Agent Architectures

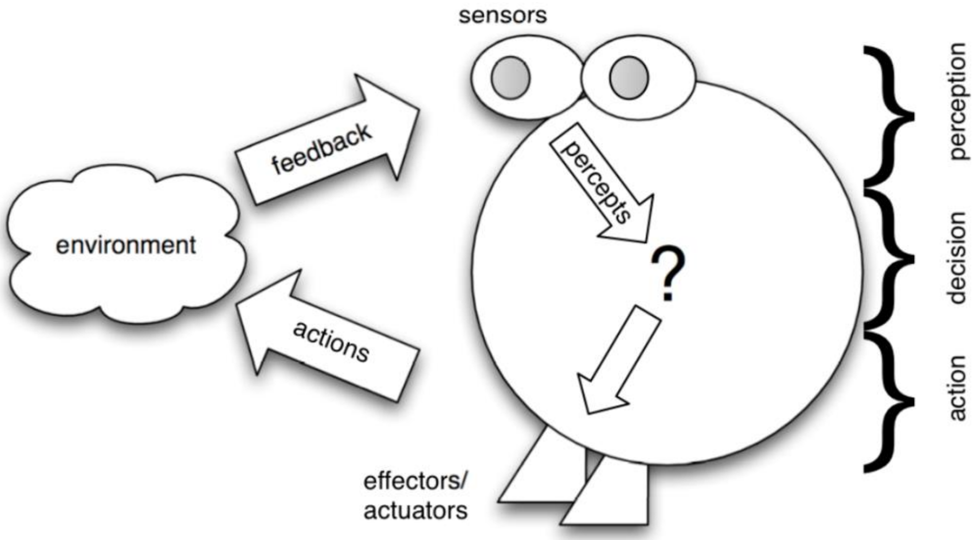
MICHAL JAKOB
Artificial Intelligence Center,
Dept. of Computer Science,
FEE, Czech Technical University

AE4M36MAS Autumn 2019 - Lect. 2

Defining Agency

INTRODUCTION TO MULTI-AGENT SYSTEMS

What is Agent?



Definition (Russell & Norvig)

- An agent is anything that can perceive its environment (through its sensors) and act upon that environment (through its effectors)

Focus on **situatedness** in the environment (**embodiment**)

The agent can only **influence** the environment but not fully control it (sensor/effector failure, non-determinism)

What is Agent? (2)

Definition (Wooldridge & Jennings)

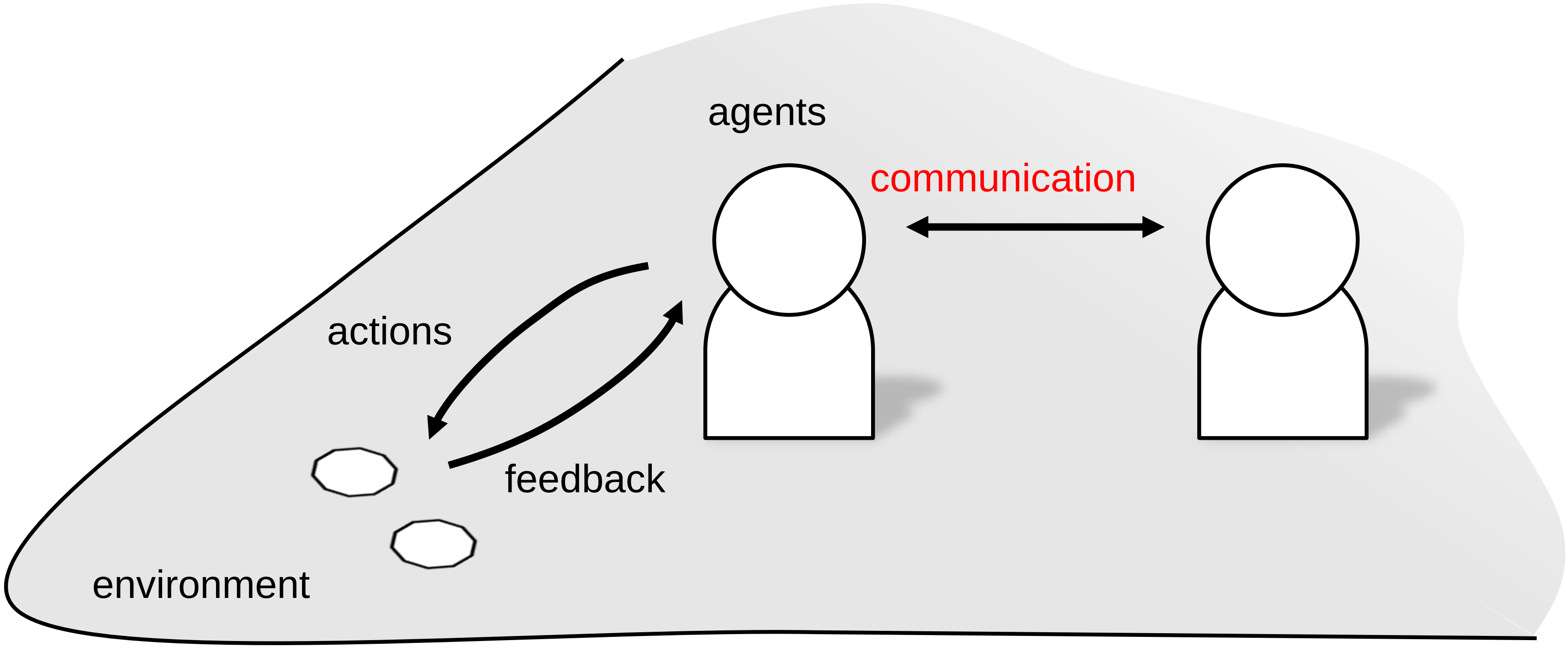
- An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to **meet its design objectives/delegated goals.**

Adds a second dimension to agent definition: the relationship between agent and designer/user

- agent is capable of **independent action**
- agent action is **purposeful**

Autonomy is a central, distinguishing property of agents

Multiagent Systems



Agents vs. Objects

An agent has unpredictable behaviour as observed from the outside

- unless its simple reflexive agent

An agent is *situated* in the environment

Agent communication model is *asynchronous*

Objects do it for free; agents do it because they want to

Where are we?

Agent architectures (inc. BDI architecture)

Non-cooperative game theory

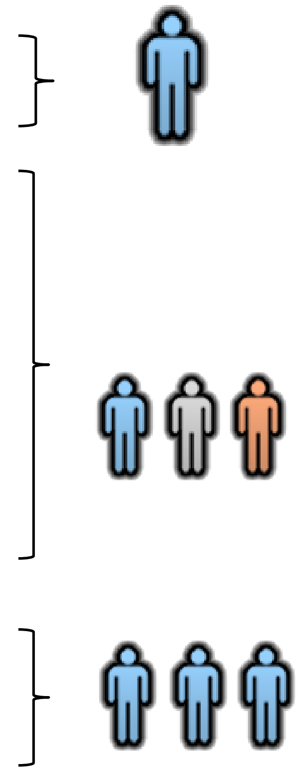
Coalition game theory

Mechanism design

Auctions

Social choice

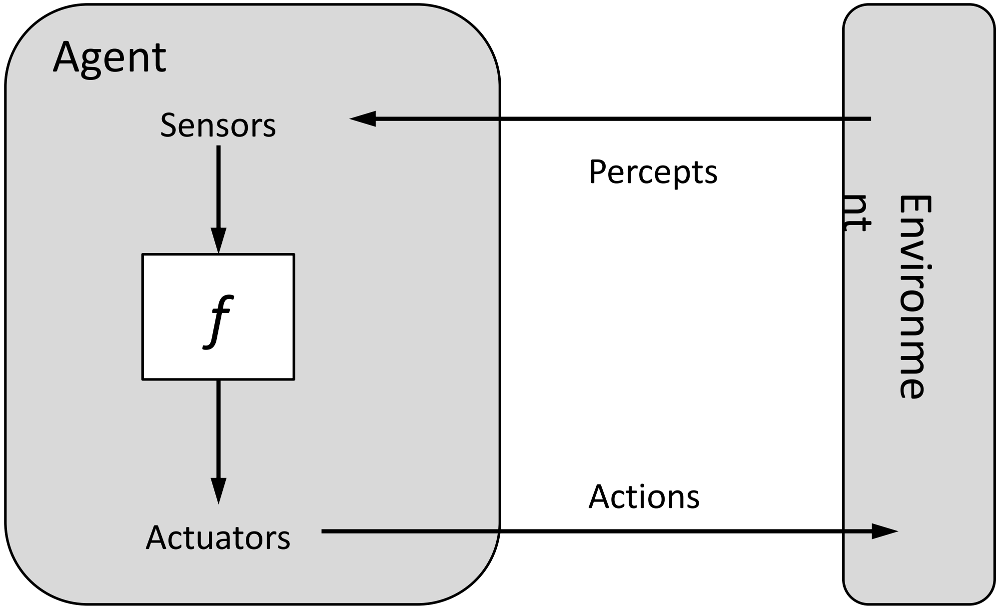
Distributed constraint reasoning
(satisfaction and optimization)



Specifying Agents

INTRODUCTION TO MULTIAGENT SYSTEMS

Agent Behavior



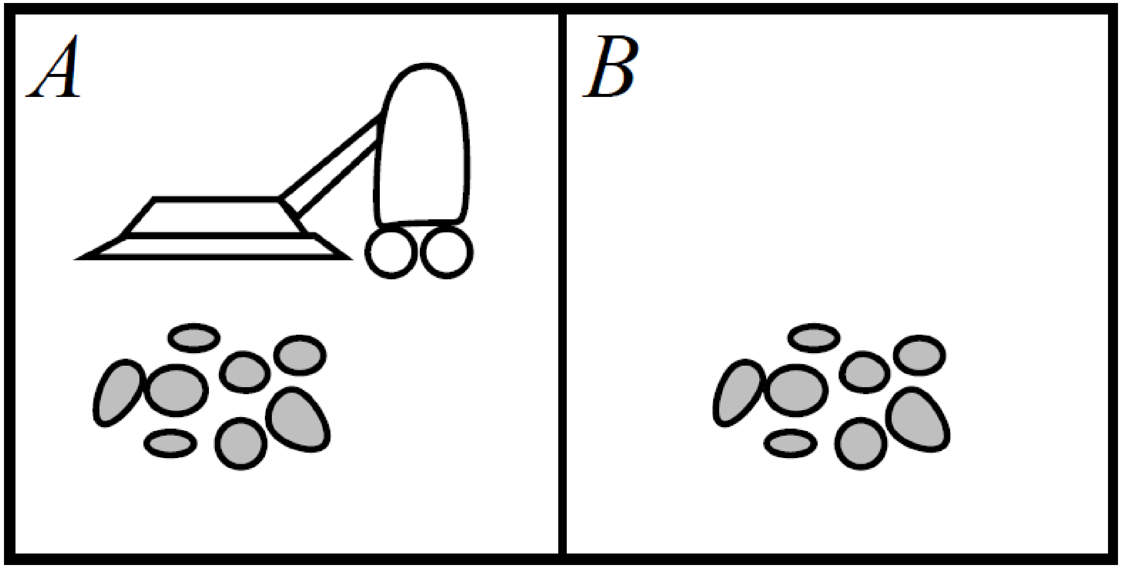
Agent's behavior is described by the **agent function** that maps percept sequences to actions

$$f : \mathscr{P} \mapsto \mathscr{A}$$

The **agent program** runs on a physical architecture to produce *f*

Key questions: 1) What is the right agent function? 2) Can it be implemented in a small agent program?

Example: Vacuum Cleaner World



Percepts: location and contents, e.g. [A, Dirty]

Actions: Left, Right, Suck, NoOp

Vacuum Cleaner Agent

Percept sequence	Action
[A,Clean]	Right
[A, Dirty]	Suck
[B,Clean]	Left
[B, Dirty]	Suck
[A,Clean], [A,Clean]	Right
[A,Clean], [A, Dirty]	Suck
…	…
[A,Clean], [A,Clean], [A,Clean]	Right
[A,Clean], [A,Clean], [A, Dirty]	Suck
…	…

Is this a good agent function?

Rational Behavior

Definition (Russell & Norvig)

Rational agent chooses whichever **action maximizes** the expected value of the **performance measure** given the **percept sequence** to date and whatever **built-in knowledge** the agent has.

Rationality is relative and depends on four aspects:

1. performance measure which defines the degree of success
2. percept sequence (complete perceptual history)
3. agent's knowledge about the environment
4. actions available to the agent

Rational $\neq$ omniscient and rational $\neq$ clairvoyant
$\Rightarrow$ **rational $\neq$ successful**

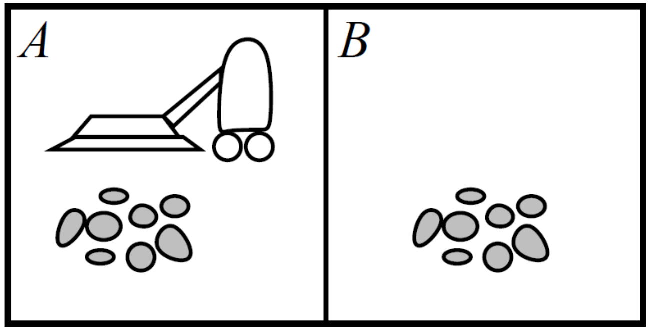
Specifying Task Environments

To design a rational agent, we must specify the **task environment (PEAS)**

1. Performance measure
2. Environment
3. Actuators
4. Sensors

Task environments define **problems** to which rational agents are the **solutions**.

Rationality of Vacuum Cleaner Agent



Agent program:
Cleans a square if it is dirty and moves to the other square if not. **Is it rational?**

PEAS:

- The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
- The "geography" of the environment is known a priori but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The Left and Right actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
- The only available actions are Left, Right, and Suck.
- The agent correctly perceives its location and whether that location contains dirt.

Yes, we can prove no other agent does better.

PEAS Examples

Agent	Performance measure	Environment	Actuators	Sensors

Properties of Environments

Fully observable vs. partially observable – Can agents obtain complete and correct information about the state of the world?

Deterministic vs. stochastic – Do actions have guaranteed and uniquely defined effects?

Episodic vs. sequential – Can agents decisions be made for different, independent episodes?

Static vs. dynamic – Does the environment change by processes beyond agent control?

Discrete vs. continuous – Is the number of actions and percepts fixed and finite?

Single-agent vs. multi-agent – Does the behavior of one agent depends on the behavior of other agents?

Example Environments

	Solitaire	Backgammon (or Člověče, nezlob se)	Online shopping	Robotic Taxi
Observable				
Deterministic				
Episodic				
Static				
Discrete				
Single-agent				

True or false?

An agent that senses only partial information about the state cannot be perfectly rational.

- False

There exists a task environment in which every agent is rational.

- True

Every agent is rational in an unobservable environment.

- False

A perfectly rational poker-playing agent never loses.

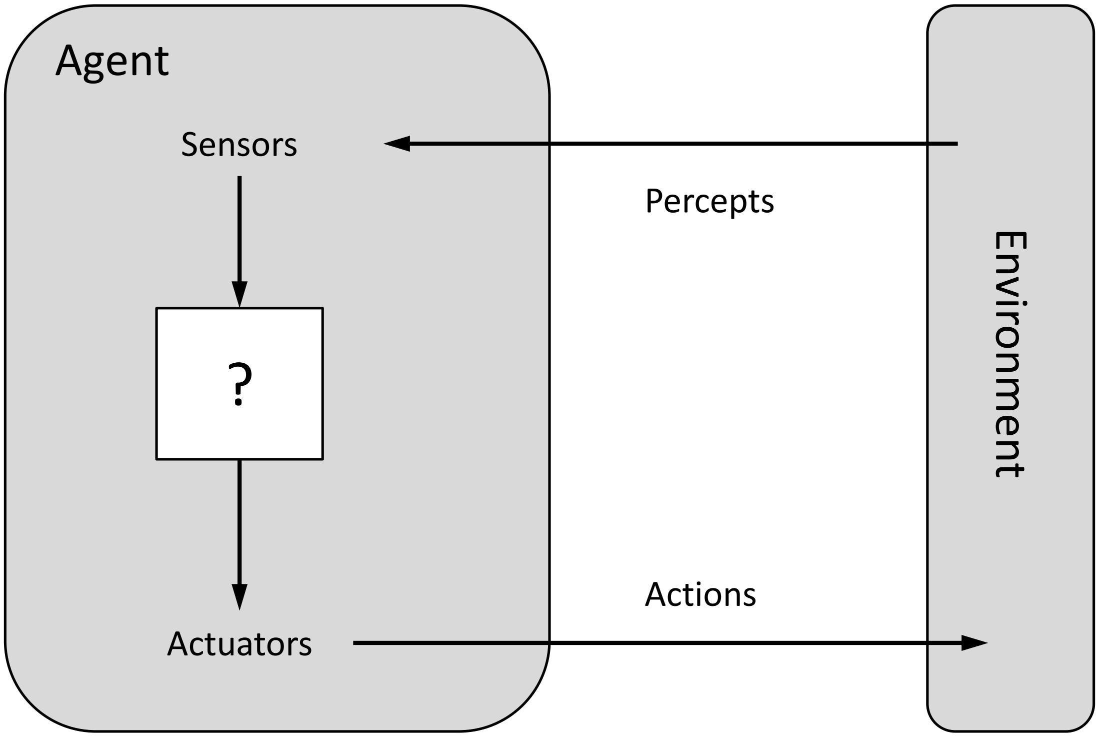
- False

Agent Architectures

INTRODUCTION TO AGENTS

Implementing the Agent

How should one implement the agent function?



Concern 1: **Rationality**

Concern 2: **Computability and tractability**

Hierarchy of Agents

The key challenge for AI is to find out how to write **programs** that produce **rational behavior** from a **small amount of code** rather than from a large number of table entries.

4+1 basic types of agents in the order of increasing capability:

1. simple reflex agents
2. model-based agents with state
3. goal-based agents
4. utility-based agents
5. (learning agents)

There is a link between the **complexity of the task** and the **minimum agent architecture** required to implement a **rational** agent.

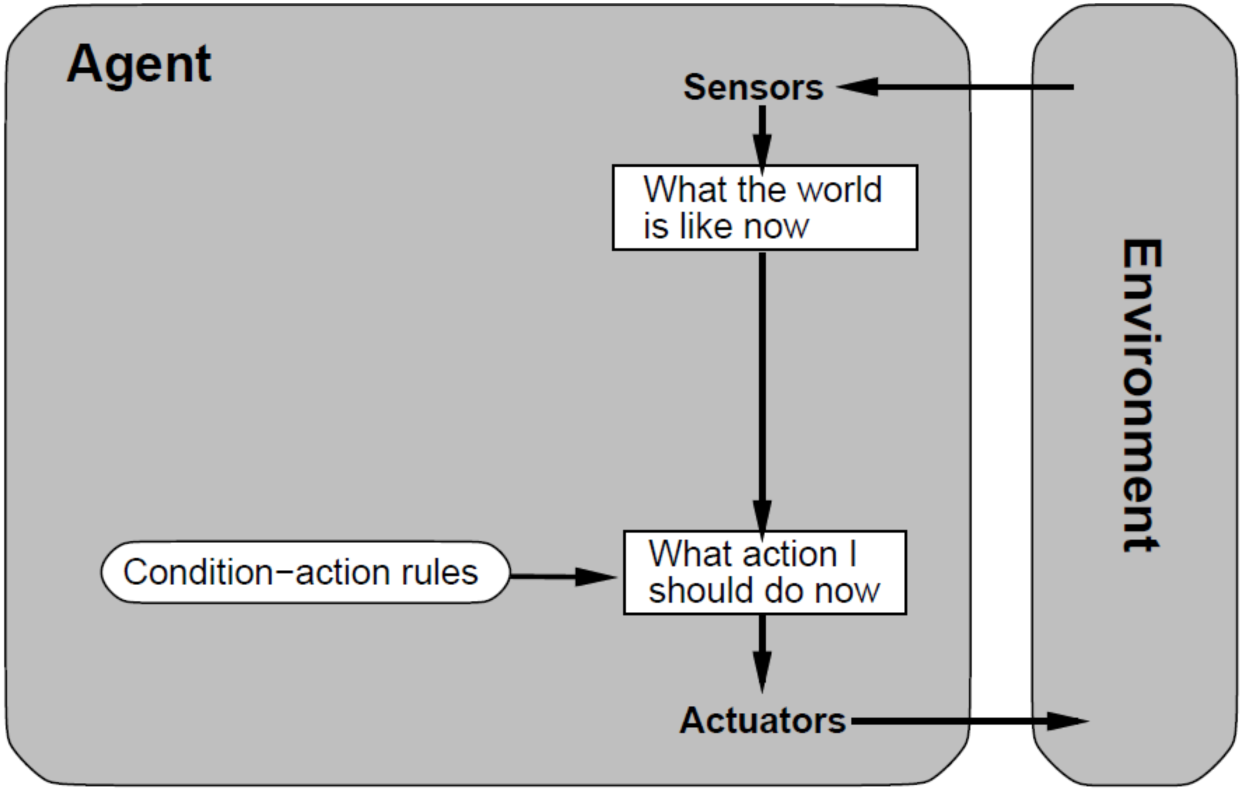
Running Example: Robotic Taxi

Task specification

- **Performance measure**: the overall profit (= passenger revenues - fines)
- **Environment**: road network with traffic signs, passengers
- **Actions (actuators)**: driving between junctions, picking up and dropping out passengers
- **Percepts (sensors)**: current GPS location, junction layout, traffic signs, passengers

Simple Reflex Agents

Simple reflex agent chooses the next action on the basis of the current percept only.



Simple Reflex Agent

Condition-action rules provide a way to present common regularities appearing in input/output associations

- Ex.: `if car-in-front-is-braking then initialize-braking`

function SIMPLE-REFLEX-AGENT(*percept*) **returns an action**
  persistent: *rules*, a set of condition–action rules

  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

Simpe Reflex Agent for Robotic Taxi

Simple program:

- If a passenger at your location ⇒ pickup the passenger
- Otherwise: Continue in the left-most direction possible

More sophisticated program:

- Turn-directions depend on the current GPS location (can implement specific fixed route through the city)

Issues with Reflex Agents

Robotic taxi

- driving to a given destination
- respecting traffic signs (e.g. speed limits)
- getting stuck in loops

In general: Reflex agents are simple but of limited intelligence – the only work if

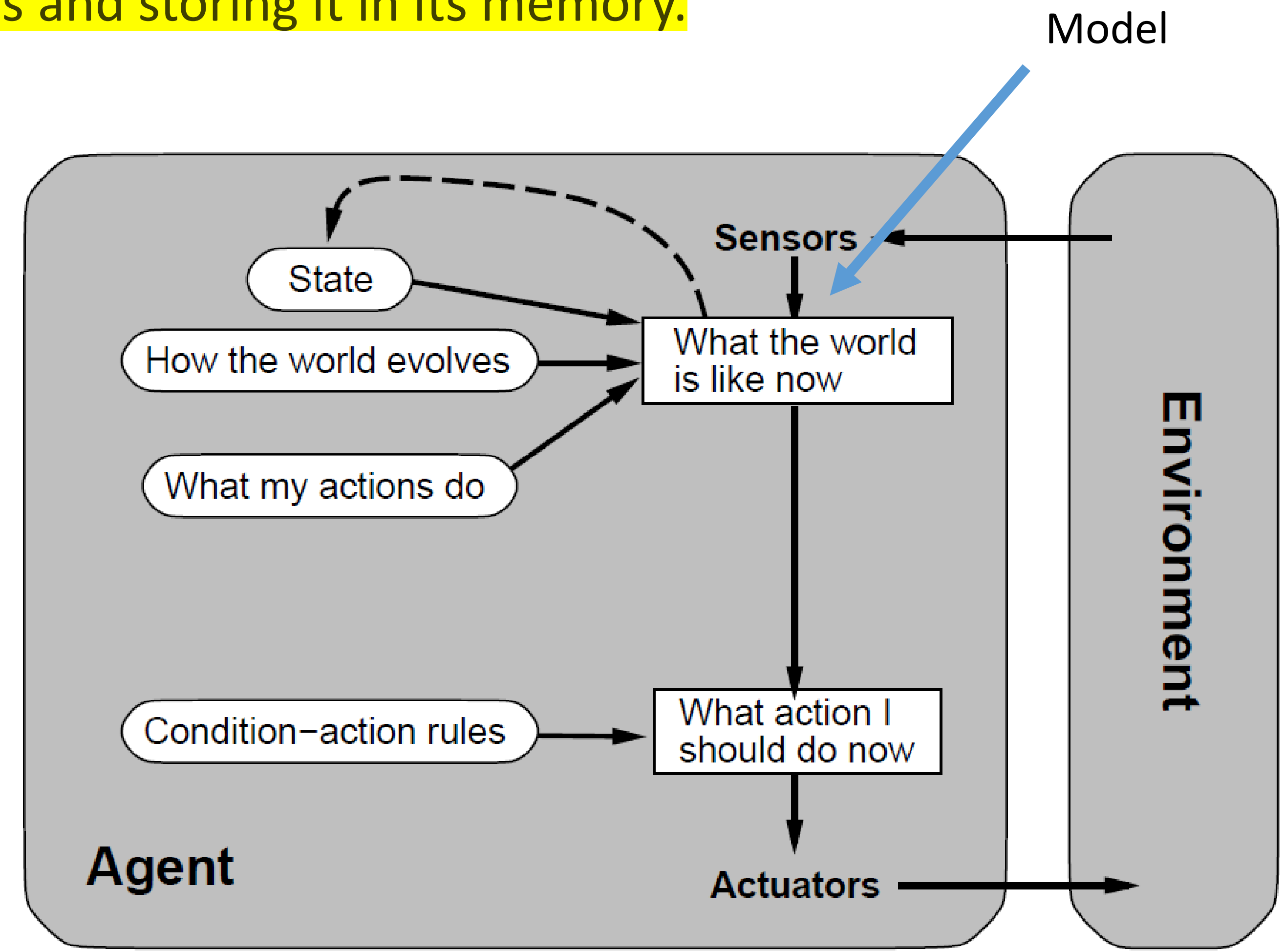
1. the environment is fully observable and
2. the decision can be made based solely on the current percept

If the above not the case ⇒ suboptimal action choices, infinite loops.

⇒ It can be advantageous to **store information about the world** in the agent.

Model-based Reflex Agent

Keeps track of the world by extracting relevant information from percepts and storing it in its memory.



Model-based Reflex Agent

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    model, a description of how the next state depends on current state and action
    rules, a set of condition–action rules
    action, the most recent action, initially none

  state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

Model-based Reflex Taxi Agent

States tracked in the model:

- passengers' destinations
- traffic signs
- visited locations (to avoid cycles)
- pickup locations (⇒ learning)

Issues with Model-based Agents

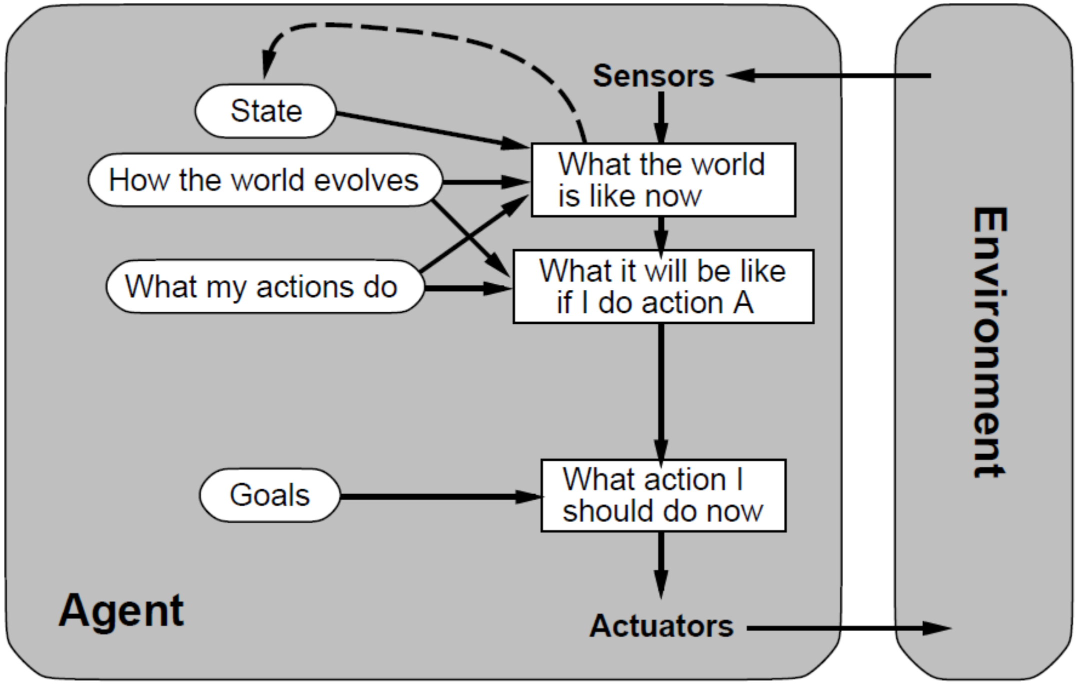
Taxi agent: Hot to get to a destination?

- Always move towards the destination location => can end-up in dead end streets
- Hard-code routes between all locations
  - memory demanding and of limited intelligence
  - e.g. requires reprogramming the agent if the street network changes

Cause:

- *whats* and *hows* **tightly coupled** (impossible to tell the agent what to do)
- the agent does not **anticipate the effects** of its actions (only finds out the result after having executed the action)

Goal-based Agents



Goal-based agents are more **flexible**

Problem: goals are not necessarily achievable by a single action:

→ **search and planning**

Goal-based Taxi Agent

Uses planning

- Uses a map to find a sequence of movement actions that brings the taxi to the destination reliable

Issue

- will not choose the fastest route
- will not balance revenue vs. fees/fines

Cause: goals alone are not sufficient for decision making:

1. there may be multiple ways of achieving them;
2. agents may have several conflicting goals that cannot be achieved simultaneously.

Utility-based Agents

Goals only a very crude (binary) distinction between “happy” and “unhappy” states.

We introduce the concept of **utility**:

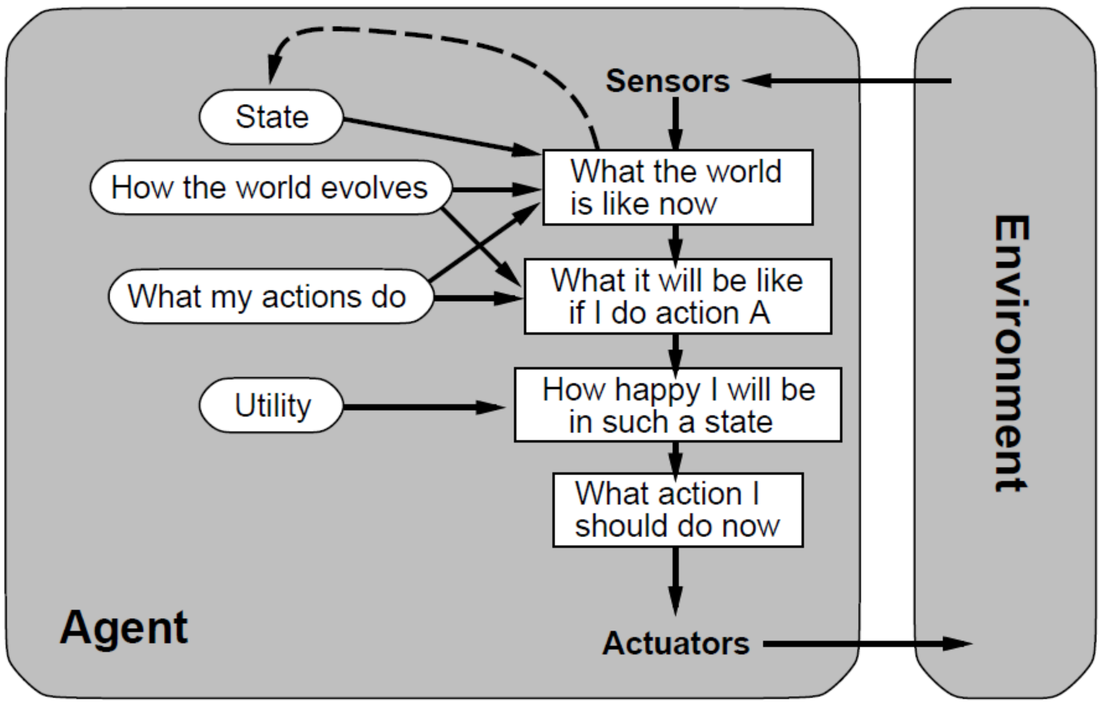
- utility is a function that maps a state onto a real number; it captures “quality” of a state
- if an agent prefers one world state to another state then the former state has higher utility for the agent.

Utility can be used for:

1. choosing the best plan
2. resolving conflicts among goals
3. estimating the successfulness of an agent if the outcomes of actions are uncertain.

Utility-based Agents

Utility-based agent use the utility function to choose the most desirable action/course of actions to take



Utility-based Taxi Agent

Uses **optimizing** planning

- searches for the plan that leads to the maximum utility

There are still issues

- irreducible preference orderings
- non-deterministic environment (→ Markov decision processes)

CZECH
TECHNICAL
UNIVERSITY
IN PRAGUE

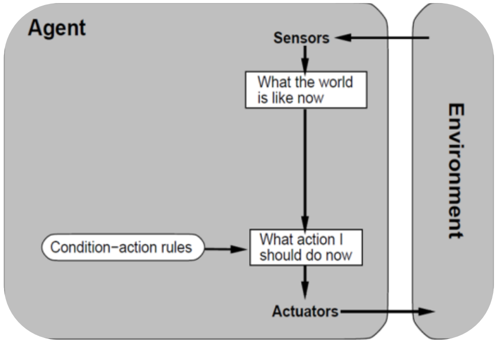
Belief-Desire-Intention (BDI) Architecture

MICHAL JAKOB
Artificial Intelligence Center,
Dept. of Computer Science,
FEE, Czech Technical University

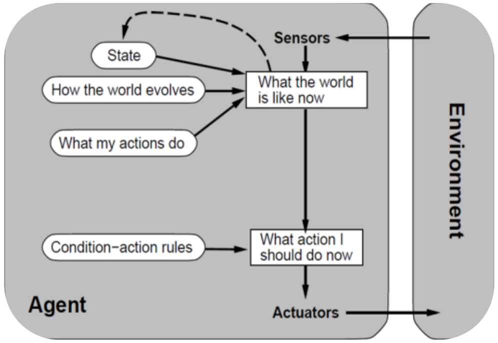
AE4M36MAS Autumn 2019 - Lect. 2

Intro and Motivation

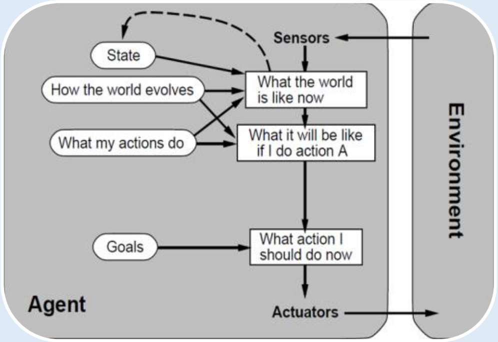
Basic Agent Architectures



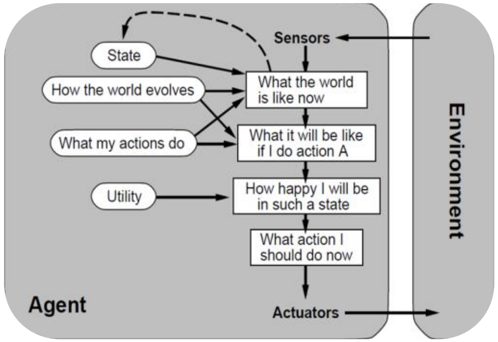
Reflex agent



Model-based agent

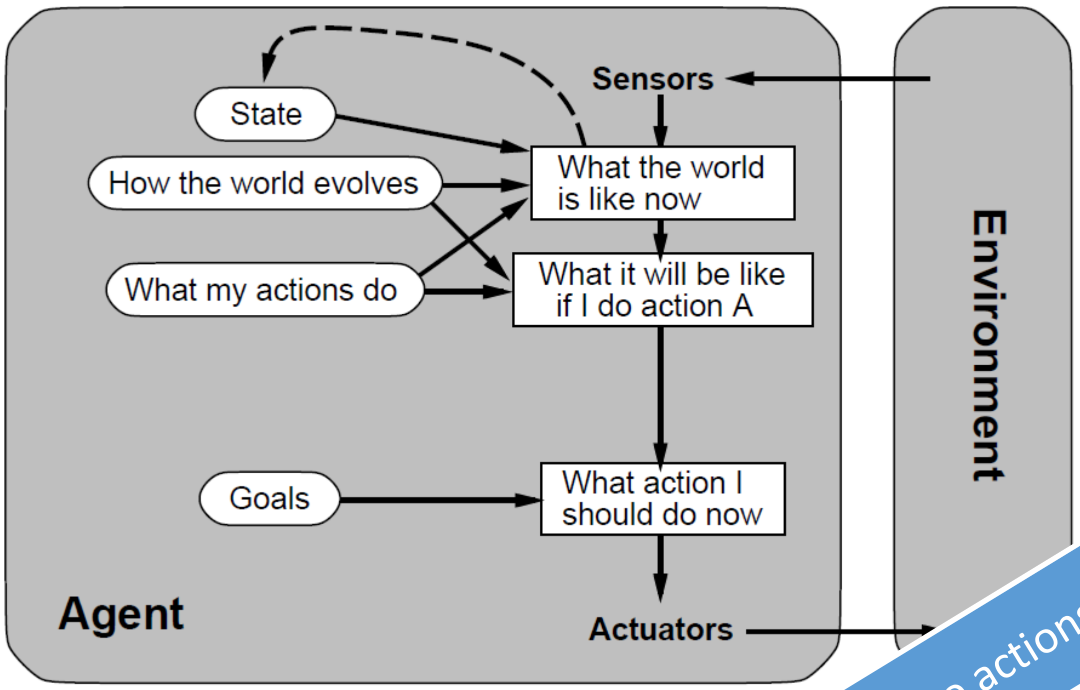


Goal-based agent



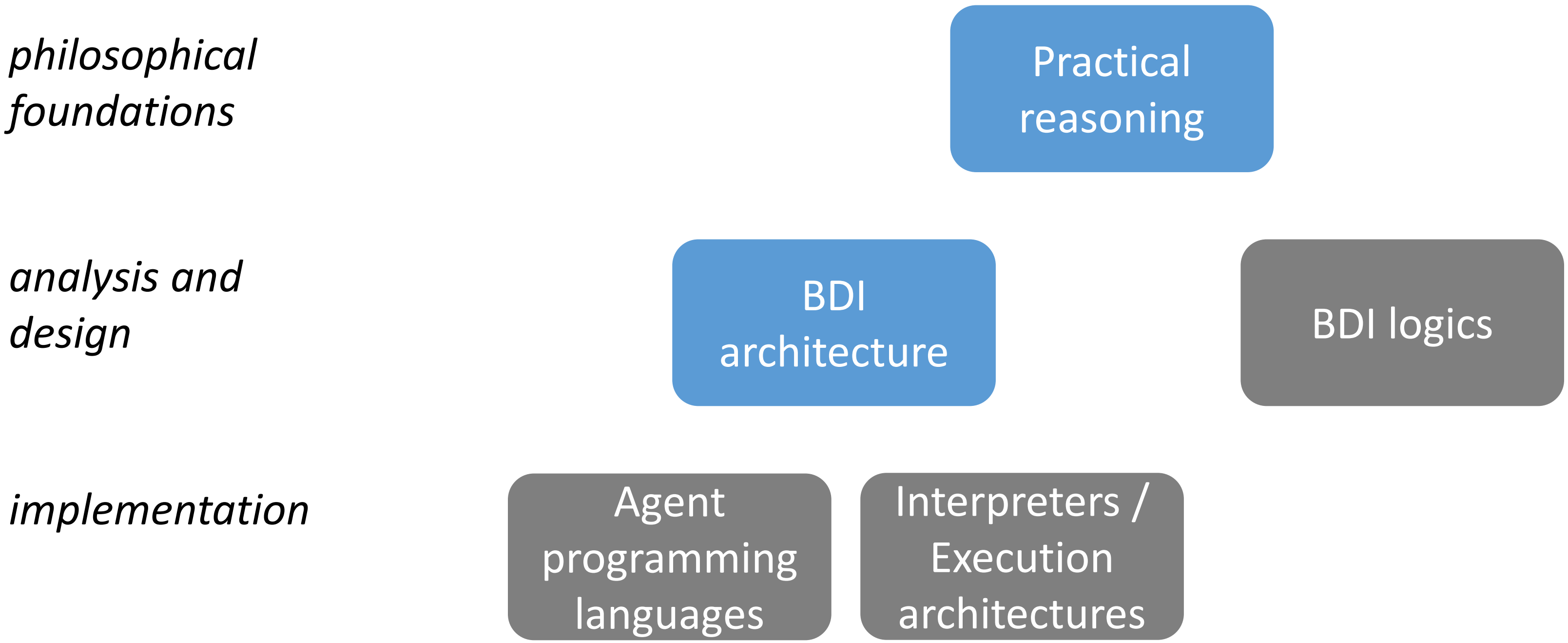
Utility-based agent

Goal-based agents



How to go from goals to actions effectively?

Big Picture



philosophical foundations
Practical reasoning
analysis and design
BDI architecture
BDI logics
implementation
Agent programming languages
Interpreters / Execution architectures

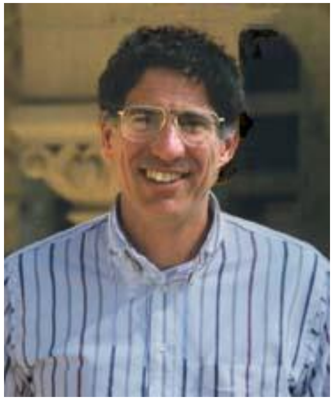
Practical Reasoning

CONCEPTUALIZING RATIONAL ACTION

Practical Reasoning

Practical reasoning is **reasoning directed towards actions** — the processof figuring out what to do (Michael Bratman, 1990).

"Practical reasoning is a matter of weighing conflicting ***considerations for and against competing options****, where the relevant considerations are provided by (i) what the agent* ***desires/values/cares about*** *and (ii) what the agent* ***believes****."*



Practical reasoning is different from theoretical reasoning.

Theoretical vs Practical Reasoning

Theoretical reasoning	Practical reasoning
Reasoning directed towards beliefs—concerned with **deciding what to believe**. ▪ Tries to assess the way things are. ▪ Process by which you change your beliefs and expectations.	Practical reasoning is reasoning directed towards actions—concerned with **deciding what to do.** ▪ Decides how the world should be and what individuals should do. ▪ Process by which you change your choices, plans, and intentions.

Components of Practical Reasoning

Deliberation (strategic)	**Means-ends reasoning** (tactical)
Deciding **what state of affairs** we want to achieve. ▪ considering preferences, choosing goals, etc.; ▪ balancing alternatives (decision-theory); ▪ the outputs of deliberation are **intentions**.	Deciding **how to** achieve these states of affairs. ▪ thinking about suitable actions, resources and how to “organize” activity; ▪ building courses of action (planning); ▪ the outputs of means-ends reasoning are **plans**.

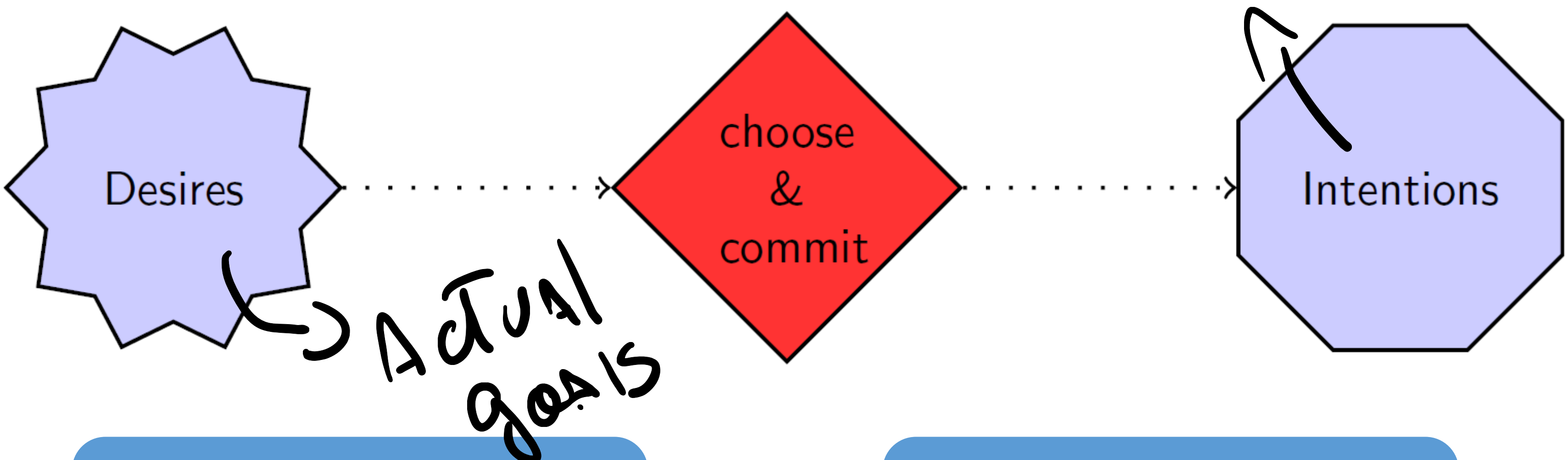
Need to combine appropriately

(⇐ agents are resource-bounded and the world is dynamic)

Deliberation

How does an agent deliberate?

Minigoals



Understand what the options available are:

- options available are **desires**.

Choose between them, and commit to some:

- chosen options are then **intentions**.

Desires

Desires describe the states of affairs that are **considered for achievement**, i.e., basic preferences of the agent.

Desires are much weaker than intentions; not directly related to activity:

> *"My **desire** to play basketball this afternoon is merely **a potential influence** of my conduct this afternoon. It must vie with my other relevant desires [...] before it is settled what I will do. In contrast, once I **intend** to play basketball this afternoon, **the matter is settled**: I normally need not continue to weigh the pros and cons. When the afternoon arrives, I will normally just proceed to execute my intentions." (Bratman 1990)*

Intentions

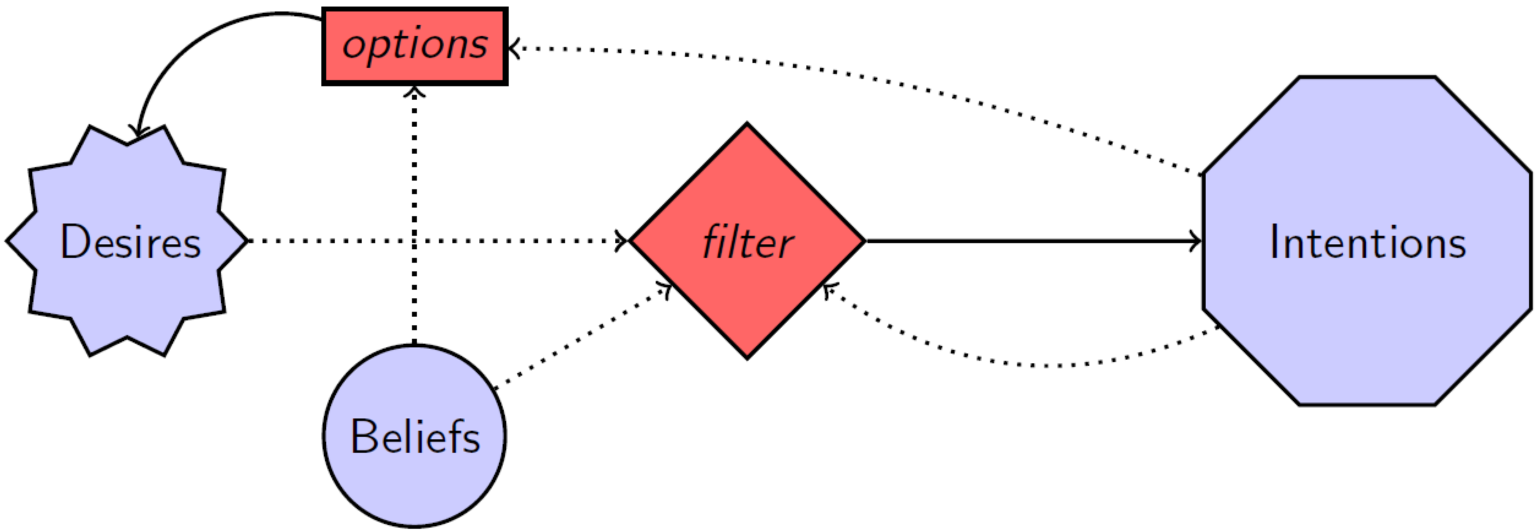
Intentions are **future-directed** intentions i.e. they are pro-attitudes leading to actions.

- Intentions are about the **(desired) future**.

We make **reasonable attempts to fulfill** intentions once we form them, but they may change if circumstances do.

- Behavior arises to fulfill intentions.
- Intentions affect action choice.

Functional Components of Deliberation



Option Generation—the agent generates a set of possible alternatives; via a function, **options**, which takes the agent's current beliefs and intentions, and from them determines a set of options/desires.

Filtering—the agent chooses between competing alternatives, and commits to achieving them. In order to select between competing options, an agent uses a **filter** function.

Properties of Intentions

1. Intentions **drive means-end reasoning**.
2. Intentions **constrain future** deliberation (i.e., provide a “filter”).
3. Intentions **persist**.
4. Intentions **influence beliefs** concerning future practical reasoning.
5. Agents believe their **intentions are possible**.
6. Agents **do not believe** they **will not bring** about their intentions.
7. Under certain circumstances, agents **believe they will bring** about their intentions.
8. Agents need not intend all the expected **side effects** of their intentions.

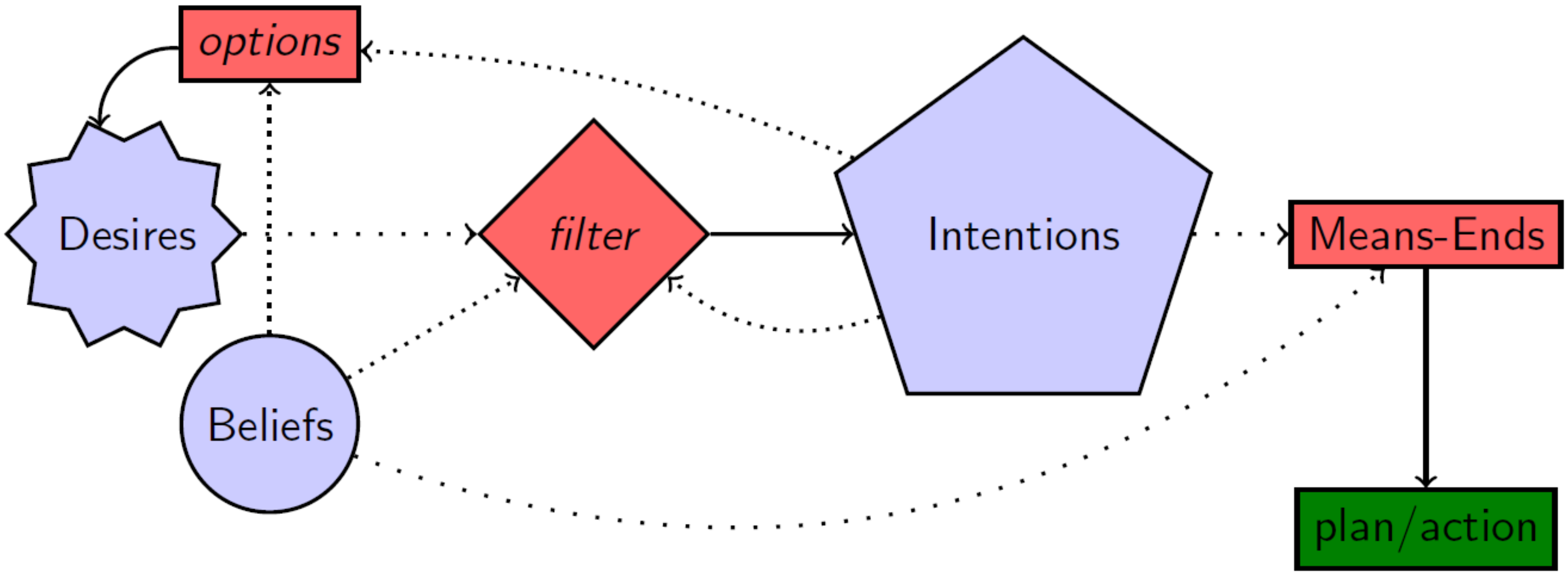
Plans

decide how to Achieve intention

Human practical reasoning consists of two activities:

- **Deliberation**: Deciding what to do. Forms **intentions**.
- **Means-ends reasoning**: Deciding how to do it. Forms **plans**.

Intentions drive means-ends reasoning: *If I adopt an intention, I will attempt to achieve it, this affects action choice.*



Means-End Reasoning: Obtaining Plans & Actions

How does the agent obtain plans/actions to realize our intentions?

Automated Planning: design a course of action that will achieve the goal. Given:

- (representation of) goal/intention to achieve;
- (representation of) actions it can perform; and
- (representation of) the environment;

... have it generate a plan to achieve the goal.

BDI-style programming (e.g., AgentSpeak, CAN, Jason, JACK, etc.)

BDI Architecture

OPERATIONALIZING PRACTICAL REASONING

BDI Programming

Objective: a programming language that can provide:

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

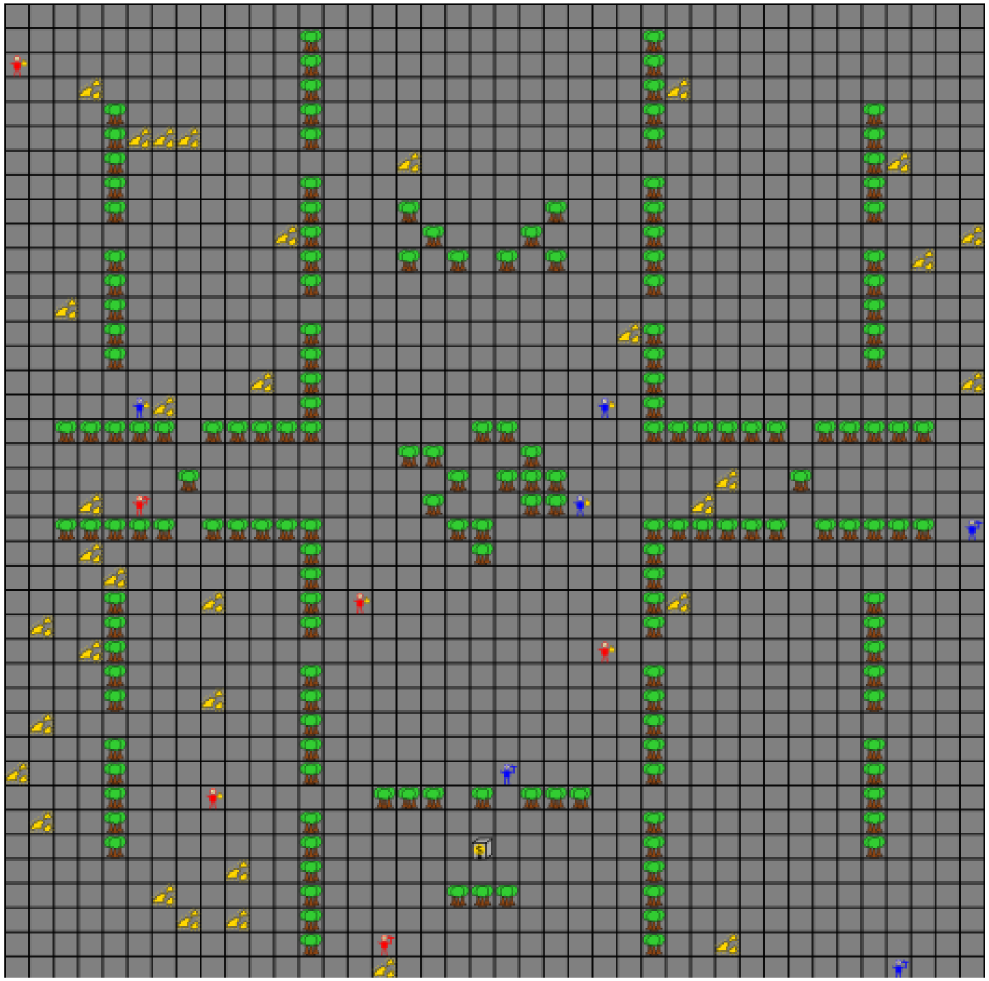
BDI Programming

BDI Programming = practical reasoning + events + plans

An Example: Gold Mining Game

Two teams competing to collect and drop gold in the depot

- dynamic
- complex
- unknown information
- failing actions
- failing sensors
- multi-agents



Some Constraints

We want to program intelligent systems under the following constraints:

The agent **interacts** with an external **environment**.

- A grid world with gold pieces, obstacles, and other agents.

The environment is (highly) **dynamic**; may **change** in unexpected ways.

- Gold pieces appear randomly.

Things can go **wrong**; plans and strategies may **fail**.

- A path may end up being blocked.

Agents have **dynamic** and **multiple** objectives.

- Explore, collect, be safe, communicate, etc
- Motivations/goals/desires may come and go.

Some Assumptions

Failure is generally not catastrophic.

- If gold is dropped, we just pick it up again.
- If a tree blocks the path, we just go around it.

We can understand the system at the "intentional" level.

- Agents "desire" to collect as much gold as possible.
- Agents "believe" they are close to the depot.

There is "some" sensible known **procedural knowledge** of the domain.

- It is "good" to avoid obstacles.
- If we see gold close, then go and collect it.
- If we bump into an unknown wall, then walk along it.

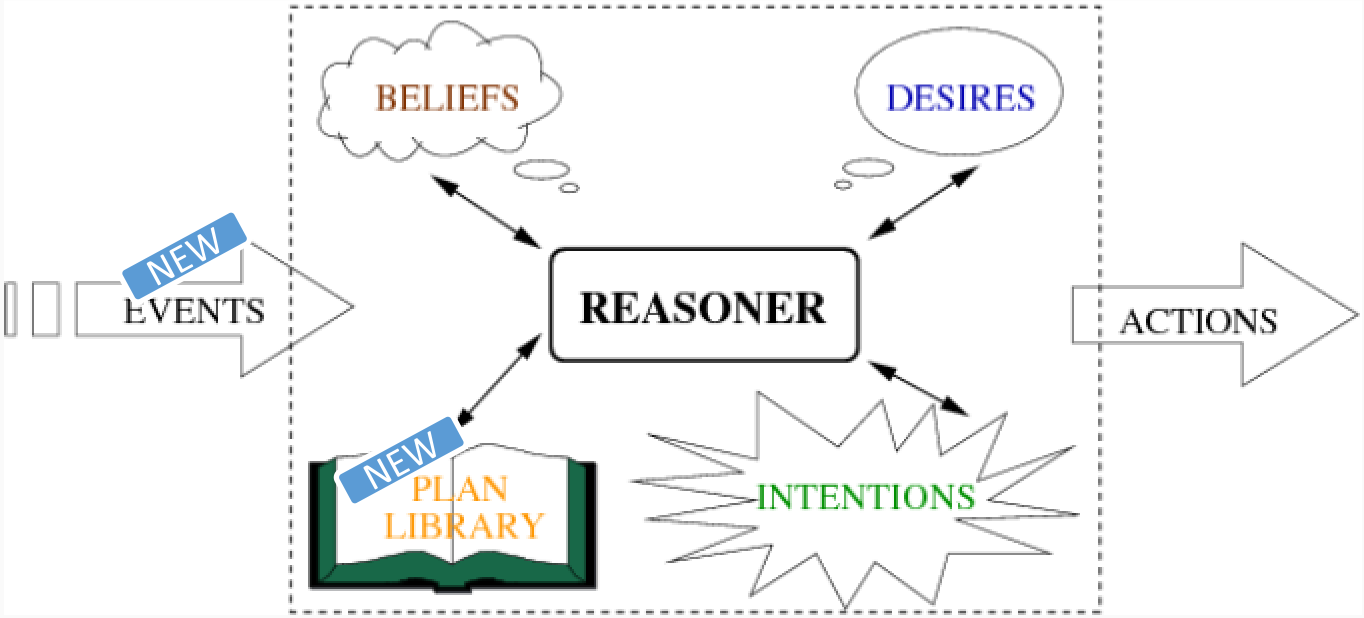
Key Features of BDI Agent-oriented Systems

Beliefs: information about the world.

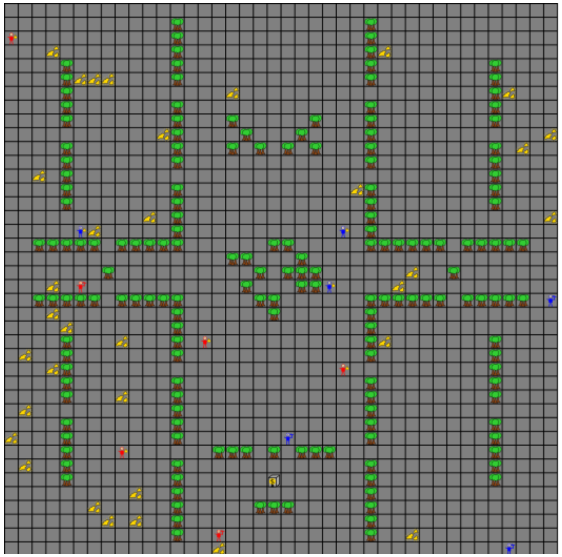
Events: goals/desires to resolve; internal or external.

Plan library: recipes for handling goals-events.

Intentions: partially uninstantiated programs with commitment.



Key Features of BDI Agent-oriented Systems (cont.)



In the gold-mining game:

Beliefs (static vs. dynamic):
- current location & location of depot.
- size of grid, # of gold pieces carrying, etc.

Events (internal vs. external):
- a gold piece is observed east;
- player3 communicates its location;
- the coordinator requests to explore the grid;
- we formed the internal goal to travel to loc(10, 22)

Plan library:
- if I see gold here & I am not full, collect it.
- if I hit an obstacle, go around it.
- if I don't know of any gold, explore grid.
- if I see gold around, move there and collect.

Intentions:
- I am currently traveling to the depot.
- I am informing my team-mates of new obstacles I find

Events & Plans

1. Events stand for the goals/desires/tasks to be achieved or resolved:
   - percepts: $goldAt(east)$, $goldDropped$, etc;
   - communication: $told(player3, loc(3, 2))$;
   - external request/goal: $achieve(explore\_grid)$;
   - internal sub-goal: $go\_to(loc(10, 22))$.

2. Plans stand for strategies useful to resolve (pending) events:
   - encode typical operational procedures in the domain;
   - non-deterministic;
   - event & context dependent;

$$e \; : \; \psi \longleftarrow P$$

P is a good strategy to resolve event e if context ψ is believed true.

Intentions

Agent's intentions are determined **dynamically** by the agent at **runtime** based on its known facts, current goals, and available plans.

An intention is just a **partially executed** strategy:

- comes from the plan library when resolving events.

An intention represent a **focus of attention**:

- something the agent is currently working on;
- actions/behavior arises as a consequence of executing intentions.

An agent may have **several intentions** active at one time.

- different simultaneous focuses of attention;

A **new intention** is created when an external event is addressed.

An intention may create/post an **internal event**:

- the intention will be updated when this event is addressed.

BDI Deliberation

Execution cycle: $\ldots \longrightarrow \mathcal{B} \longrightarrow \mathcal{D} \longrightarrow \mathcal{I} \longrightarrow \ldots$

1. collect new events
2. collect applicable plans (options w.r.t. events/goals)
3. select a plan π (deliberate and update intentions)
4. execute π
5. drop failed/unachievable plans

Time check!

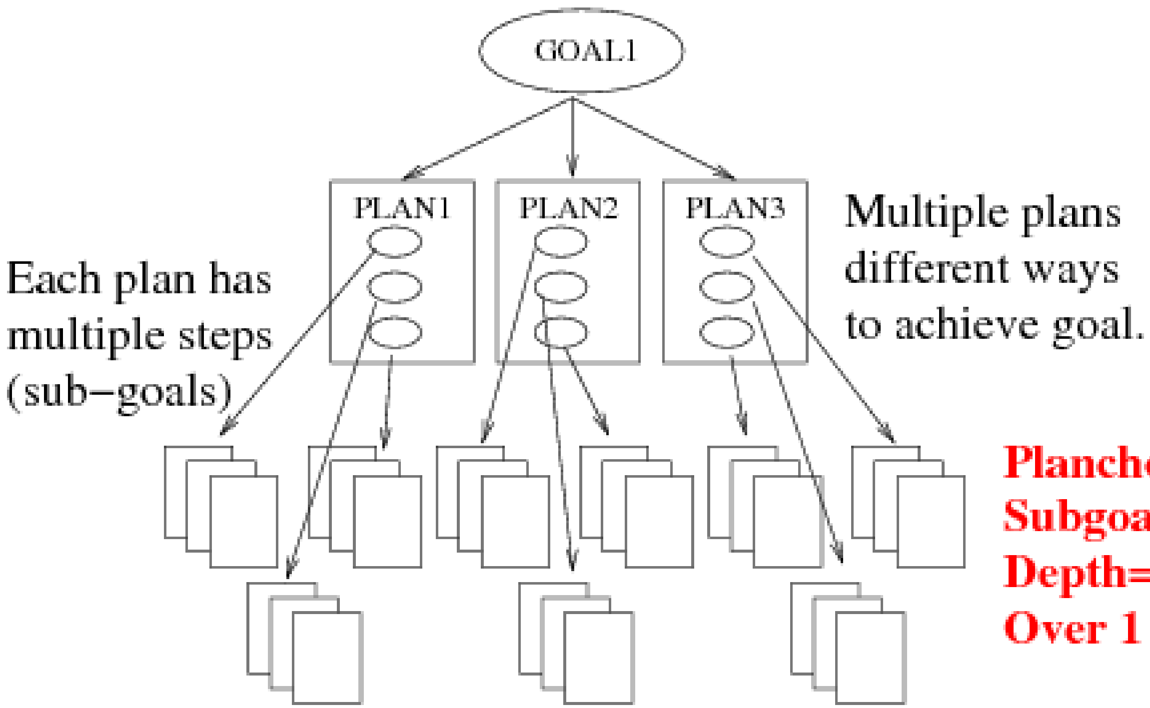
BDI Programming Guidelines

Key Points of BDI Programming

BDI Programming =
implicit goal-based programming + rational online executor

- Flexible and responsible to the environment: "reactive planning."
  - Well suited for soft real-time reasoning and control.
- Relies on context sensitive subgoal expansion: "act as you go."
- Leave for as late as possible the choice of which plans to commit to as the chosen course of action to achieve (sub)goals.
- Modular and incremental programming.
- Nondeterminism on choosing plans and bindings.

Possibility of Many Options



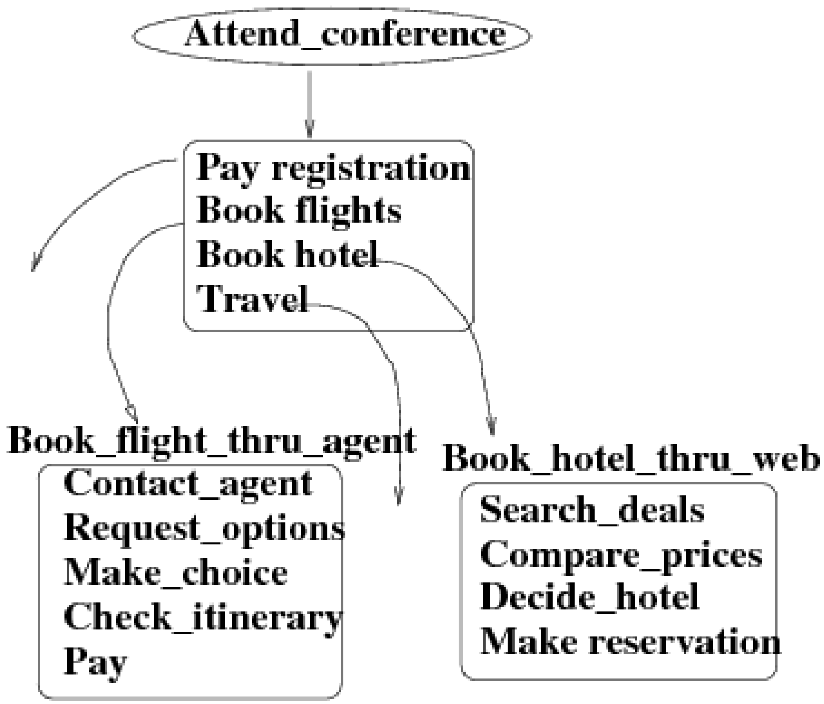
Planchoice =2
Subgoal steps=3
Depth=4
Over 1 million ways

Here we have 30 plans, 81 way to achieve the goal.
depends on choice of plans, number of steps, and
depth of tree:

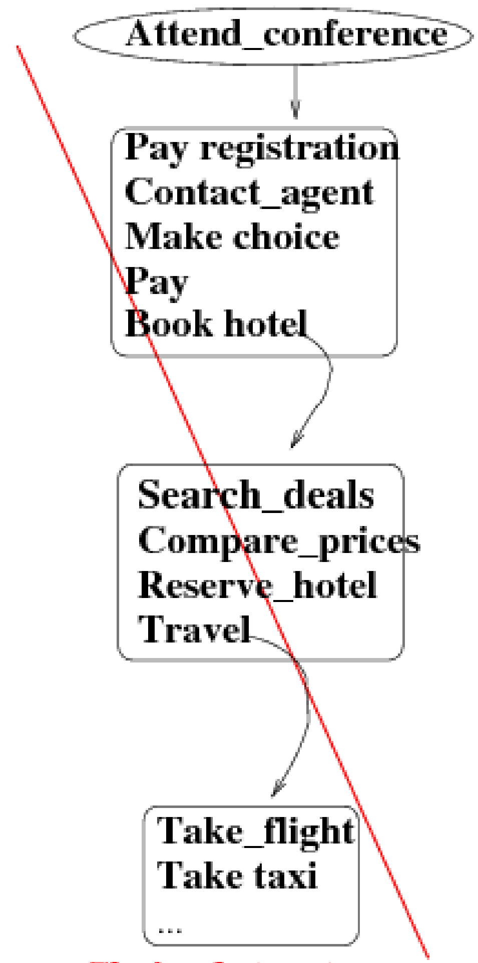
Making Use of the BDI Framework

1. Provide alternative plans where possible.
2. Break things down into subgoal steps.
3. Use subgoals and alternative plans rather than **if... then** in code.
4. Keep plans small and modular.
5. Plans are abstract modules - don’t chain them together like a flowchart.

Plan Structure



Hierarchical structure
– each plan is complete at its level of abstraction



Chained structure
– do some stuff then call next step...

~ HTN planning

Structuring Plans and Goals

Make each plan **complete** at a particular **abstraction level**.

- A high-level but complete plan for Attend Conference.

Use a **subgoal** - even if only one plan choice for now.

- Decouple a goal from its plans.

Modular and easy to **add other plan choices** later.

- Booking a flight can now be done with the Internet, if available!

Think in terms of **subgoals**, not function calls.

- What way-points do we need to achieve so as to realize a goal?

Learn to **pass information** between subgoals.

  - How are these way-points inter-related w.r.t. data?

BDI Programming: Is it a good idea?

Benefits:

- facilitates quick response to changes in the environment
- allows layering of the system
  - knowledge representation + reasoning vs. plan selection
- plans can be encoded in the design time
  - better control on what the system does (software engineering)

Shortcomings:

- (potentially) too much control of the system (is this still AI?)
- no (straightforward) guarrantees that the means leads to reaching the end (unlike planning!)
- plan selection is quite greedy → the system can get stuck

Summary

Agent design problems can be specified in terms of **task environments**.

There are different agent architectures with different capabilities and complexity.

BDI architecture is a practical way to implement **goal-oriented agents.**

- based on the theory of **practical reasoning** that human appear to use in daily lives.
- programming using **mentalistic concepts** of beliefs, desires and intentions.

Related reading:

- Russel and Norvig: Artificial Intelligence: A Modern Approach – Chapter 2
- Wooldrige: An Introduction to Multiagent Systems – Chapters 1 and 2
- BDI lecture notes (Tambe/Greenstadt)

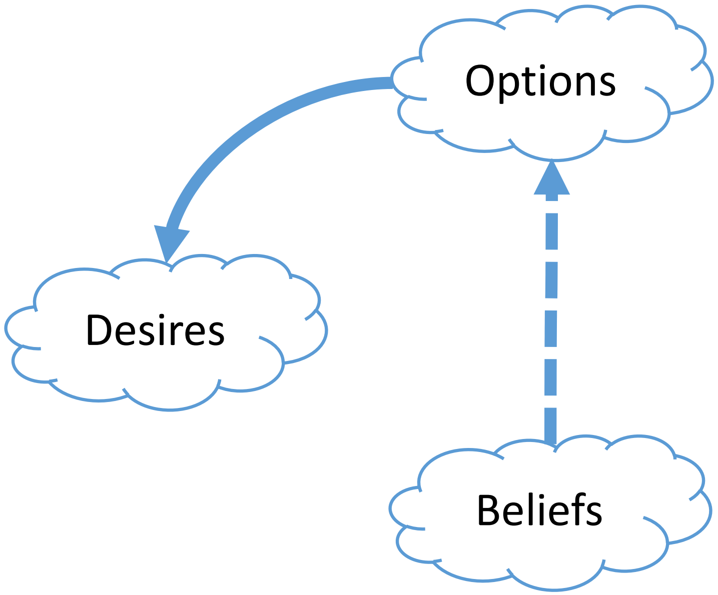
Agent-Oriented Programming

1. Philosophers have produced theories of (human) rational action:
   - Folk psychology.
   - Practical reasoning.
   - Intentional systems.
2. Theorists have taken this and developed theories to represent the properties of agents (humans or not).
   - Relation between mental attitudes.
   - Commitment.
   - Rational architectures.

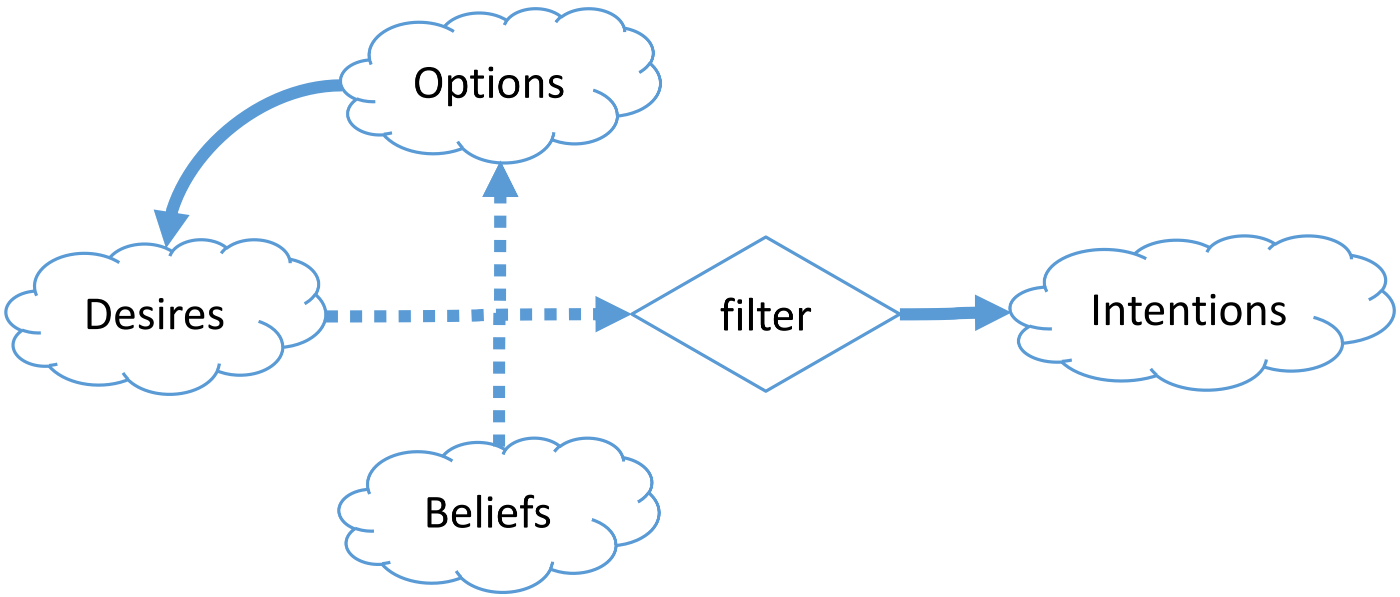
So, why not directly program agents in terms of the mentalistic, intentional notions?

∴ we will study one agent-oriented approach: BDI-style Programming.

Deliberation



1. Begin by trying to understand what the options available to you are:
   - options available are desires.



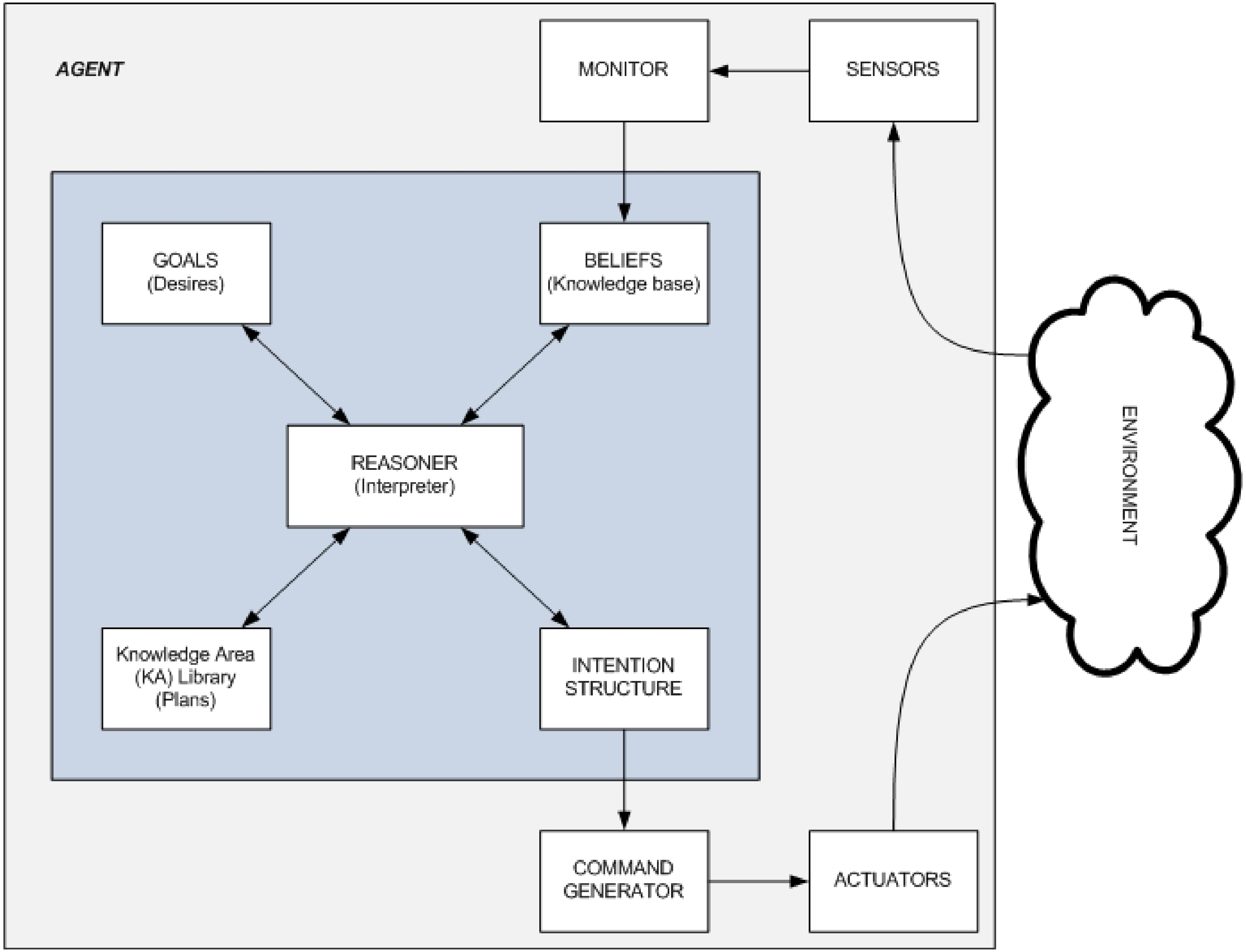
2 Choose between them, and commit to some:

- chosen options are then intentions.

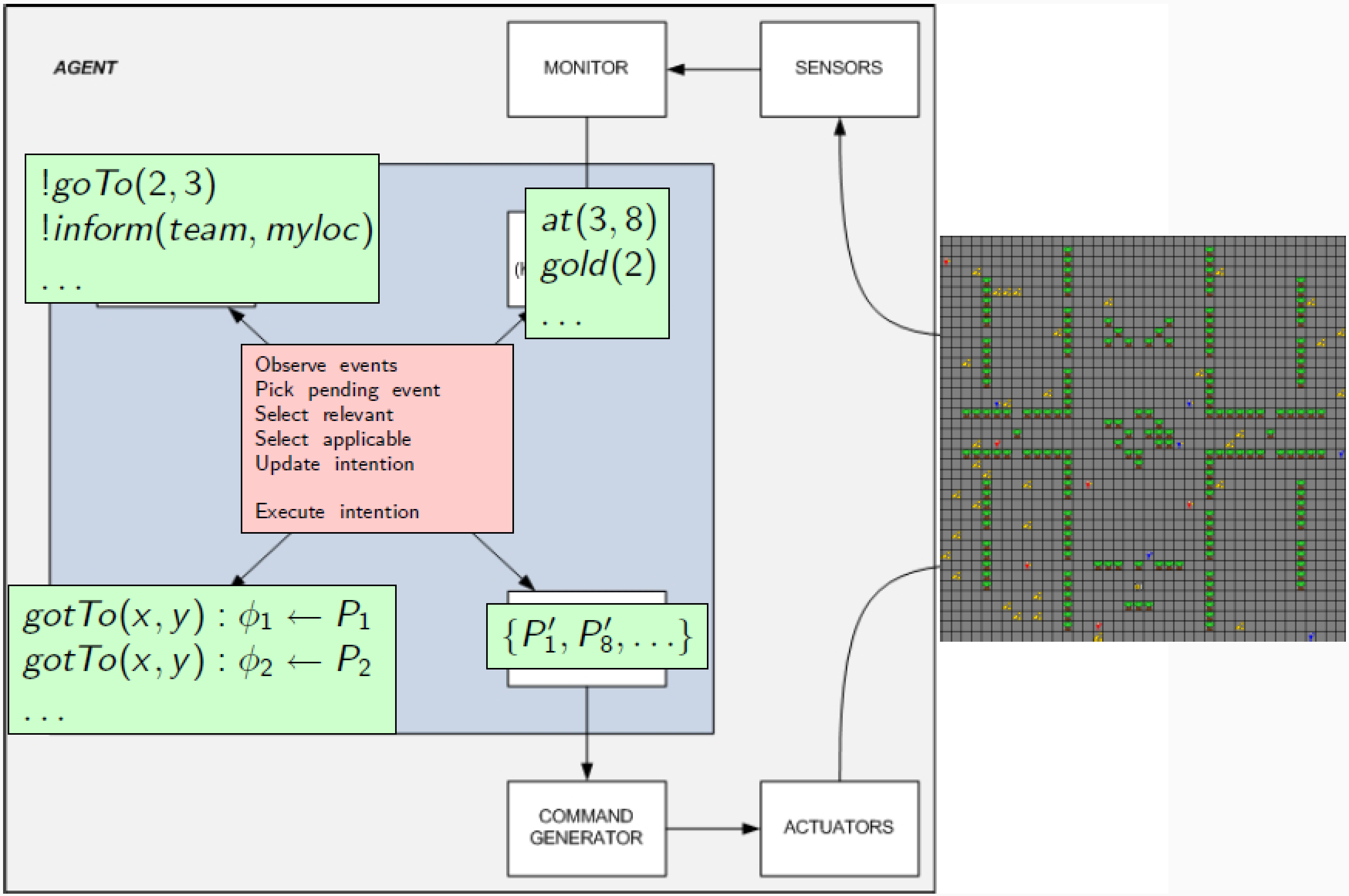
Practical Reasoning: Summary

1. Intentional stance (Daniel Dennett (1987))
   - High-level abstraction of behavior at the level of minds.
   - Rational behavior can be understood in terms of mental properties:
     - beliefs, desires, goals;
     - fear, hopes, etc.
2. Practical reasoning (Michael Bratman (1990))
   - Reasoning for acting: the process of figuring out what to do.
   - Two activities: deliberation and means-end analysis.
3. Commitments (on goals/intentions & plans)
   - fanatical, single-minded, open-minded.
4. Agent architectures
   - IRMA & PRS.
   - Built around: beliefs, desires, plan libraries, intentions, filter, etc.

Typical BDI-style



Typical BDI-style System



Events and Plans

1. Events stand for the goals/desires/tasks to be achieved or resolved:
   - percepts: $goldAt(east)$, $goldDropped$, etc;
   - communication: $told(player3, loc(3,2))$;
   - external request/goal: $achieve(explore\_grid)$;
   - internal sub-goal: $go\_to(loc(10,22))$.

2. Plans stand for strategies useful to resolve (pending) events:
   - encode typical operational procedures in the domain;
   - non-deterministic;
   - event & context dependent;

$$e \; : \; \psi \longleftarrow P$$

P is a good strategy to resolve event e if context ψ is believed true.

Plans in PRS: Clearing a Block

```
Plan: {
NAME: "Clear a block"
GOAL:
    ACHIEVE CLEAR $OBJ;
CONTEXT:
    FACT ON $OBJ2 $OBJ;
BODY:
    EXECUTE print "Clearing " $OBJ2 " from on top of " $OBJ "\n";
    EXECUTE print "Moving " $OBJ2 " to table.\n";
    ACHIEVE ON $OBJ2 "Table";
EFFECTS:
    EXECUTE print "CLEAR: Retracting ON " $OBJ2 " " $OBJ "\n";
    RETRACT ON $OBJ1 $OBJ;
FAILURE:
    EXECUTE print "\n\nClearing block " $OBJ " failed!\n\n";
}
```

Cognitive agents

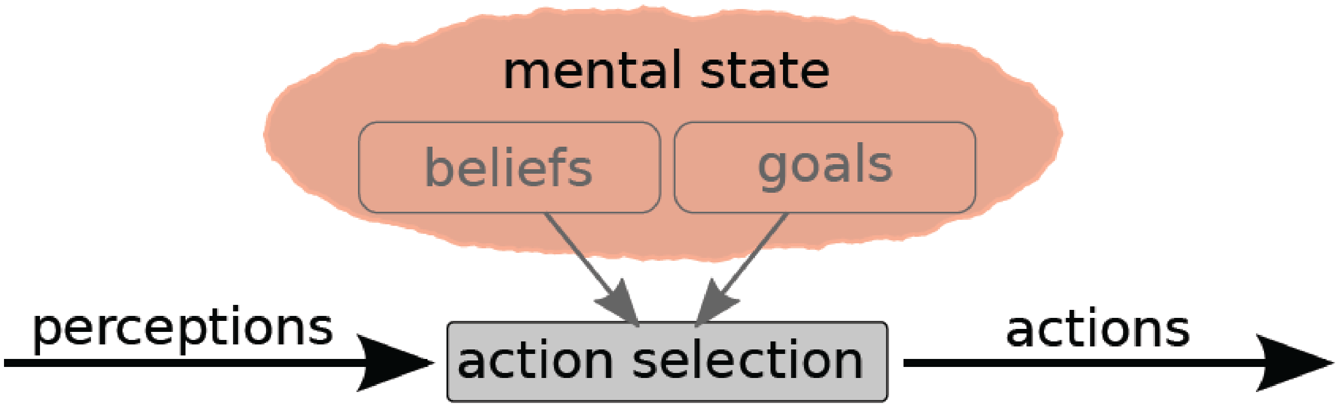
cognitive/knowledge intensive agent

employ cognitive processes, such as knowledge representation and reasoning as the basis for decision making and action selection. I.e., they construct and maintain a *mental state*.

mental state

agent's internal explicit representation of the environment, itself, its peers, etc. $\rightsquigarrow$ agent's memory

Cognitive agents (cont.)



beliefs a database of agent's information about itself, the world (environment), other agents, etc.
⇝ NOW

goals description of states the agent "wants" to bring about
⇝ FUTURE

How to select actions leading from NOW to the FUTURE ?

The Problem (with planning)

How to select actions leading from NOW to the FUTURE ?

plan - execute - monitor cycle

1. plan from the current state to a goal state(s)
2. sequentially execute actions from the plan
3. monitor success of action execution
   - in the case of action failure, (re-)plan again (goto 1)

speed of planning vs. environment dynamics

planning $\overset{speed}{\succ}$ environment can perform relatively well

planning $\overset{speed}{\prec}$ environment can lead to fatal inefficiencies

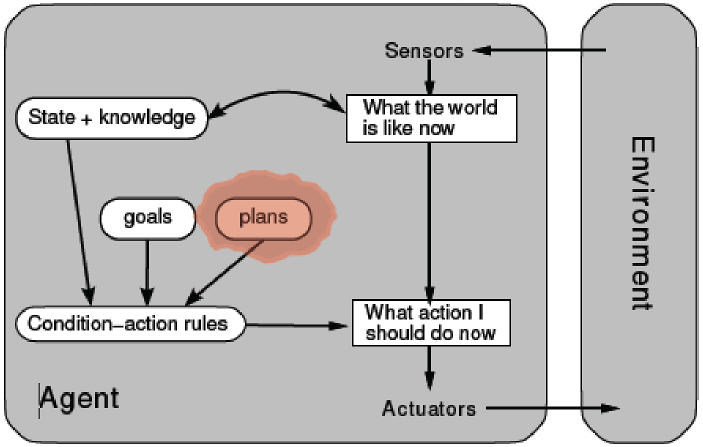
$\rightsquigarrow$ the system "suffocates" in (re-)planning

The Idea: Reactive Planning

reactive planning

Instead of plan-execute-monitor cycle, select partial plans reactively on the ground of the current state of the world.

Reactive planning agent:



current beliefs + future goals $\rightsquigarrow$ choose from a plan library

Belief-Desire-Intention

Belief-Desire-Intention (BDI)

⇝ a reactive planning agent architecture. Establishes intentions as a first-class concept.

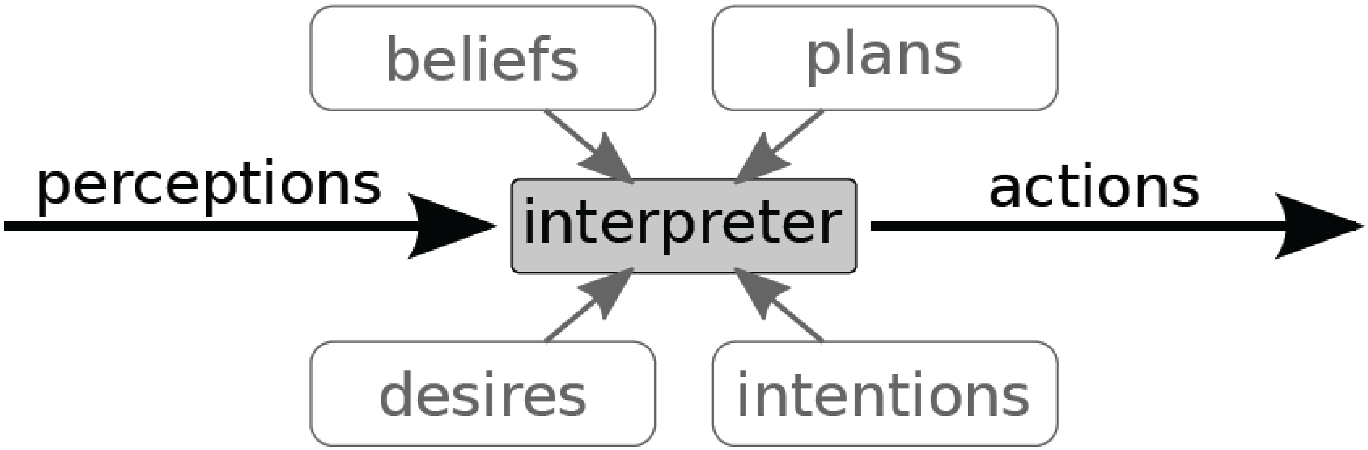
structural decomposition:

- ***(B)eliefs:*** reflect agent's *static* beliefs about its environment, itself, its peers, etc. (*now*)
- ***(D)esires:*** descriptions of situations the agent wants to bring about (*future*)

system dynamics: ⇝ from *now* to the *future*

- ***(I)ntentions:*** courses of action, plans, the agent commits to
  - *partial plans of action that the agent is committed to execute in order to fulfill the goals*

Means-end reasoning



explicit partial plans ⇝ *recipes* for how to proceed from now to the future

means-end reasoning ⇝ to reach an end, there are means to employ...

- plans (recipes) are selected from a pre-encoded plan library

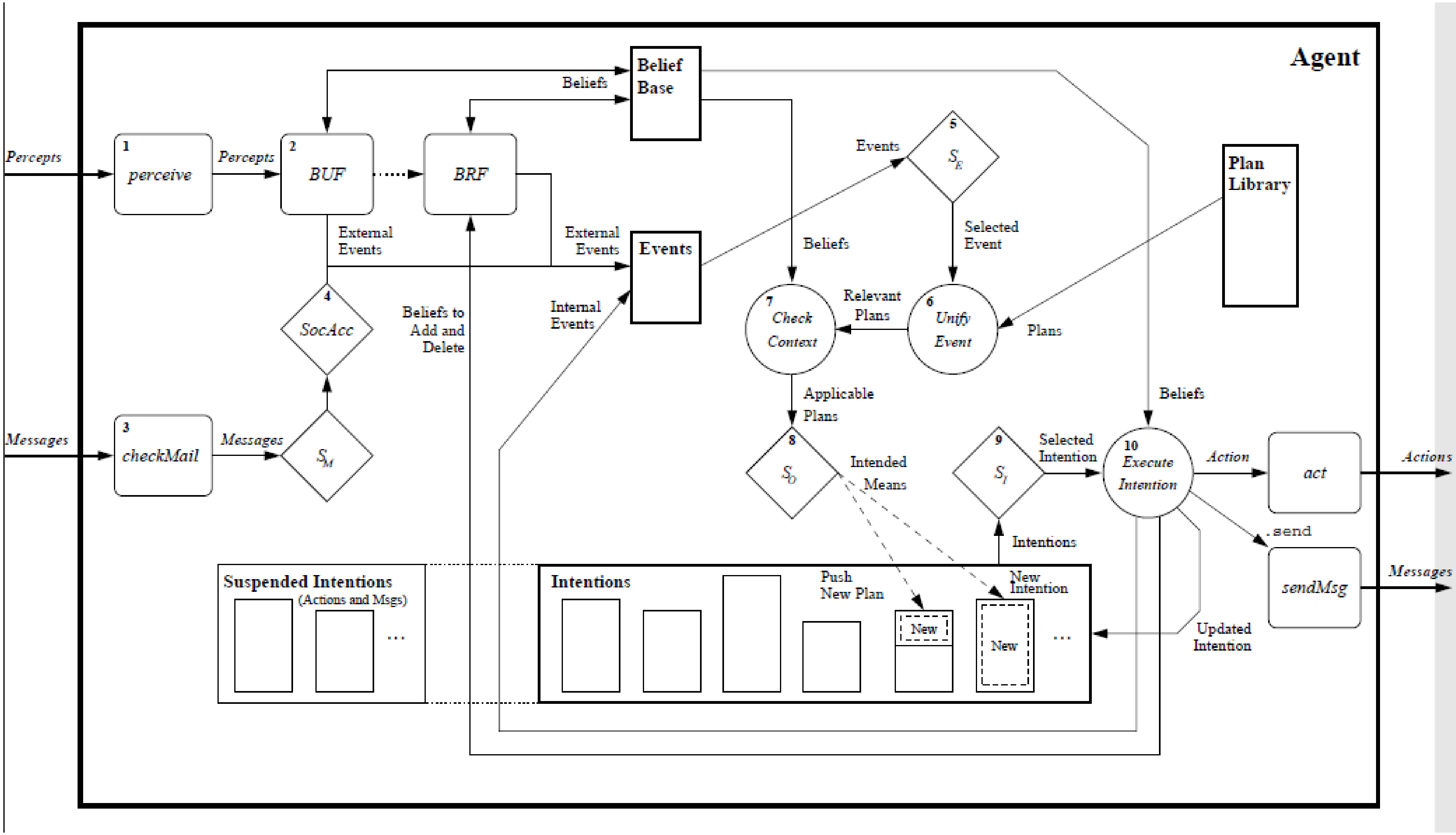
IRMA: Intelligent Resource-bounded Machine Architecture

IRMA has four key symbolic data structures:

1. a plan library: normal operations;
2. beliefs: information available to the agent — may be represented symbolically, but may be as simple as PASCAL variables;
3. desires: those things the agent would like to make true — think of desires as tasks that the agent has been allocated; in humans, not necessarily logically consistent, but our agents will be! (goals);
4. intentions: desires that the agent has chosen and committed to.

+ following slides from Sardina_lect06

JASON Rreasoning Cycle



Continue: from “"Bordini_chapter13_handout_Programming MAS.pdf"