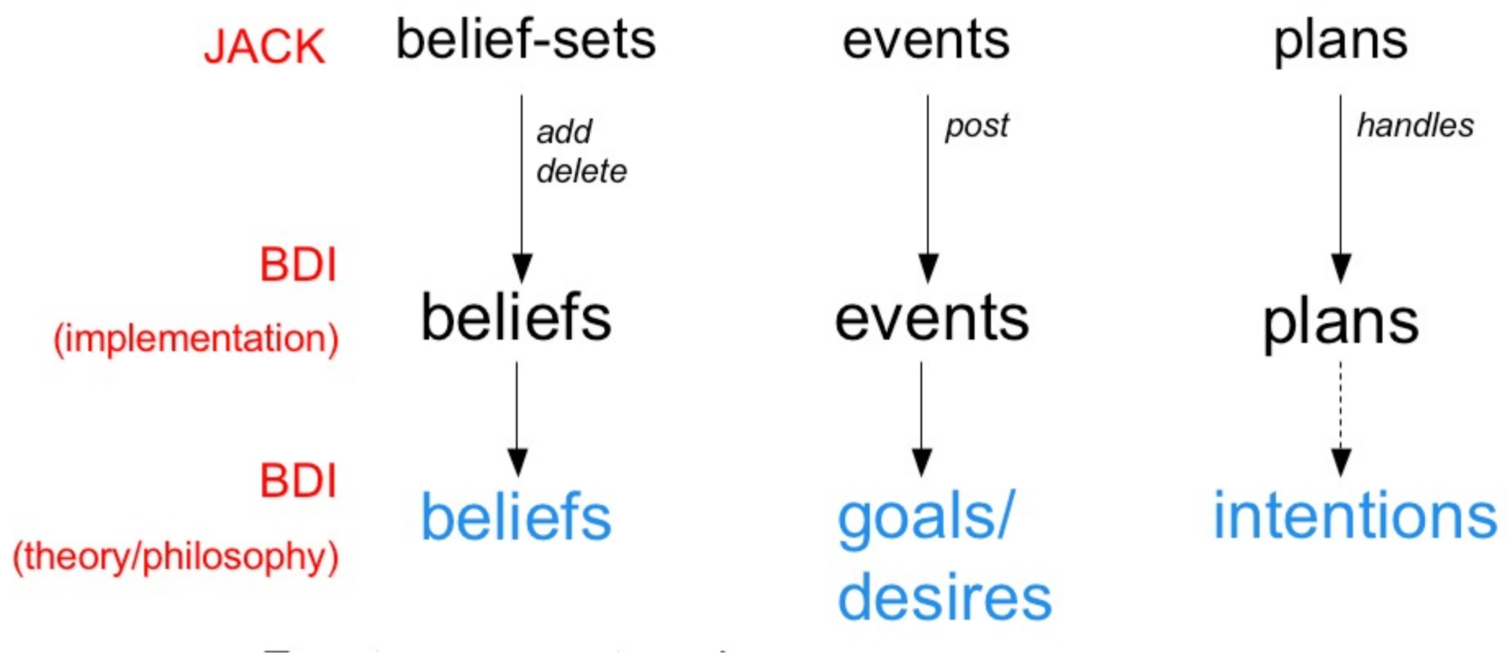
- You are in the middle of following an intention, and the world changes, should you re-plan?
  - Classical decision theory: yes, always
  - Classical computing (no goals): no, never
- Neither is ideal - want mixture!
- A deliberating agent has many available options (too many to consider!), so want to reduce the options to consider
  - Don't consider options that conflict with selected intentions
  - Don't reconsider chosen options (unless there's a good reason)
- When doing further planning, assume that intentions will have been achieved
  - e.g., intend to return library book in the afternoon and go to movies in the evening, since library is at UNITN@Albere, plan to go to movies in the city

BDI: Implementation in Jack

- Architecture vs implementations
- **B**eliefs - local knowledge base (really database)
- **E**vents - used to model goals
- **P**lans - predetermined sequences of actions or sub-goals that can accomplish specified tasks
- **I**ntentions - currently running plans



BDI Theory and Jack implementation


JACK
belief-sets
add
delete
events
post
plans
handles
BDI
(implementation)
beliefs
events
plans
BDI
(theory/philosophy)
beliefs
goals/
desires
intentions