COMP474/6741

Intelligent Agent Systems and Architectures

Poole: Chap.10

Some slides are part of my Research work, 2010

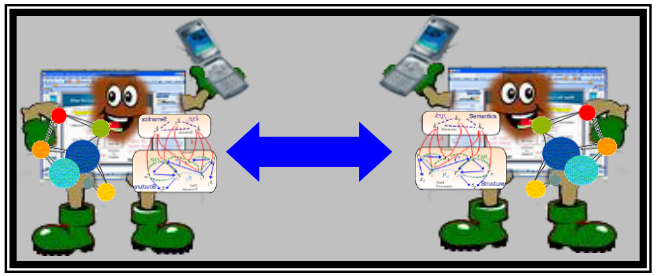
Today

- Web of knowledge
- Why Agent?
- What is an agent
- Features of Intelligent Agent
- Types of Agents
  - Logic-based agents
  - Reactive agents
  - Belief-desire-intention agents
  - Layered architectures
- Multiagent Systems

Web of Knowledge (Semantic Web, Web 3.0)

Knowledge and Data Collections

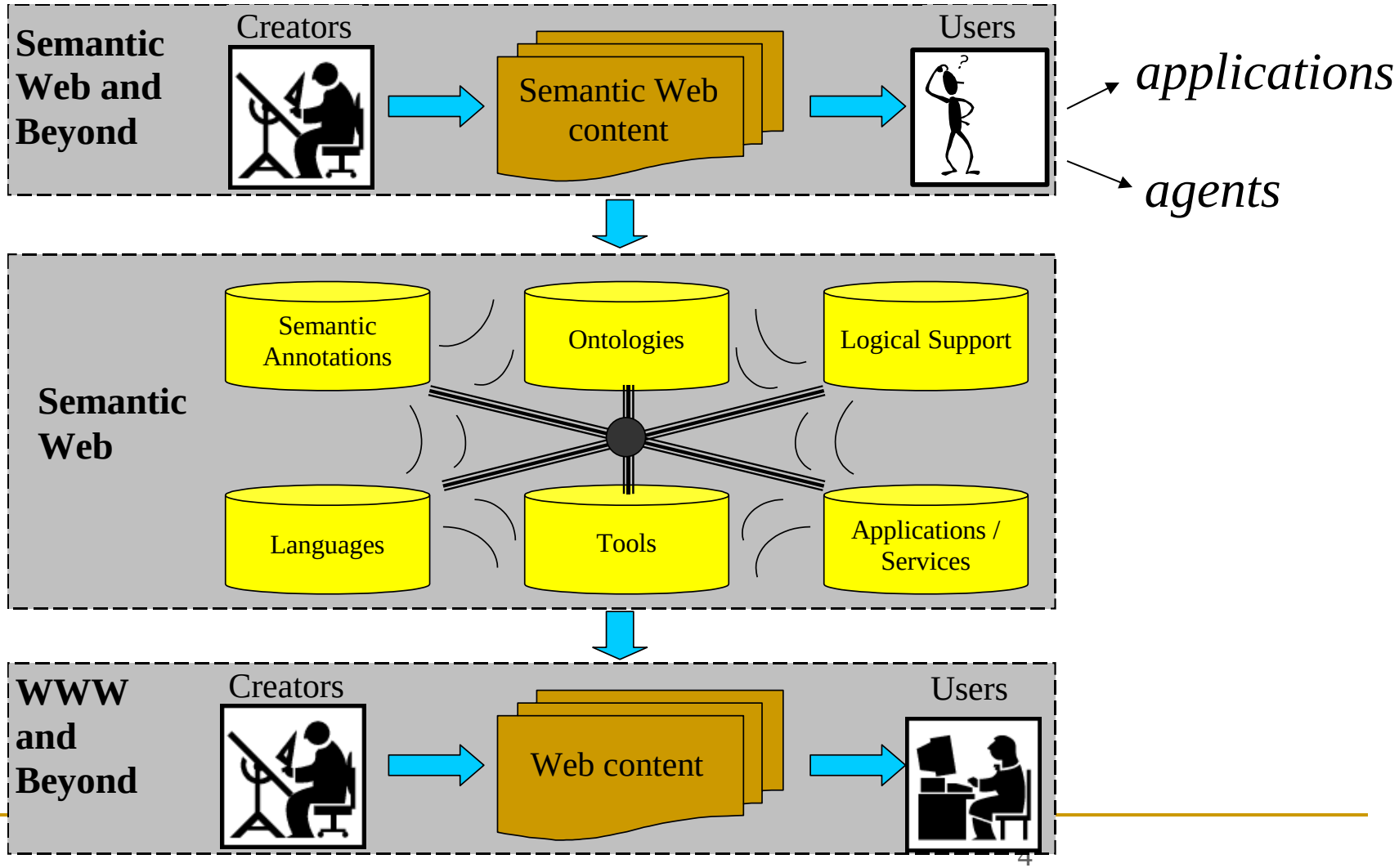
Facilitates Knowledge-to-Knowledge interaction



Web of Knowledge
Ontologies
Metadata
Reasoning
Semantic Technology
Integration

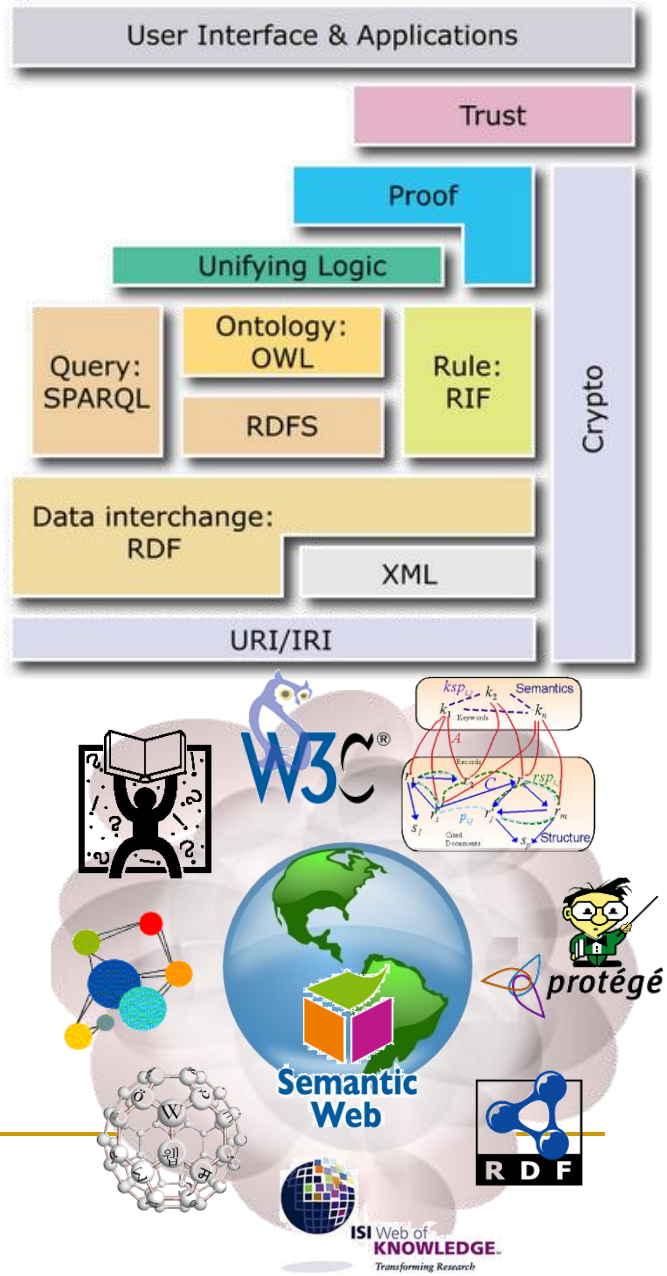
KaaS: Knowledge-as-a-Service
KaaU: Knowledge-as-a-User

Semantic Web: New "Users"



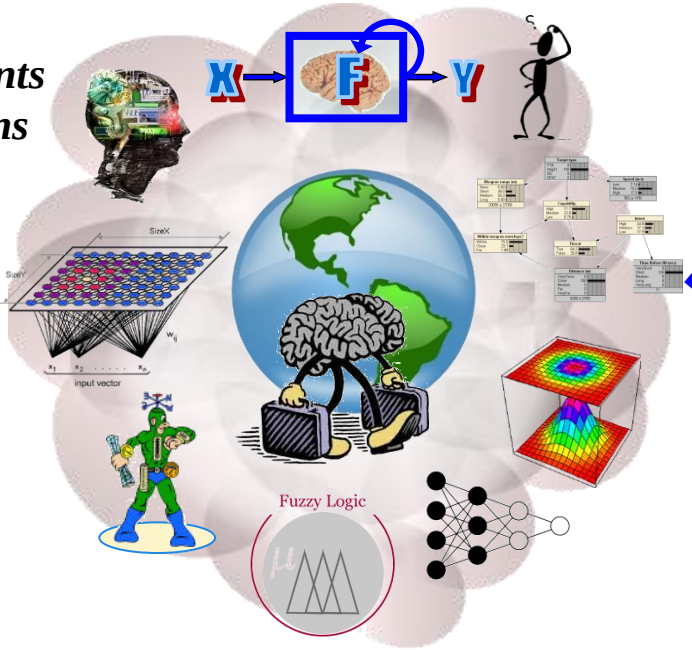
What is Semantic Web ?

- *The **Semantic Web** is an evolving development of the World Wide Web in which the meaning (semantics) of information and services published on the Web and their inter-relationships are explicitly defined, making it possible for the Web-based software tools, agents, applications and systems to discover, extract and "understand" Web information resources and capabilities and automatically utilize it.*
- ***Semantic Technologies** are designed to standardize and support interoperability and integration of information content and capabilities (services) of Web-based systems and components at local and global scale.*
- *As a **software technology**, semantic technology encodes meanings separately from data and from application code to enable machines to understand, share and reason with them at execution time.*



Web of Intelligence (Distributed AI, Web 4.0)

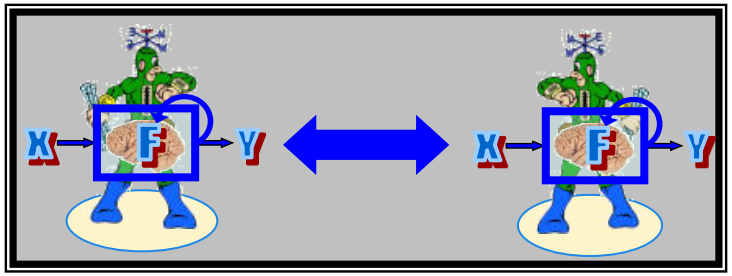
Intelligent Agents and Applications



Facilitates Intelligence-to-Intelligence interaction

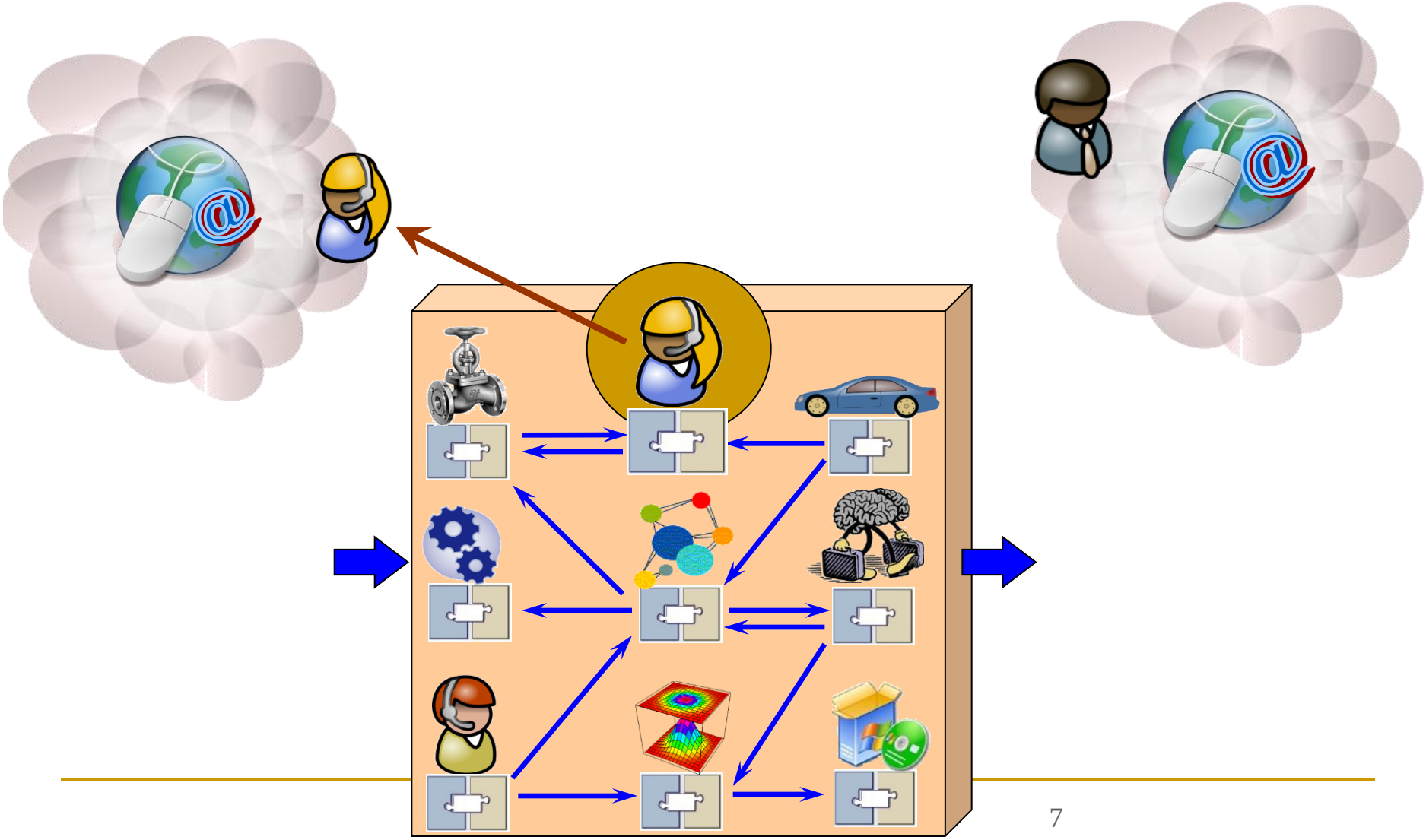
Web of intelligent entities (intelligence services), browseable, searchable, composable, self-managed, dynamic, mobile …

Web of Intelligence
Agents and MAS
Data and Web Mining
Machine Learning
Self-Management
Context-Awareness

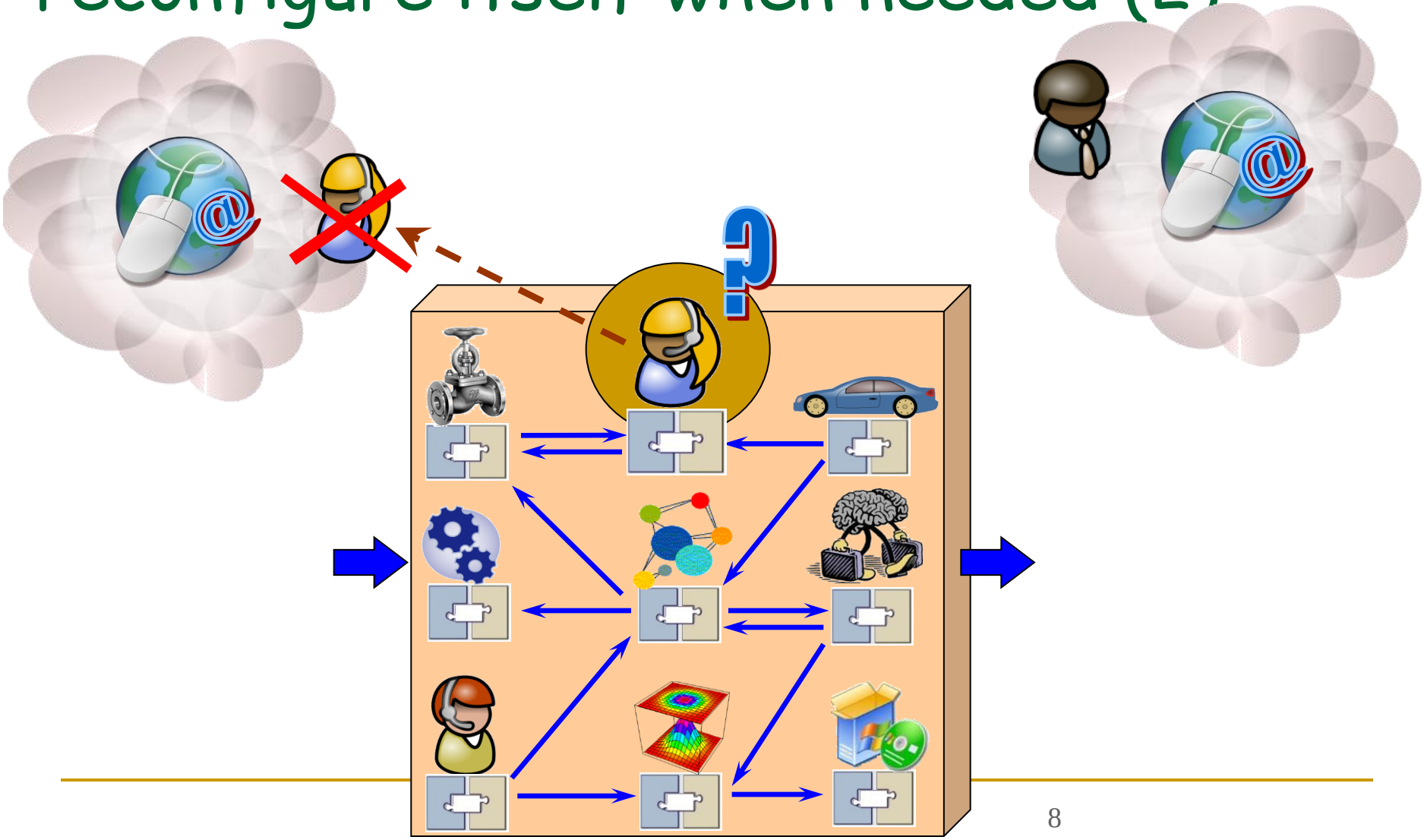


INTaaS: Intelligence-as-a-Service
INTaaU: Intelligence-as-a-User

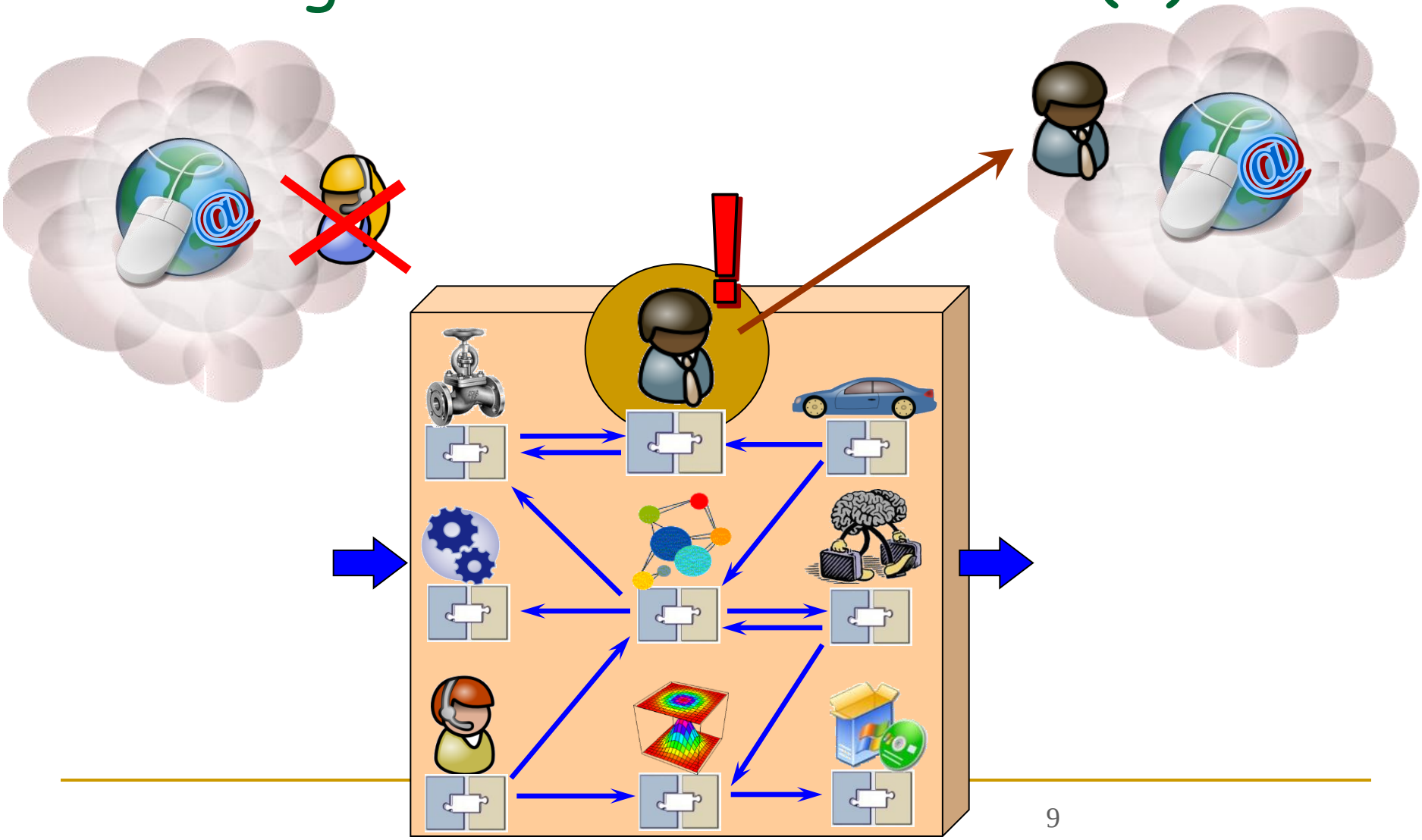
A system should be open and ready to reconfigure itself when needed (1)



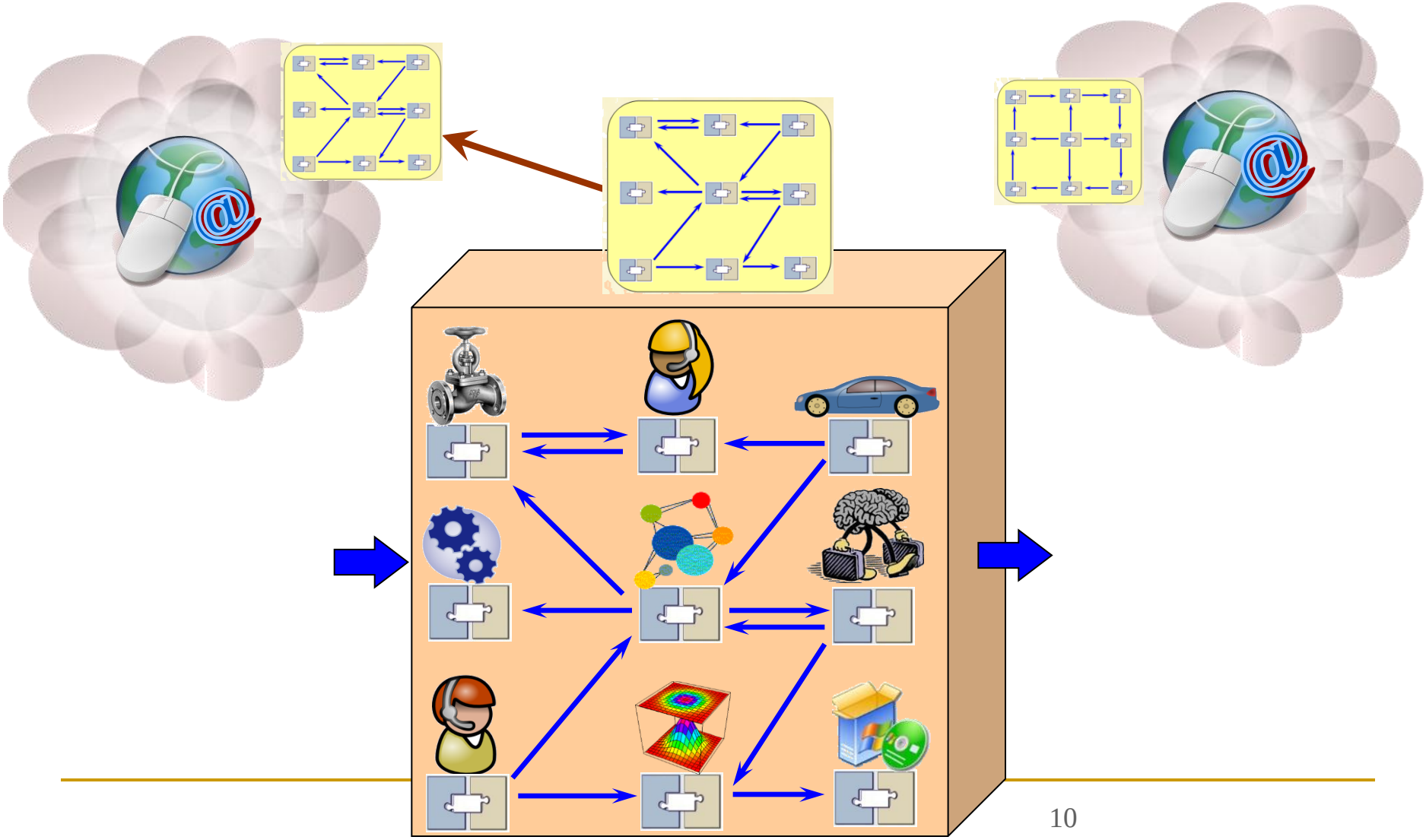
A system should be open and ready to reconfigure itself when needed (2)



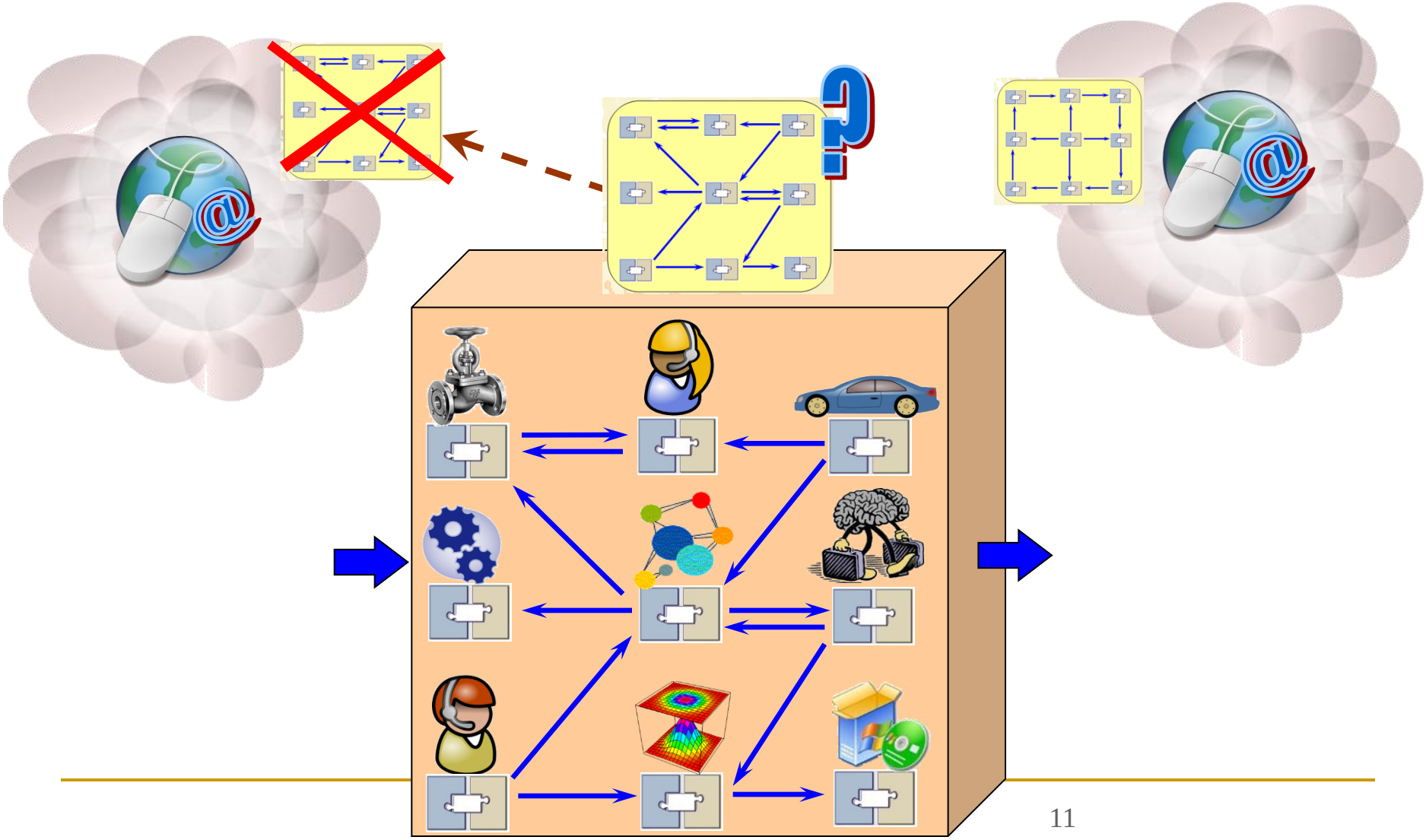
A system should be open and ready to reconfigure itself when needed (3)



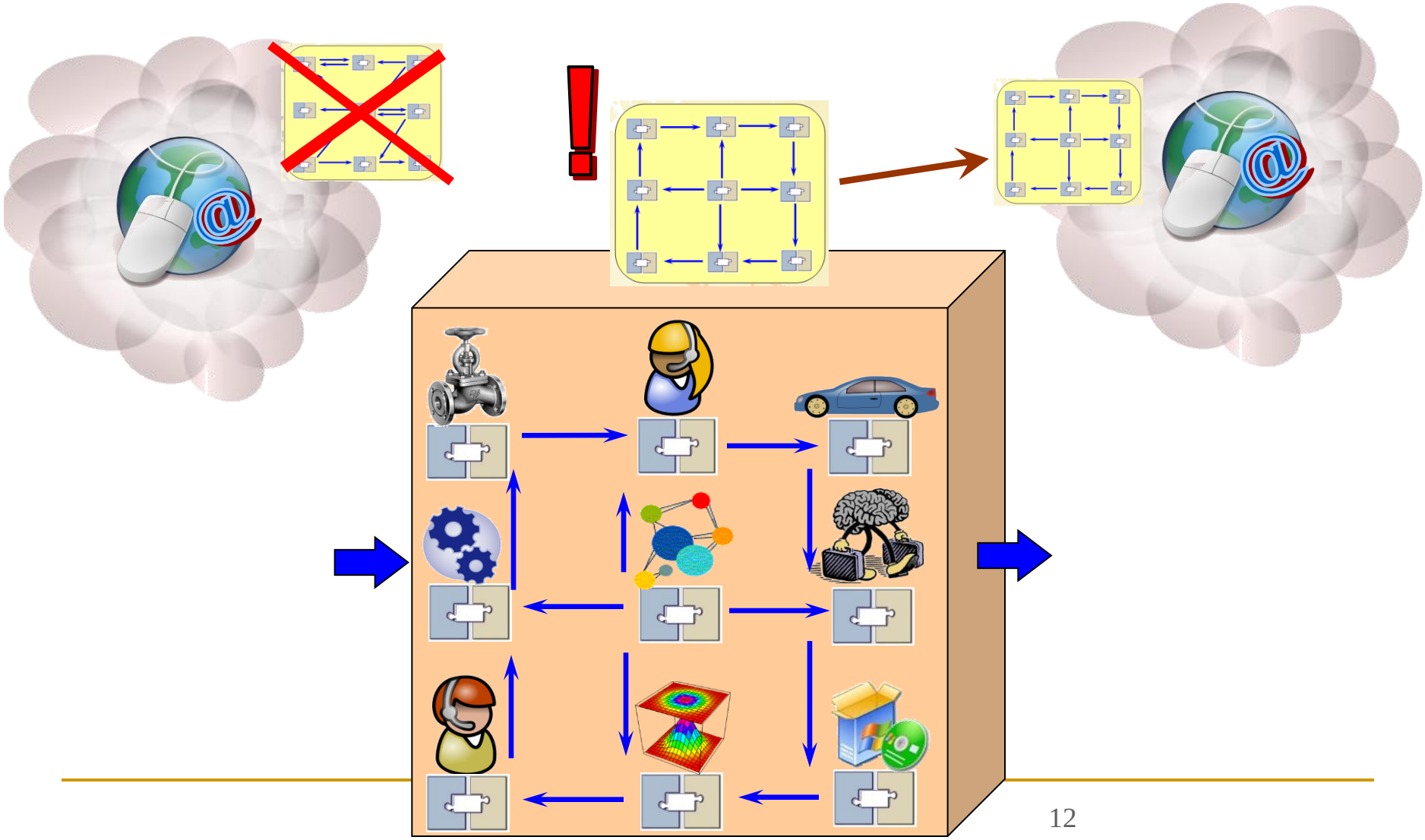
Even a business logic of a system can be imported and reconfigured on-the-fly (1)



Even a business logic of a system can be imported and reconfigured on-the-fly (2)

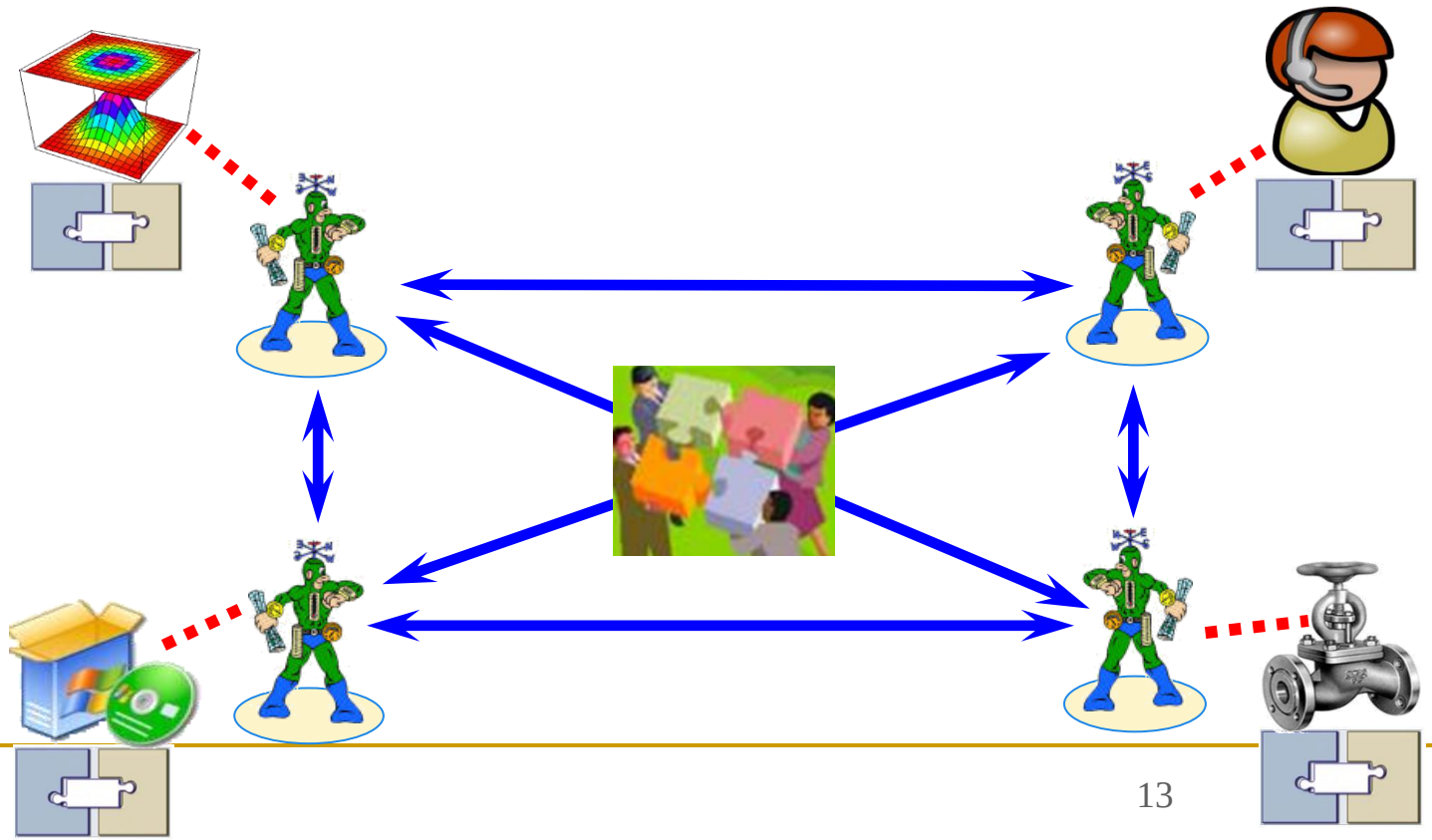


Even a business logic of a system can be imported and reconfigured on-the-fly (3)



Agents are able to help !

Adding a ***"virtual representative"*** for every resource solves the global interoperability problem. Intelligent **agent** (a kind of "software robot") will act, communicate and collaborate on behalf of each Web resource



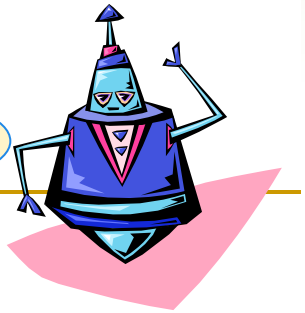
Today

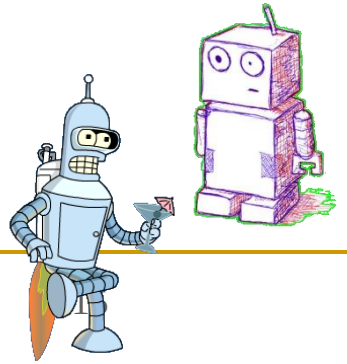
- Web of knowle
- Why Agent?
- What is an agent
- Features of Intelligent Agent
- Types of Agents
  - Logic-based agents
  - Reactive agents
  - Belief-desire-intention agents
  - Layered architectures
- Multiagent Systems

Why Agents ?

- Growing complexity of computer systems and networks used in industry → need for new approaches to manage and control them
- IBM vision: Autonomic computing – *Self-Management* (includes self-configuration, self-optimization, self-protection, self-healing)
- Ubiquitous computing, "Internet of Things" → huge numbers of heterogeneous devices are interconnected
  - "nightmare of pervasive computing" when almost impossible to centrally *manage* the complexity of interactions, neither even to *anticipate* and *design* it.
- We believe that self-manageability of a complex system requires its components to be autonomous, i.e. be realised as *agents*.
- Agent-based approach to SE is also considered to be facilitating the design of complex systems








Today

- Web of knowledge
- Why Agent?
- What is an agent

- Features of Intelligent Agent
- Types of Agents
  - Logic-based agents
  - Reactive agents
  - Belief-desire-intention agents
  - Layered architectures
- Multiagent Systems

What is an agent?

- "An over-used term" (Patti Maes, MIT Labs, 1996)
- Many different definitions exist …
- Who is right?
- Let's consider 9 complementary ones …

Agent Definition (1)

I can relax, my agents will do all the jobs on my behalf

American Heritage Dictionary:

agent -

" … one that acts or has the power or authority to act… or represent another"

Agent Definition (2) [IBM]

- *"...agents are software entities that carry out some set of operations on behalf of a user or **another program** ..."* [IBM]

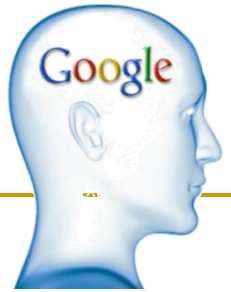
I can relax, my agents will do all the jobs on my behalf

Potentially agents may have "Everything-as-a-User" !

Agent Definition (3)

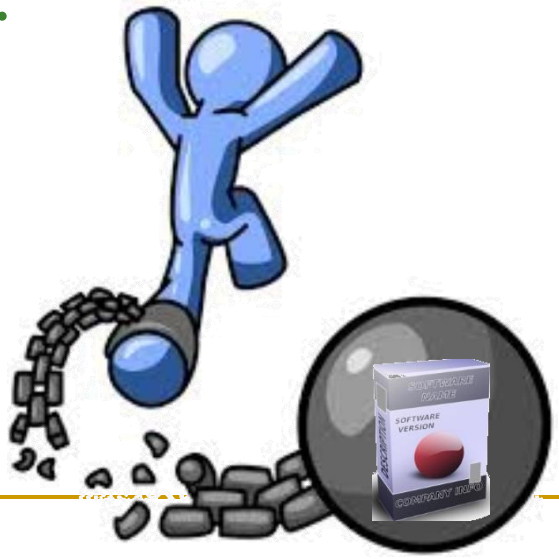
Posted by
Margaret Rouse
Whatls.com

On the Internet, an intelligent agent (or simply an agent) is a program that gathers information or performs some other service without your immediate presence and on some regular schedule.



Agent Definition (4)
[FIPA: (Foundation for Intelligent Physical Agents), www.fipa.org]

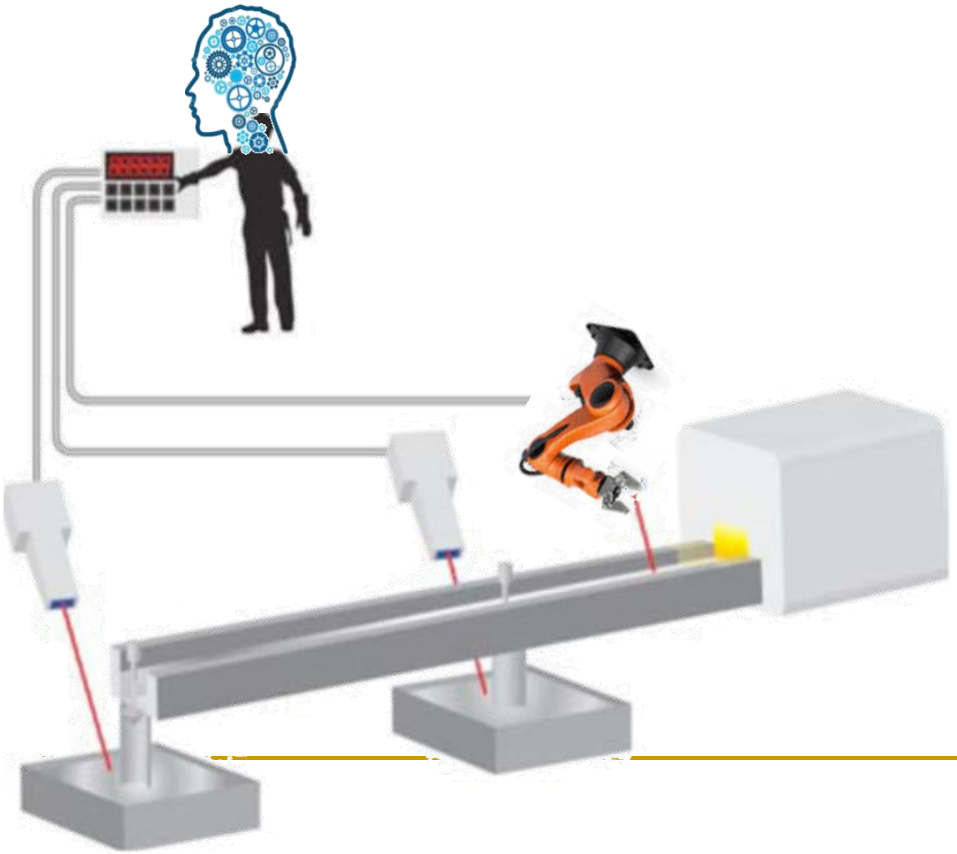
- An agent is a computational process that implements the ***autonomous… functionality of an application***.



Agent Definition (5)

- *"An agent is anything that can be viewed as **perceiving** its **environment** through sensors and **acting** upon that **environment** through effectors."*

Russell & Norvig



Agent Definition (6)

- *"…agents are computational systems that inhabit some **complex dynamic environment**, sense and **act autonomously** in this environment, and by doing so realize a set of **goals or tasks** for which they are designed."*

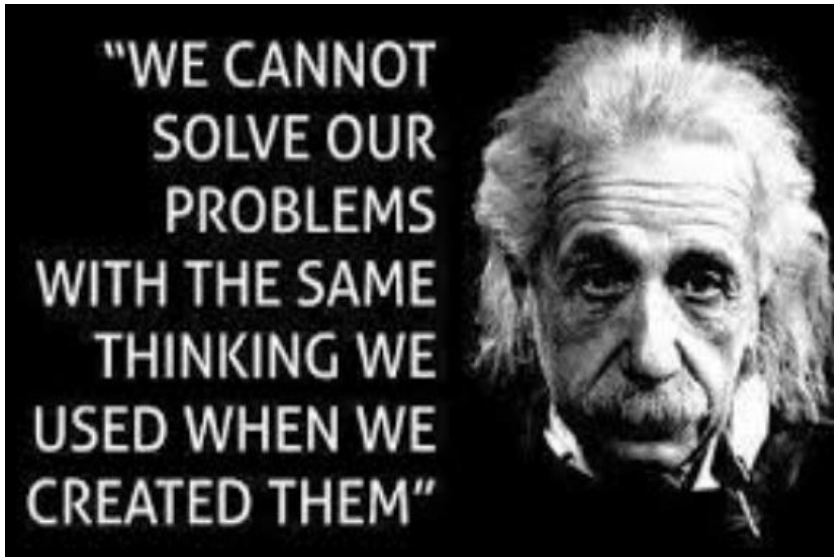
Pattie Maes

Agent Definition (7)

- "… An agent is anything that is capable of acting upon information it perceives. An intelligent agent is an agent capable of ***making decisions*** about how it acts ***based on experience.*** "

F. Mills & R.

CCSI Consortium on Cognitive Science Instruction



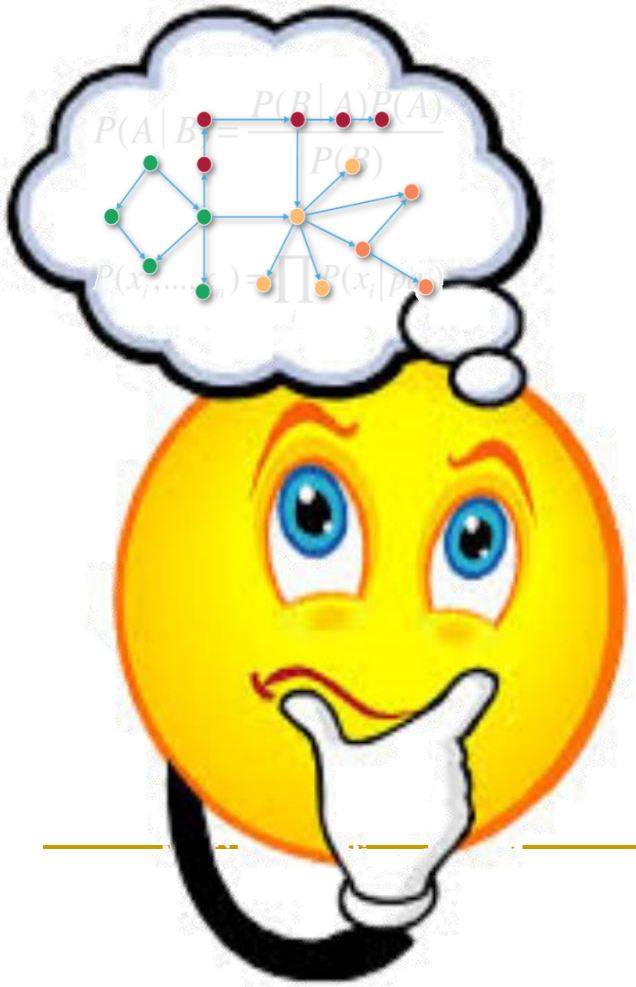
"Good decisions come from experience. Experience comes from making bad decisions."

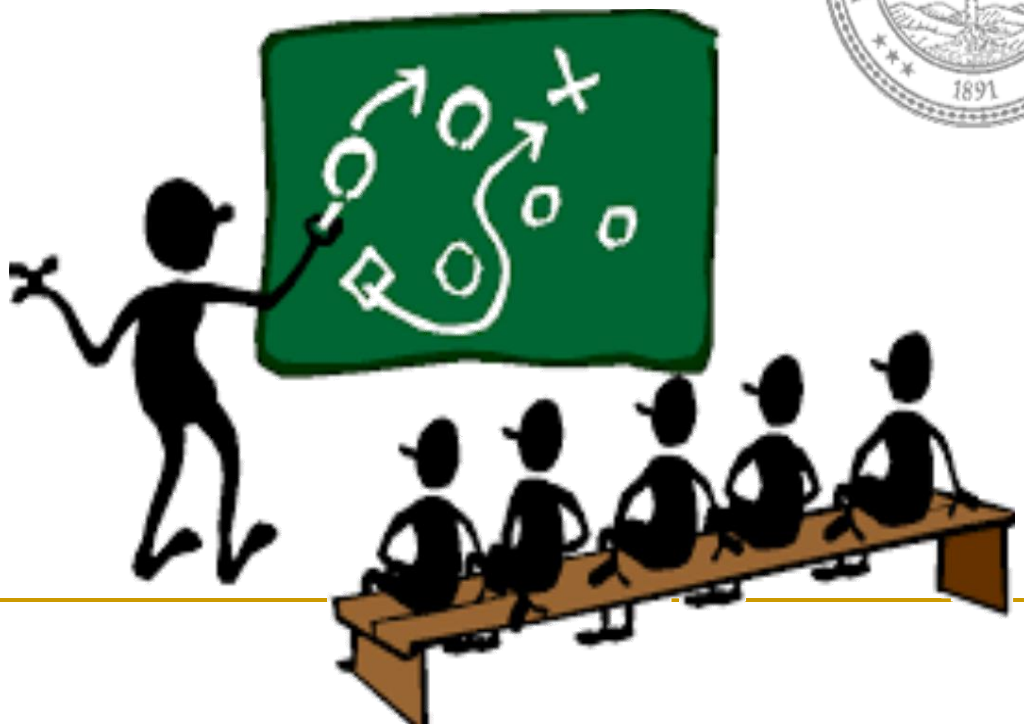
- Mark Twain (1835 - 1910), American Novelist and Journalist

Agent Definition (8)

- *"Intelligent agents continuously perform … **reasoning** to interpret perceptions, solve problems, draw inferences, and determine actions."*

Barbara Hayes-Roth





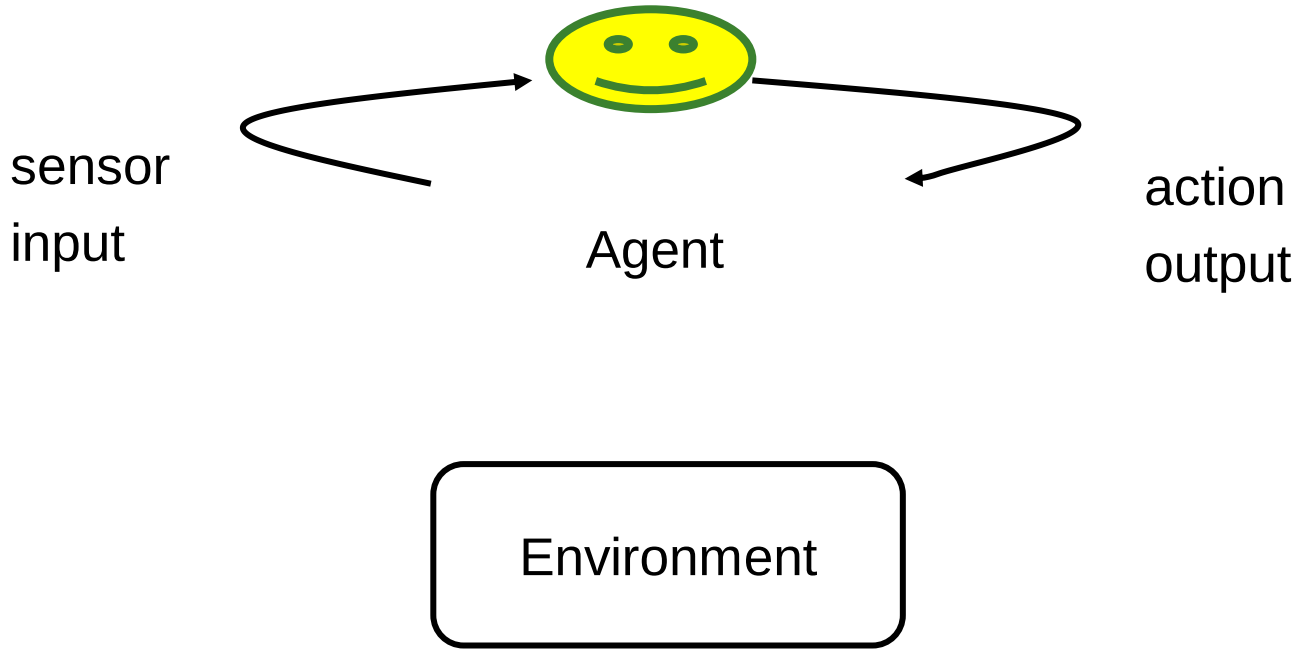
Agent Definition (9)

- An agent is an entity which is: … ***proactive***: … should not simply act in response to their environment, … should be able to exhibit ***opportunistic, goal-directed behavior*** and take the initiative when appropriate; … ***social:*** … should be able to interact with humans or other artificial agents …

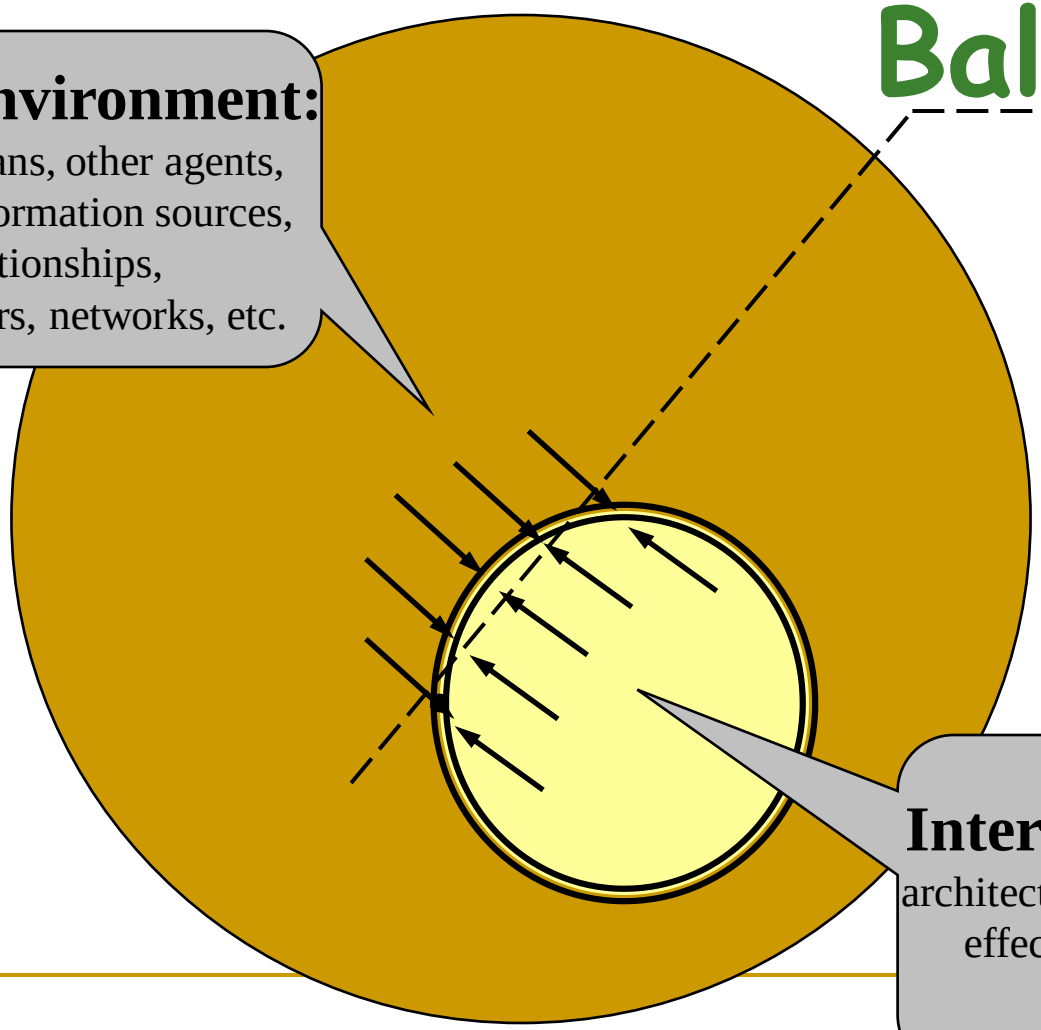
"A Roadmap of agent research and development", N. Jennings, K. Sycara, M. Wooldridge (1998)

Agents & Environments

- The agent takes sensory input from its environment, and produces as output actions that affect it.



Internal and External Environment of an Agent



External Environment:
user, other humans, other agents, applications, information sources, their relationships, platforms, servers, networks, etc.

Balance !

Internal Environment:
architecture, goals, abilities, sensors, effectors, profile, knowledge, beliefs, etc.

What "Balance" means?

For example a balance would mean: …

… for a human – possibility to complete the personal mission statement;

… for an agent – possibility to complete its design objectives.

Three groups of agents [Etzioni and Daniel S. Weld, 1995]

- **Backseat driver:** helps the user during some task (e.g., Microsoft Office Assistant);
- **Taxi driver:** knows where to go when you tell the destination;
- **Concierge:** know where to go, when and why.

What *intelligent* agents are ?

- "An intelligent agent is one that is capable of flexible ***autonomous*** action in order to meet its design objectives, where flexible means three things:
  - ***reactivity***: *agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it* in order to satisfy its design objectives;
  - ***pro-activeness***: *intelligent agents are able to exhibit goal-directed behavior by taking the initiative* in order to satisfy its design objectives;
  - ***social ability***: *intelligent agents are capable of interacting with other agents (and possibly humans)* in order to satisfy its design objectives";

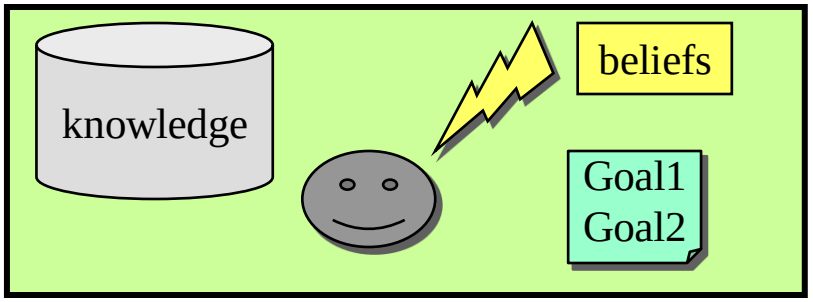
Wooldridge & Jennings

Today

- Web of knowledge
- Why Agent?
- What is an agent
- Features of Intelligent Agent
- Types of Agents
  - Logic-based agents
  - Reactive agents
  - Belief-desire-intention agents
  - Layered architectures
- Multiagent Systems

Agent Characterisation

- An agent is responsible for satisfying specific *goals*. There can be different types of goals such as achieving a specific status (defined either exactly or approximately), keeping certain status, optimizing a given function (e.g., utility), etc.

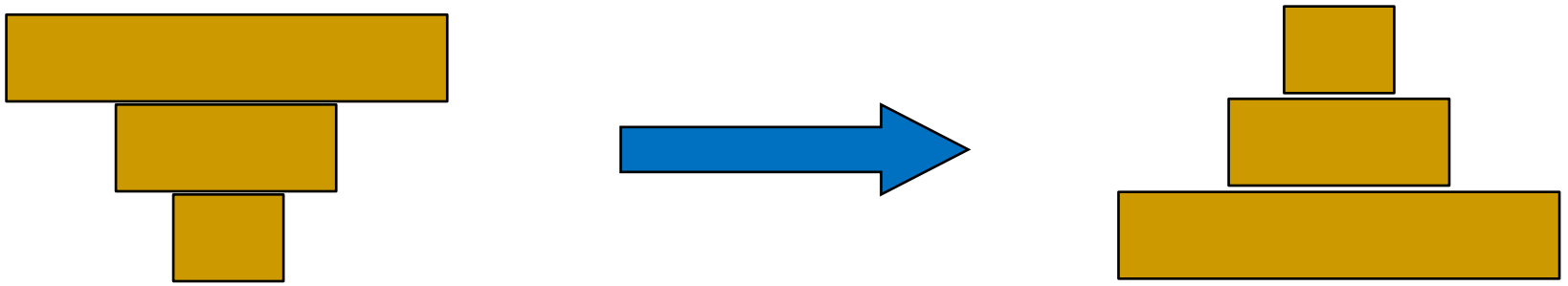


- The *state* of an agent includes state of its *internal environment* + state of *knowledge* and *beliefs* about its *external environment*.

Goal I (achieving exactly defined status)

Initial State

Goal

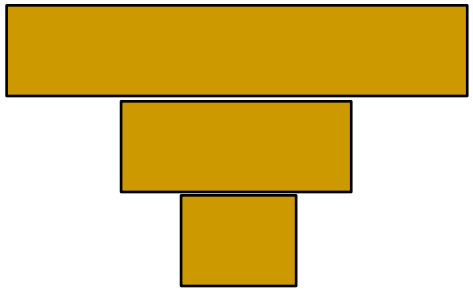


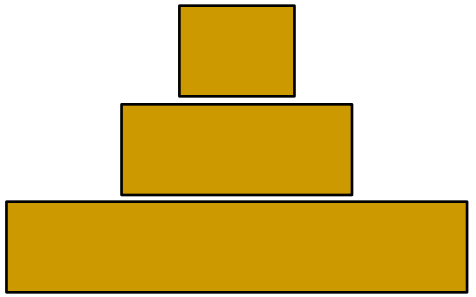
Goal II (achieving constrained status)

Goal

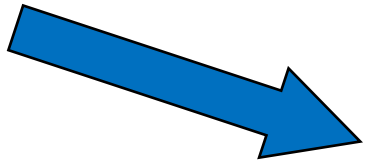
Constraint:
" *The smallest in on top*"

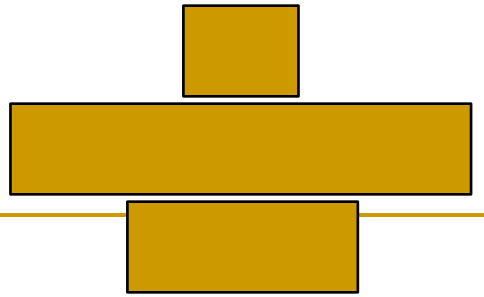
Initial State





OR

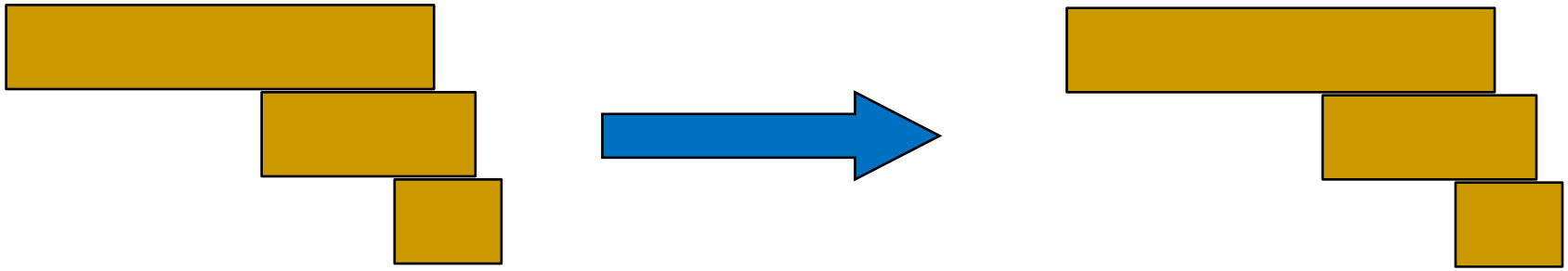




Goal III (continuously keeping instable status)

Initial State

Goal



Goal IV (maximizing utility)

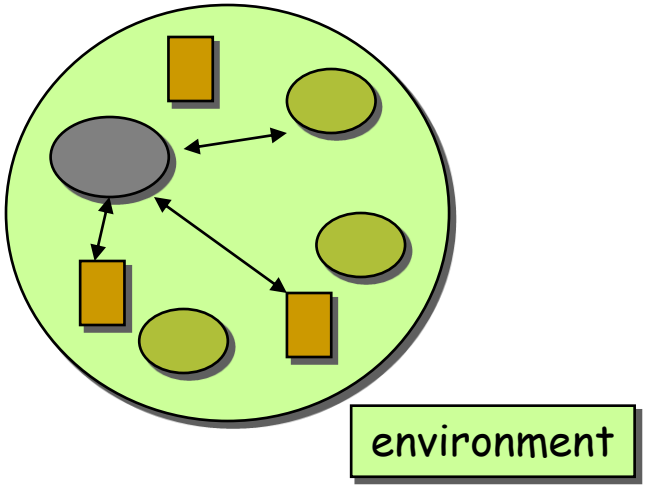
Initial State

Goal:

The basket filled with mushrooms that can be sold for maximum possible price

Situatedness

- An agent is *situated* in an *environment*, that consists of the objects and other agents it is possible to interact with.



- An agent has an *identity* that distinguishes it from the other agents of its environment.

Situated in an environment, which can be:

- Accessible/partially accessible/inaccessible

(with respect to the agent's precepts);

- Deterministic/nondeterministic

(current state can or not fully determine the next one);

- Static/dynamic

(with respect to time).

Agents & Environments

- In complex environments:
  - An agent do not have ***complete*** control over its environment, it just have ***partial*** control
  - Partial control means that an agent can ***influence*** the environment with its actions
  - An action performed by an agent may ***fail*** to have the desired effect.
- Conclusion: environments are ***non-deterministic***, and agents must be prepared for the possibility of ***failure***.

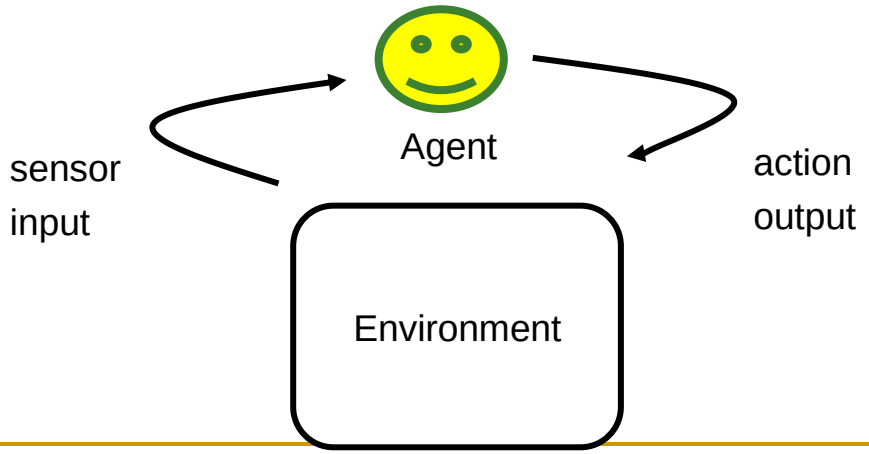
Agents & Environments

- Agent's *environment states* characterized by a set:

  S={ s1,s2,…}

- Effectoric capability of the Agent characterized by a set of *actions*:

  A={ a1,a2,…}



Standard agents

- A ***Standard agent*** decides what action to perform on the basis of his history (experiences).

- A Standard agent can be viewed as function

***action*: S* → A**

S* is the set of sequences of elements of S (states).

Environments

- *Environments* can be modeled as function

  $$env: S \times A \rightarrow P(S)$$

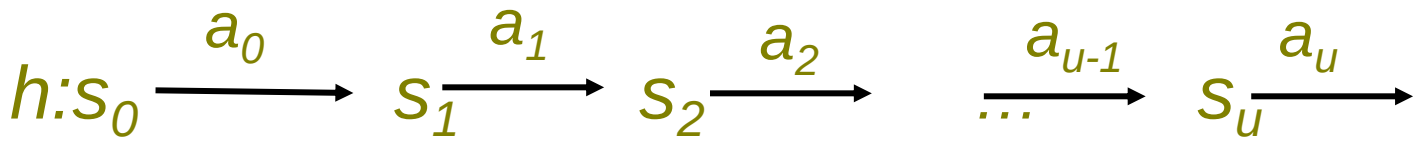
  where **P(S)** is the power set of *S (the set of all subsets of S)*;

  This function takes the current state of the environment $s \in S$ and an action $a \in A$ (performed by the agent), and maps them to a set of environment states *env(s,a)*.

- *Deterministic environment*: all the sets in the range of *env* are singletons (contain 1 instance).
- *Non-deterministic environment*: otherwise.

History

- History represents the interaction between an agent and its environment. A history is a sequence:



Where:

s_0 is the initial state of the environment

a_u is the u'th action that the agent choose to perform

s_u is the u'th environment state

Purely reactive agents

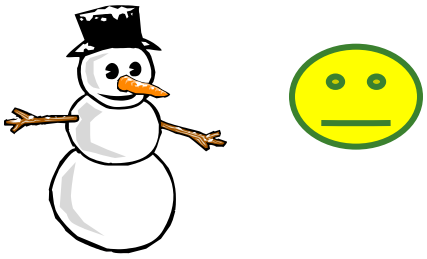
- A purely reactive agent decides what to do without reference to its history (no references to the past).
- It can be represented by a function

$$action: S \rightarrow A$$

- Example: thermostat

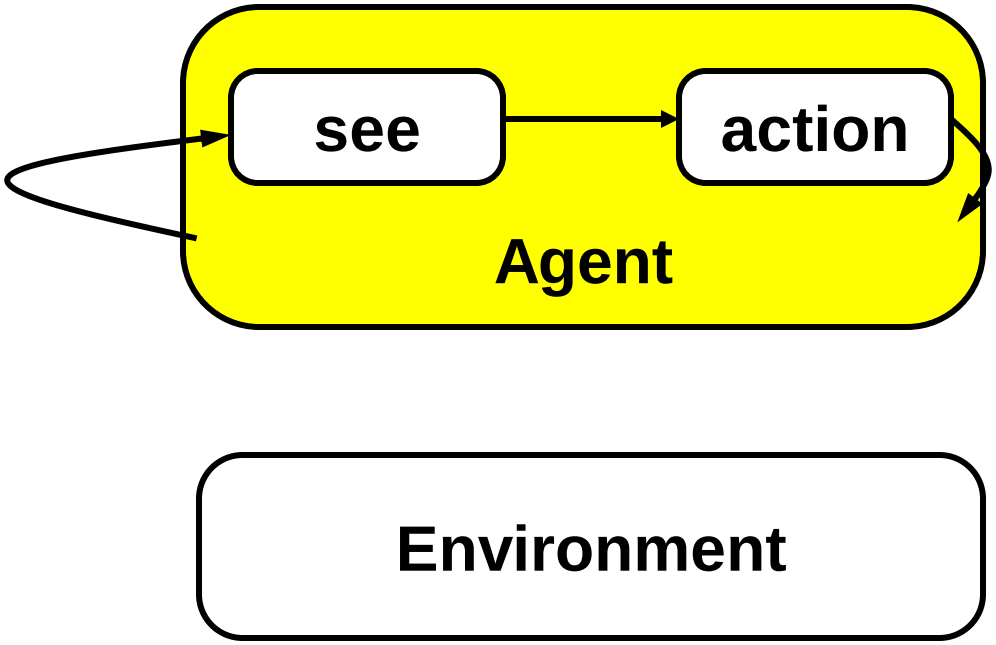
  Environment states: temperature OK; too cold

$$action(s) = \begin{cases} \text{heater off} & \text{if } s = \text{temperature OK} \\ \text{heater on} & \text{otherwise} \end{cases}$$



Perception

- *see* and *action* functions:



Perception

- *Perception* is the result of the function

  see: S → P

  where
  - *P* is a (non-empty) set of *percepts* (perceptual inputs).

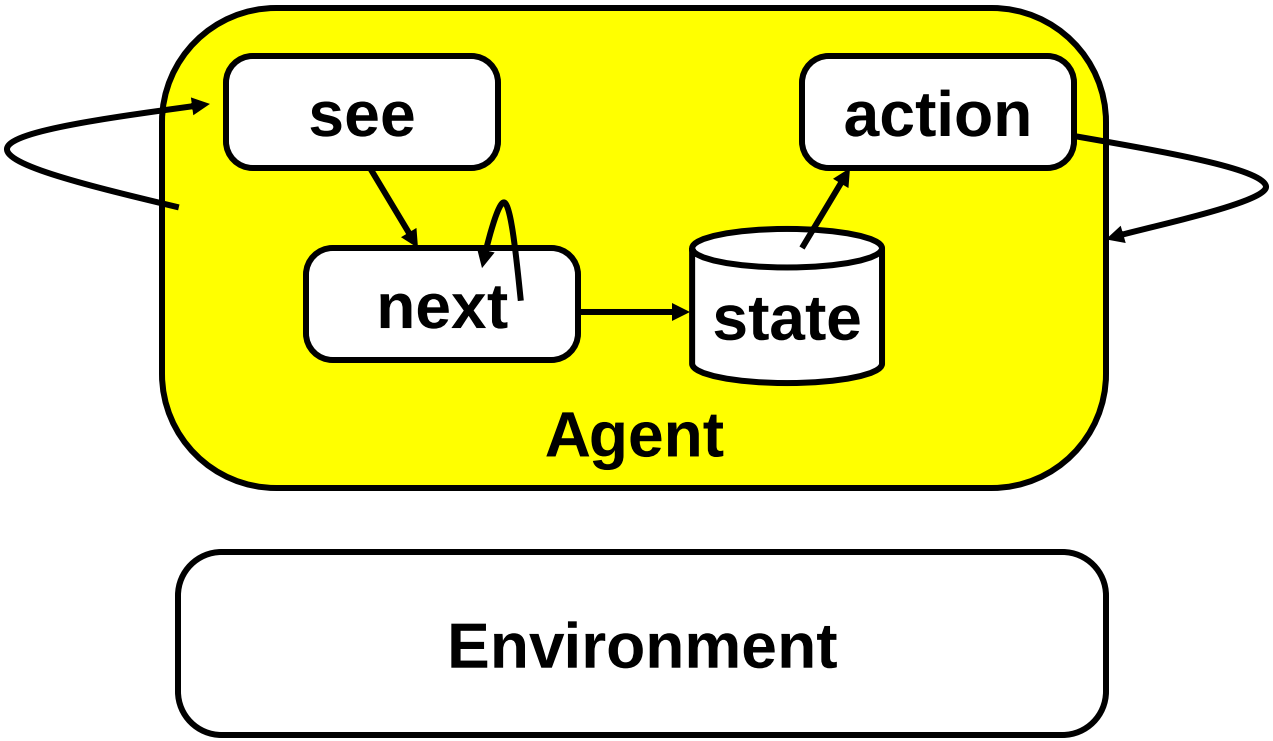
- Then, the *action* becomes:

  action: P → A*

  which maps sequences of percepts to actions

Agents with state

- *see*, *next* and *action* functions



Agents with state

- The same perception function:

  $see: S \rightarrow P$

- The action-selection function is now:

  $action: I \rightarrow A$

  where

  I: set of all internal states of the agent

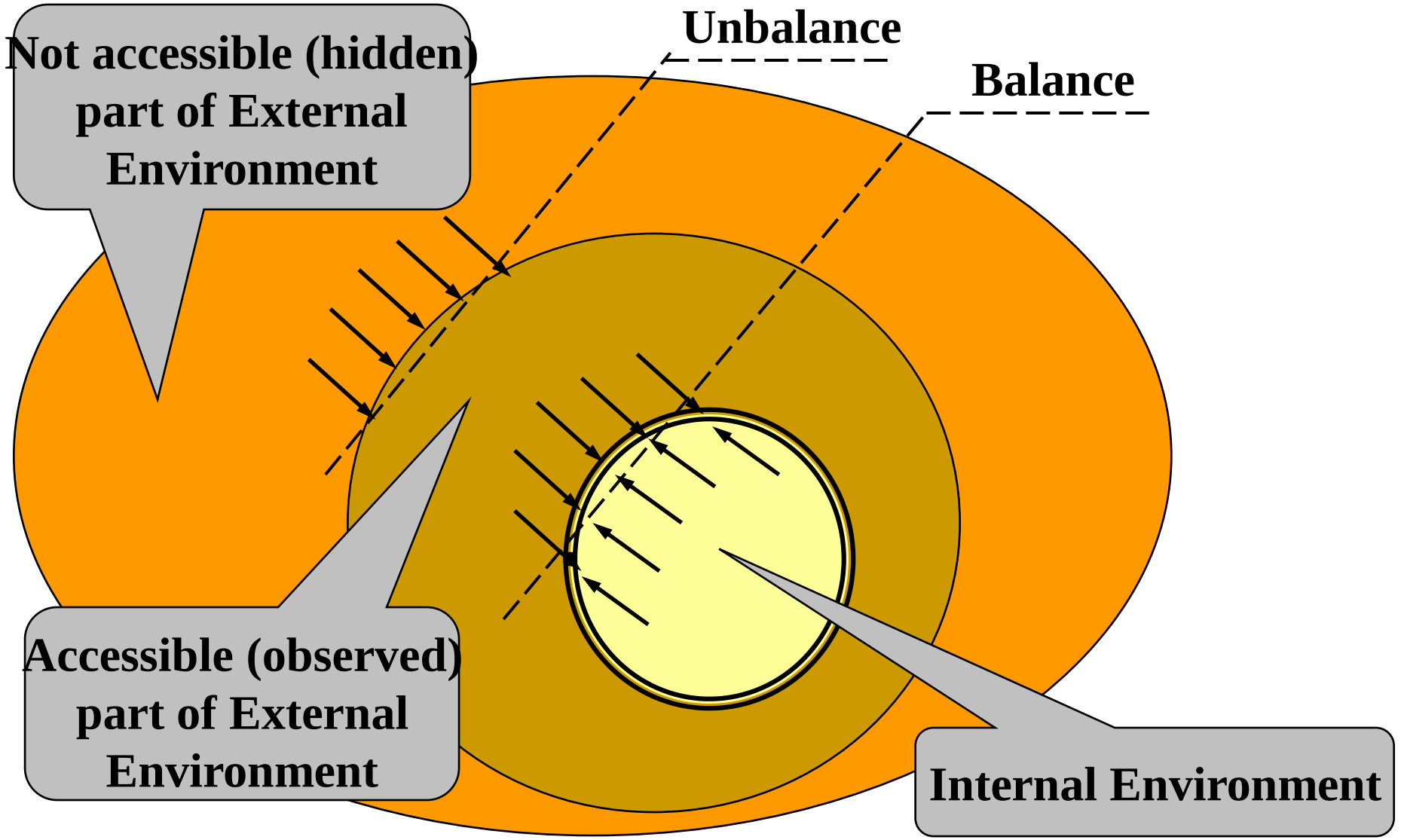
- An additional function is introduced:

  $next: I \times P \rightarrow I$

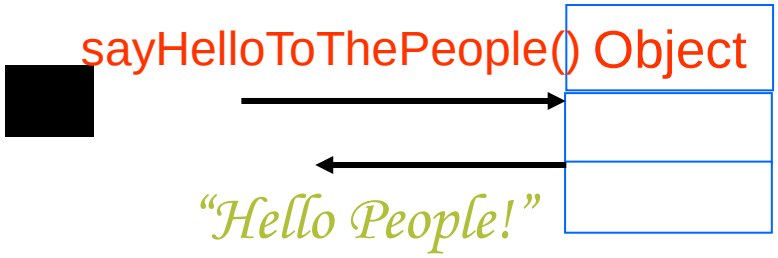
Agents with state

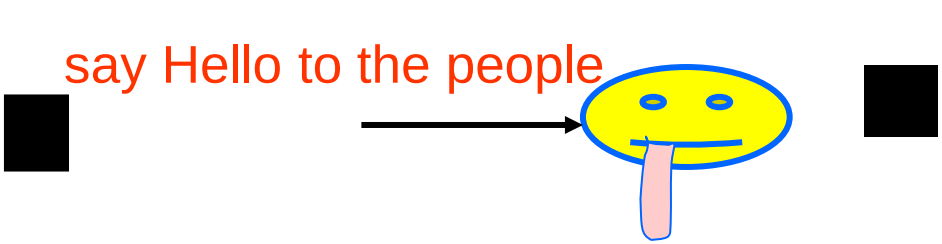
- Behavior:
  - The agent starts in some **internal initial state** i_0
  - Then **observes** its environment state ***s***
  - The **internal state** of the agent is **updated** with ***next*(i_0, *see*(*s*))**
  - The action selected by the agent becomes ***action*(*next*(i_0, *see*(*s*)))**, and it is performed
  - The agent repeats the cycle observing the environment

Unbalance in Agent Systems



Objects & Agents





- Classes control its states
- Agents control its states and **behaviors**

"Objects do it for free; agents do it for money"

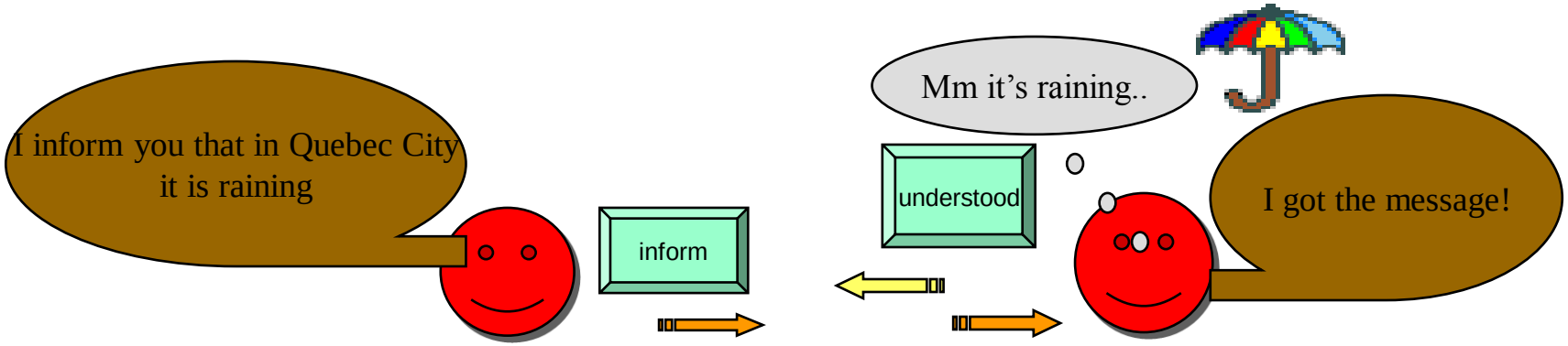
Agent's Activity

Agents actions can be:

- *direct*, i.e., they affect properties of objects in the environment;

- *communicative / indirect*, i.e., send messages with the aim of affecting mental attitudes of other agents;

Messages have a wel-defined *semantics*, they embed a content expressed in a given *content language* and containing terms whose meaning is defined in a given *ontology.*



- *planning*, i.e. making decisions about future actions.

Today

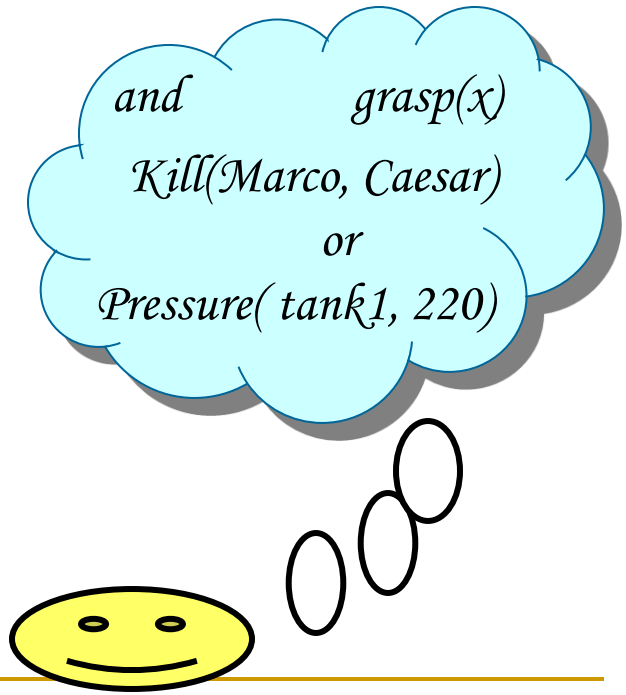
- Web of knowledge
- Why Agent?
- What is an agent
- Features of Intelligent Agent
- Types of Agents
  - Logic-based agents
  - Reactive agents
  - Belief-desire-intention agents
  - Layered architectures
- Multiagent Systems

Types of Agents

- Logic-based agents
- Reactive agents
- Belief-desire-intention agents
- Layered architectures

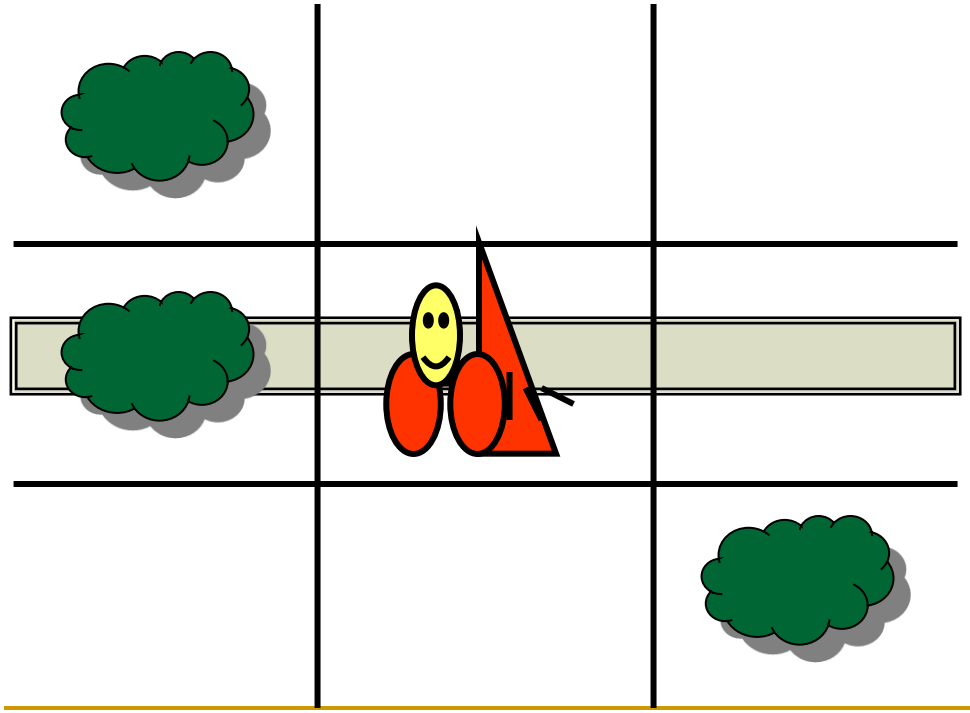
Logic-based architectures

- "Traditional" approach to build artificial intelligent systems:
  - ***Logical formulas***: symbolic representation of its environment and desired behavior.
  - ***Logical deduction*** or ***theorem proving***: syntactical manipulation of this representation.



Logic-based architectures: example

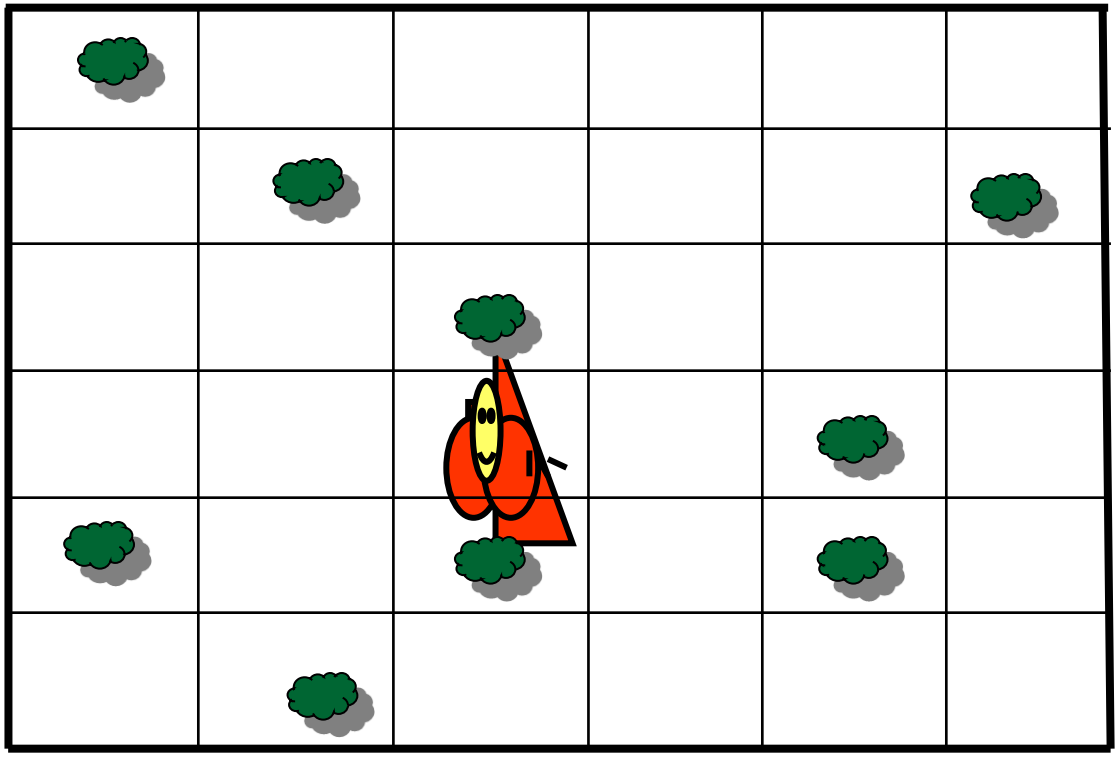
- A cleaning robot



- *In(x,y)* agent is at *(x,y)*
- *Dirt(x,y)* there is a dirt at *(x,y)*
- *Facing(d)* the agent is facing direction *d*
- $\forall x,y\ (\neg\ Dirt(x,y))$ – **goal**
- **Actions:**
  - *change_direction*
  - *move_one_step*
  - *suck*

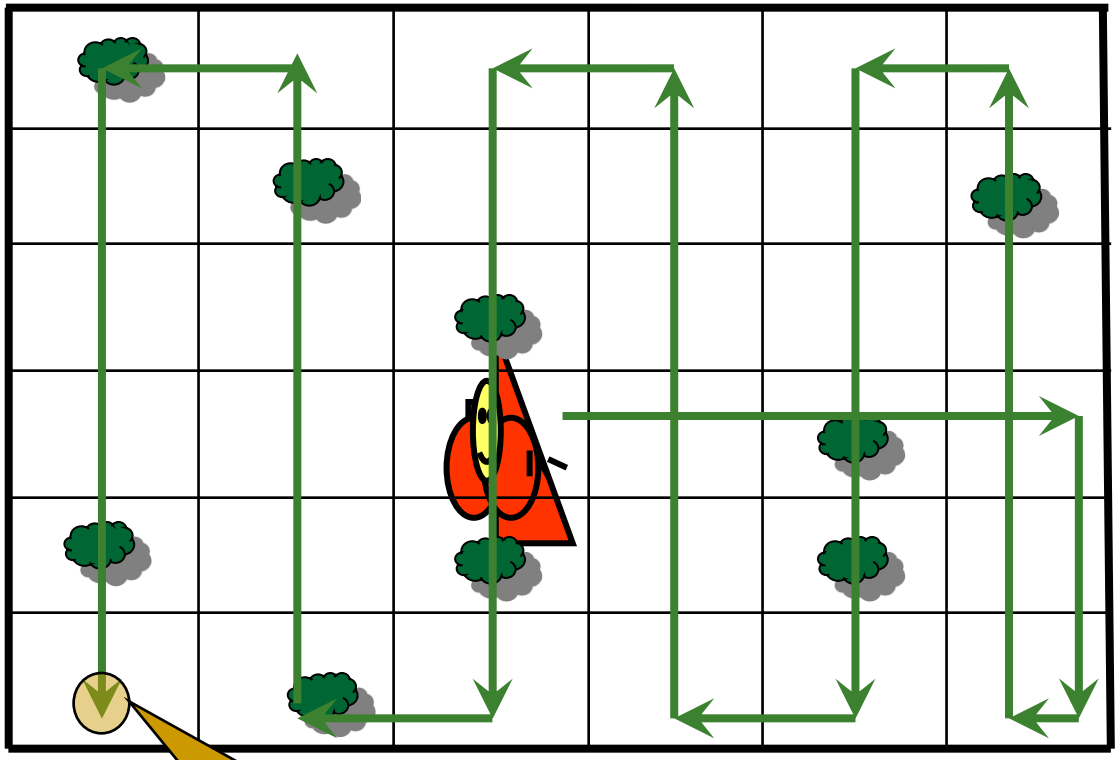
Logic-based architectures: example

- What to do ?



Logic-based architectures: example

Solution

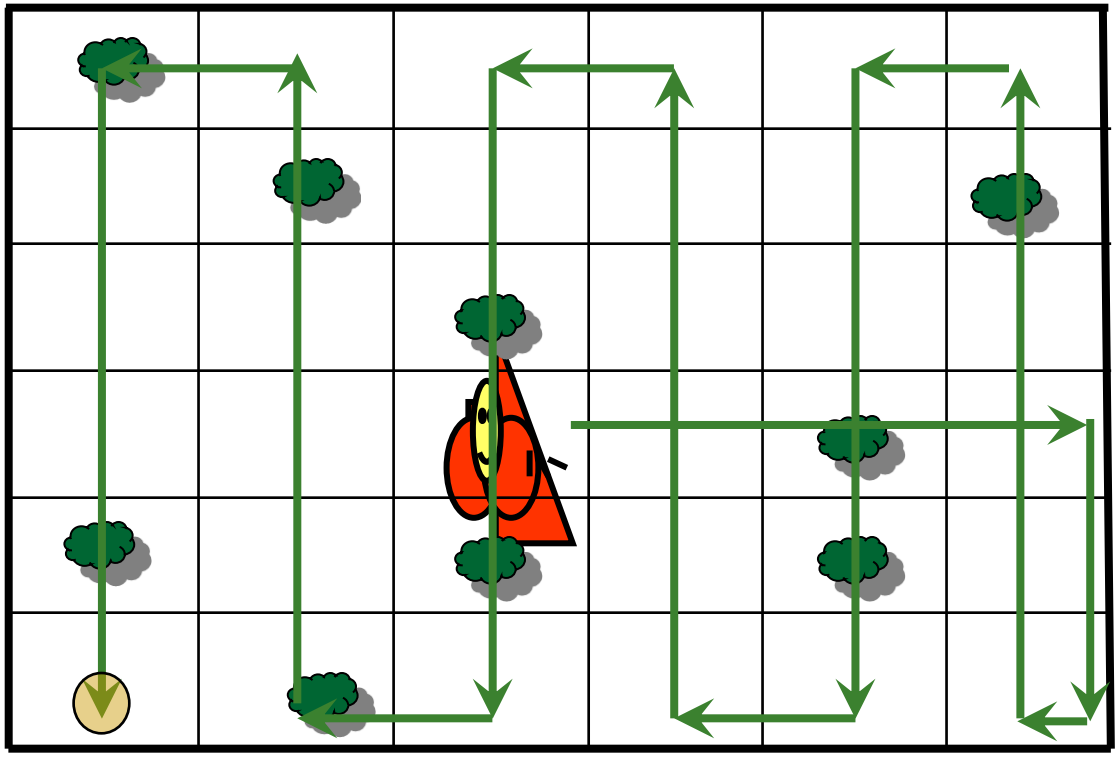


What is stopping criterion ?!

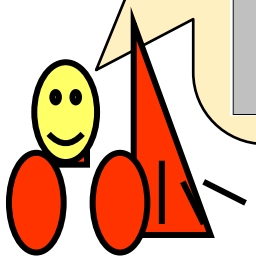
```
start
        // finding corner
continue while fail  { do move_one_step}
do change_direction
continue while fail  {do move_one_step}
do change_direction
         finding corner //
         // cleaning
continue  {
      remember In(x,y) to Mem
      do change_direction
      continue while fail      {
           if Dirt(In(x,y)) then suck
           do move_one_step   }
      do change_direction
      do change_direction
      do change_direction
      continue while fail      {
           if Dirt(In(x,y)) then suck
          do move_one_step   }
      if In(x,y) equal Mem then        stop
                }
       cleaning //
```

Logic-based architectures: example

is that intelligent?



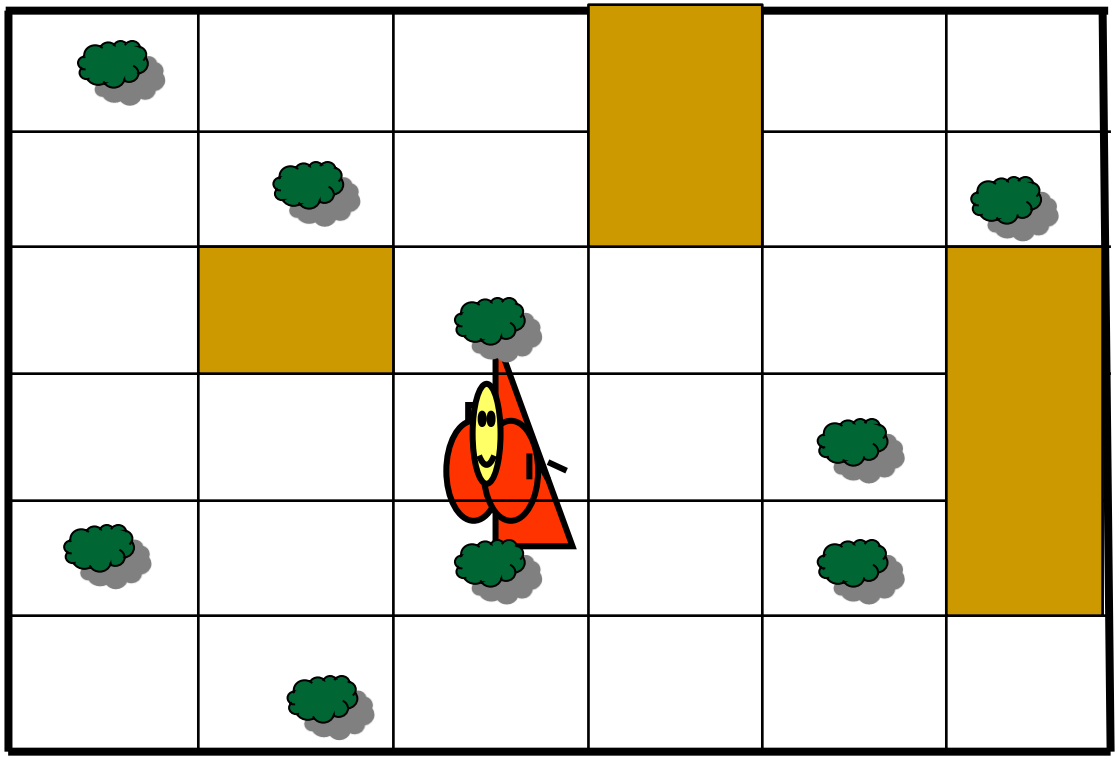
```
start
      // finding corner
continue while fail { do move_one_step}
do change_direction
continue while fail {do move_one_step}
do change_direction
       finding corner //
      // cleaning
continue {
    remember In(x,y) to Mem
    do change_direction
    continue while fail     {
       if Dirt(In(x,y)) then suck
       do move_one_step  }
    do change_direction
    do change_direction
    do change_direction
    continue while fail     {
       if Dirt(In(x,y)) then suck
       do move_one_step  }
    if In(x,y) equal Mem then       stop
          }
     cleaning //
```



How to make our agent capable to "invent" (derive) such a solution (plan) autonomously by itself ?!

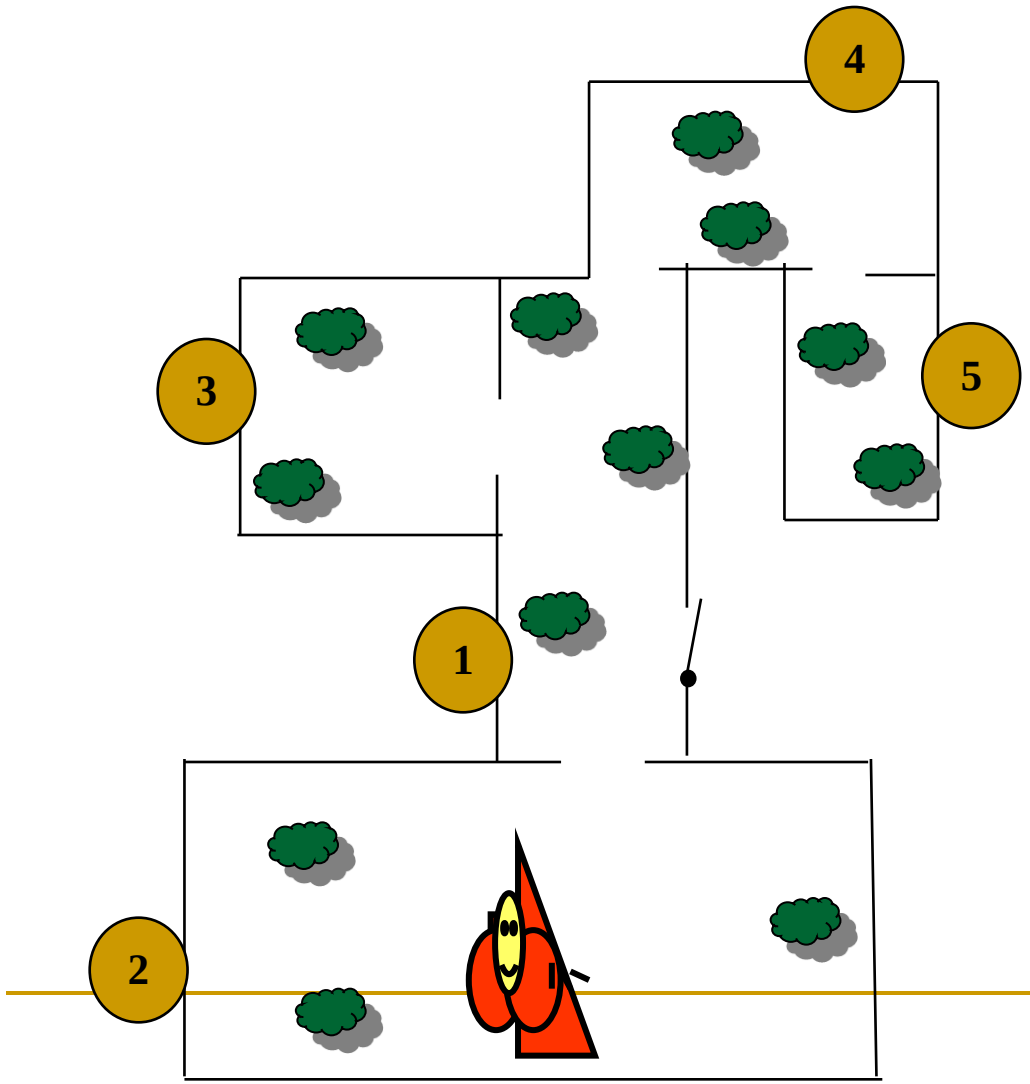
Logic-based architectures: example

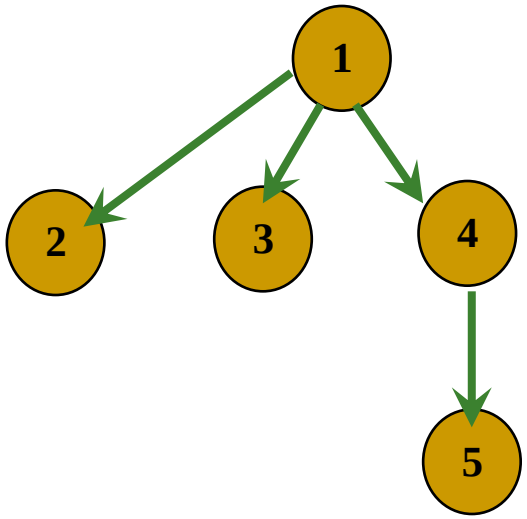
Looks like previous solution will not work here. What to do ?



Logic-based architectures: example

- What to do now?

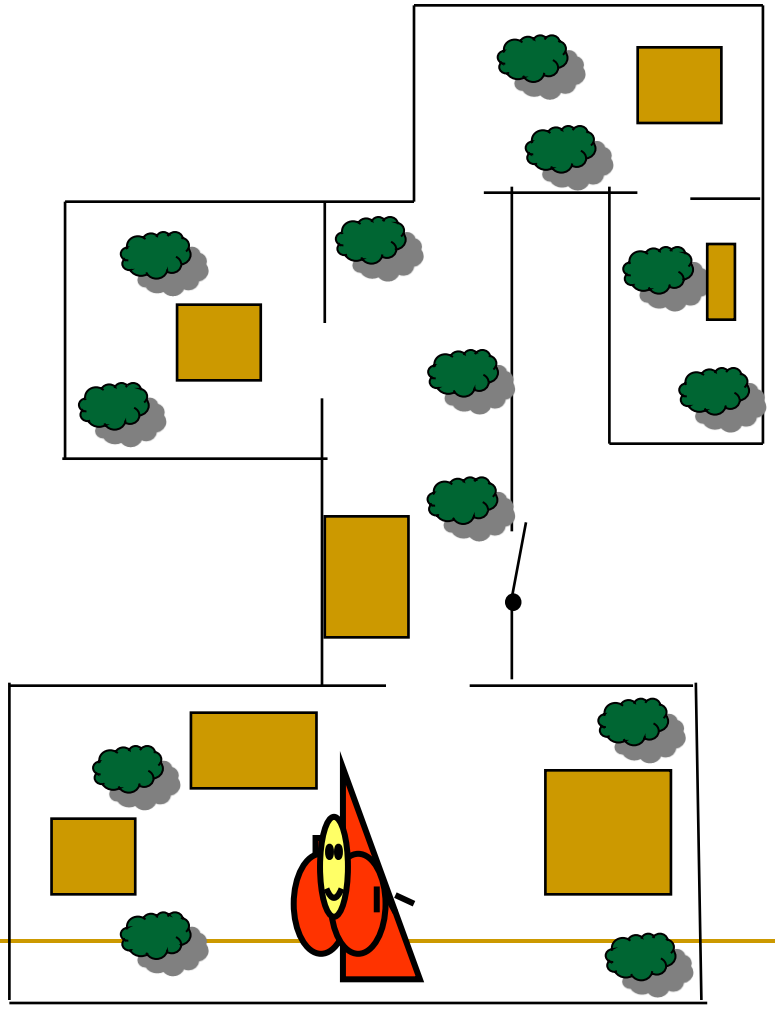




Restriction: an Apartment has a tree-like structure of rectangle rooms !

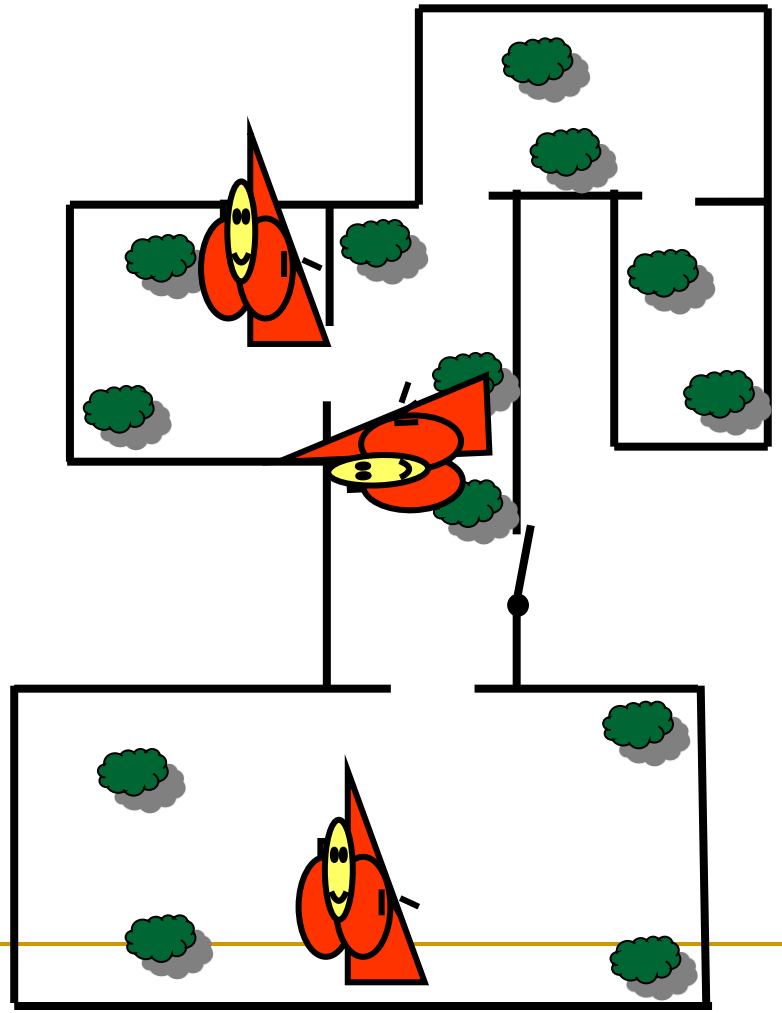
Logic-based architectures: example

What now???



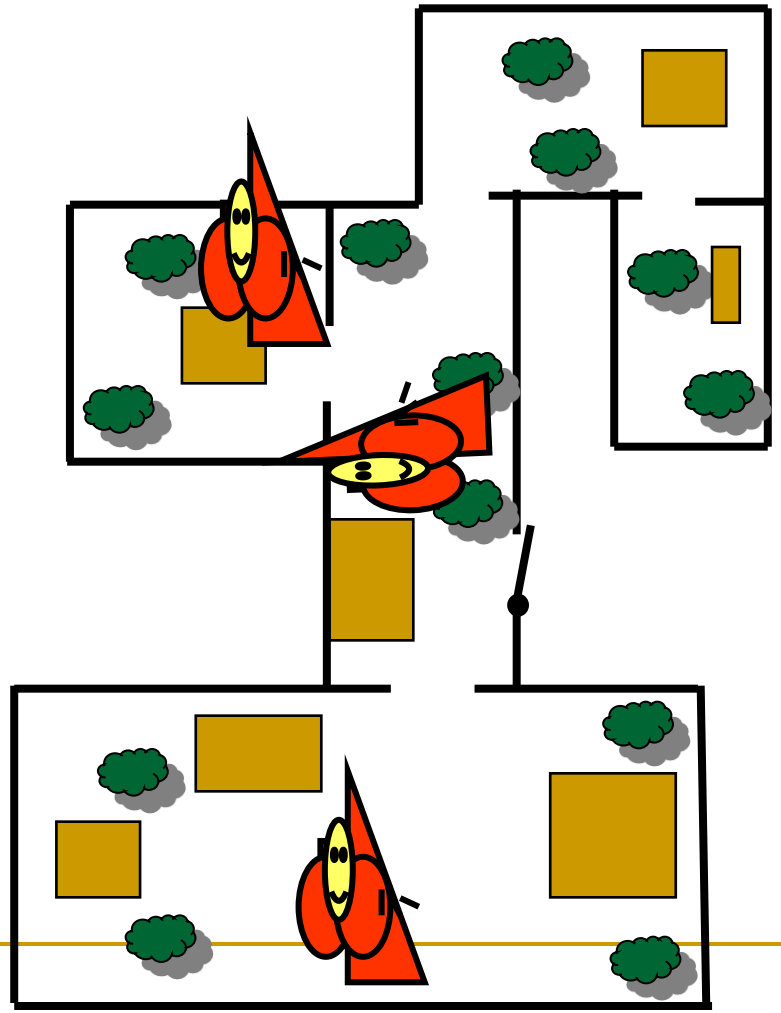
Logic-based architectures: example

- or now … ??!



Logic-based architectures: example

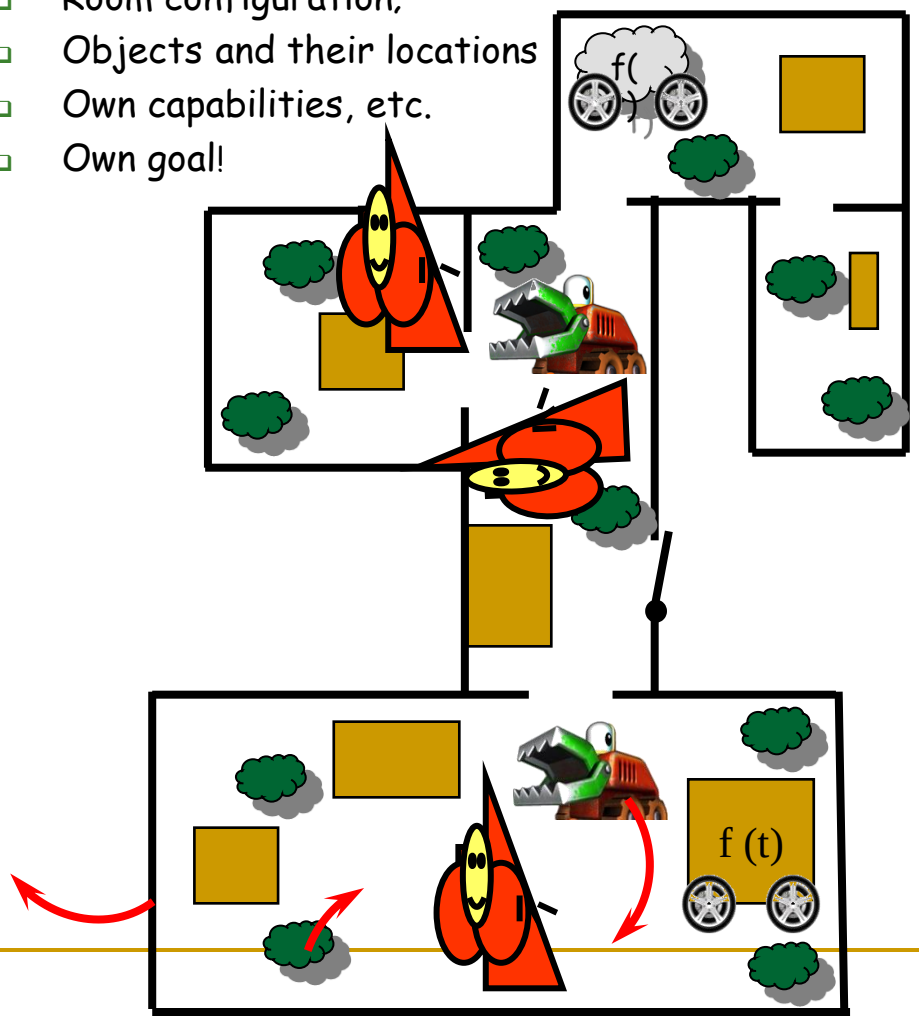
- or now … ??!



Logic-based architectures: example

Now … ???!!!!!!

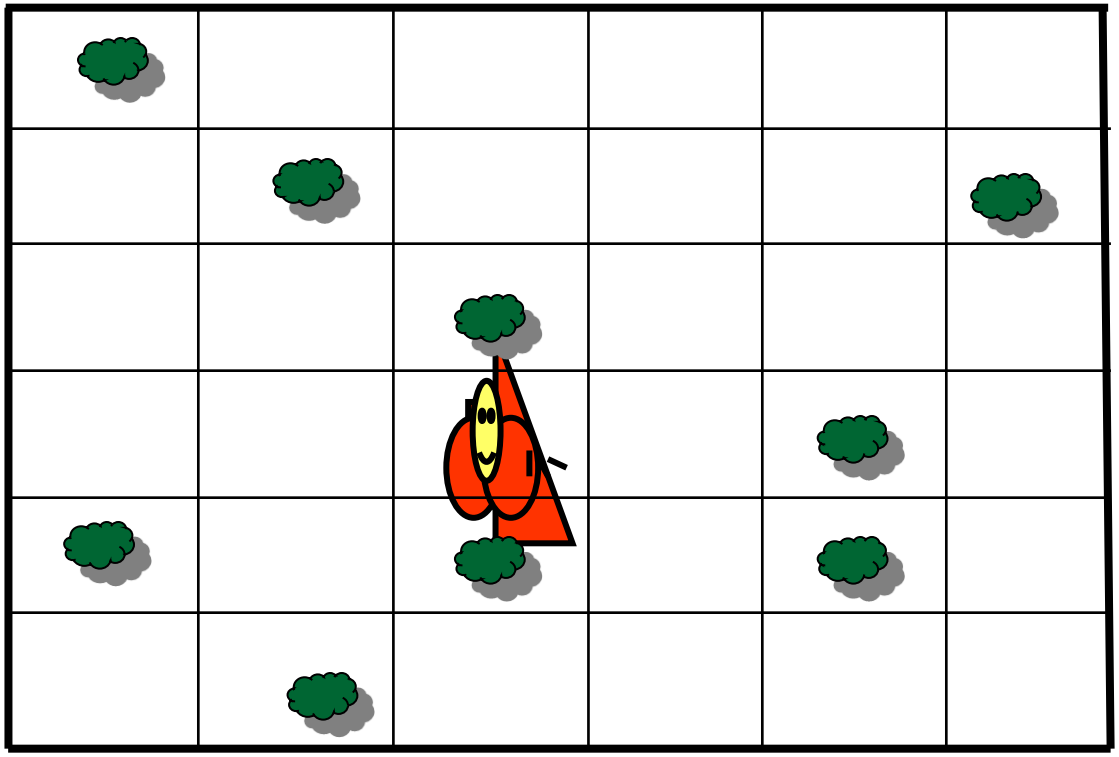
- Everything may change:
  - Room configuration;
  - Objects and their locations
  - Own capabilities, etc.
  - Own goal!



- When you will be capable to design such a system, this means that you have learned more than everything you need from the course "Intelligent Systems"

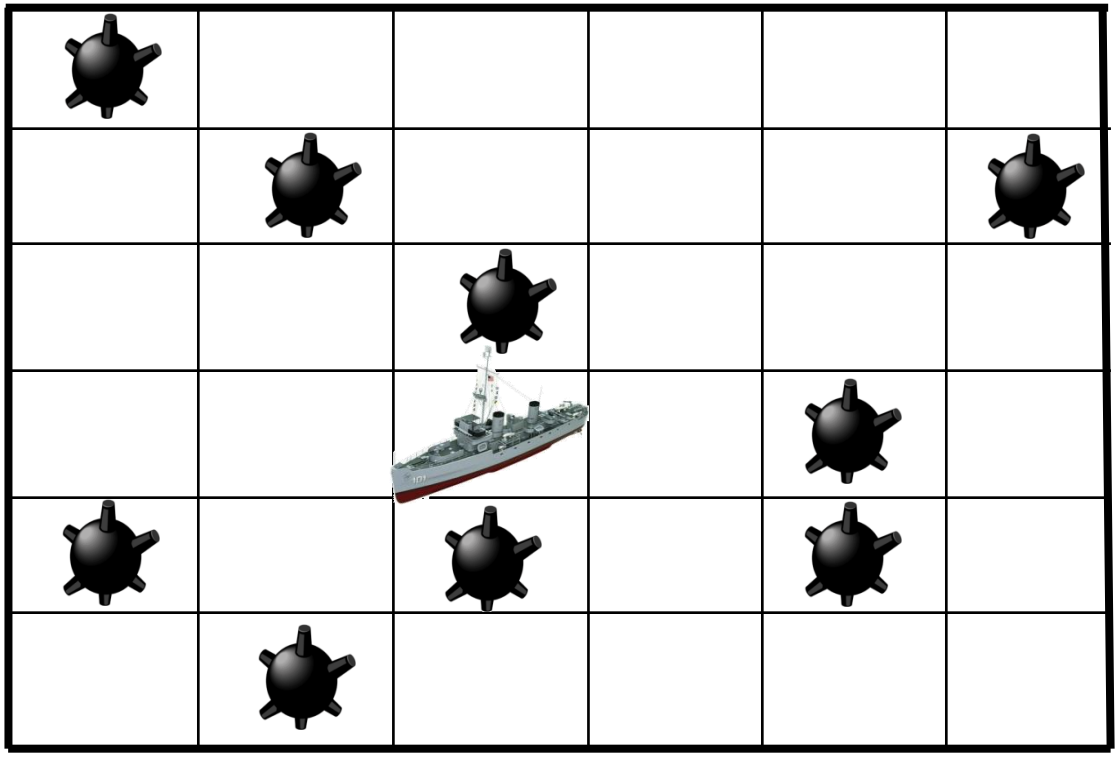
Logic-based architectures: example

- You may guess that the problem …



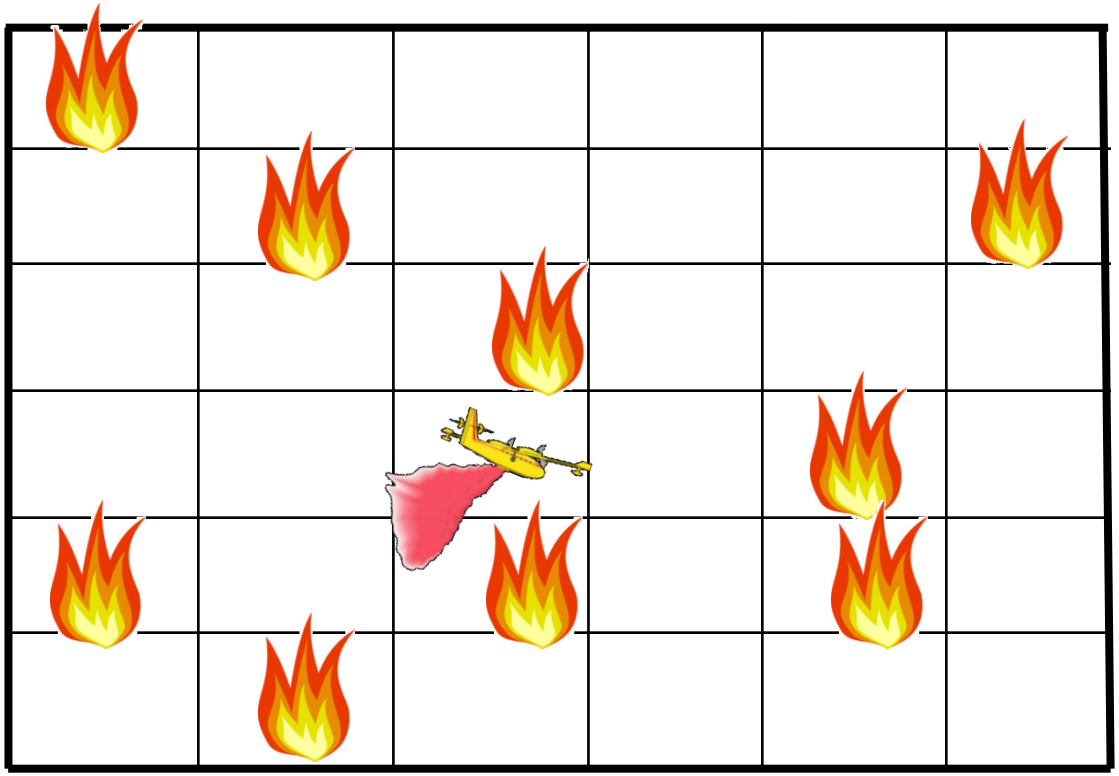
Logic-based architectures: example

- … is very similar in many other domains also!



Logic-based architectures: example

- You may guess that the problem is very similar in many other domains also!



The Open World Assumption (1)

The Open World Assumption (OWA): a lack of information does not imply the missing information to be false.

Relational Approach	(Open) Semantic Web Approach
Closed World Assumption (CWA)	**Open World Assumption (OWA)**
That which is not known to be true is presumed to be false; it needs to be explicitly stated as true. *Negation as failure* (NAF) is a related assumption, since it assumes as false every predicate that cannot be proven to be true. Under CWA, any statement not known to be true is false.	The lack of a given assertion or fact being available does not imply whether that possible assertion is true or false: it simply is not known. In other words, lack of knowledge does not imply falsity.
Everything is prohibited until it is permitted.	Everything is permitted until it is prohibited.
Unique Name Assumption (UNA)	**Duplicate Labels Allowed**
The unique name assumption (UNA) is premised that different names always refer to different entities in the world.	OWL allows different synonym labels to be used for the same object; same names may refer to different objects. Identity assertions must be explicitly stated.

http://www.mkbergman.com/852/

The Open World Assumption (2)

Relational Approach	(Open) Semantic Web Approach
Closed World Assumption (CWA)	**Open World Assumption (OWA)**
Complete Information	**Incomplete Information**
The data system at hand is assumed to be complete. (Missing information is often handled via the null statement in SQL, but that has been controversial and contentious in its own right.) This is also known as the *domain-closure assumption*.	A central tenet of OWA is that information is incomplete. A corollary is that the attributes of specific objects or instances may also be incomplete or partially known.
Single Schema (one world)	**Many World Interpretations**
A single schema is necessary to define the scope and interpretation of the world (domain at hand).	Schema and data instance assertions are kept separate. Multiple interpretations (worlds) for the same data are possible.
Integrity Constraints	**Logical Axioms (restrictions)**
Integrity constraints prevent "incorrect" values from being asserted in the relational model. It is useful for validation/parsing/data input and is related to the single model that contains only the facts asserted. Strict cardinality is used for checking validation.	Logical axioms provide restrictions through property domains and ranges. Everything can be true unless proven otherwise, and multiple possible models can satisfy the axioms. This provides more powerful inferencing, though can also be unintuitive at times. Cardinality and range restrictions exhibit different behavior for objects (inferred) or datatypes.

http://www.mkbergman.com/852/

The Open World Assumption (3)

Relational Approach	(Open) Semantic Web Approach
Closed World Assumption (CWA)	**Open World Assumption (OWA)**
Non-monotonic Logic	**Monotonic Logic**
The set of conclusions warranted on the basis of a given knowledge base does not increase (in fact, it likely shrinks) with the size of the knowledge base [*].	The hypotheses of any derived fact may be freely extended with additional assumptions. Additional assertions tend to reduce the inferences or entailments that can be applied. A new piece of knowledge cannot reduce what is known [*]. New knowledge can arise through inference.

* The logic or inference system of classical model theory is ***monotonic***. That is, it has the behavior that if S entails E then (S + T) entails E. In other words, adding information to some prior conditions or assertions cannot invalidate a valid entailment. The basic intuition of model-theoretic semantics is that asserting a statement makes a claim about the world: it is another way of saying that the world is, in fact, so arranged as to be an interpretation which makes the statement true.

In comparison, a ***non-monotonic*** logic system may include *default reasoning*, where one assumes a 'normal' general truth unless it is contradicted by more particular information (birds normally fly, but penguins don't fly); *negation-by-failure*, commonly assumed in logic programming systems, where one concludes, from a failure to prove a proposition, that the proposition is false; and *implicit closed-world assumptions*, often assumed in database applications, where one concludes from a lack of information about an entity in some corpus that the information is false (*e.g.*, that if someone is not listed in an employee database, that he or she is not an employee.)

http://www.mkbergman.com/852/

The Open World Assumption (4)

Relational Approach	(Open) Semantic Web Approach
Closed World Assumption (CWA)	**Open World Assumption (OWA)**
Fixed and Brittle	**Reusable and Extensible**
Changing the schema requires re-architecting the database; not inherently extensible.	Designed from the ground up to reuse existing ontologies (axioms) and to be extensible. Database design and management can be more agile, with schema evolving incrementally.
Flat Structure; Strong Typing	**Graph Structure; Open Typing**
Information organized into flat tables; linkages and connections between tables based on foreign keys or joins. Strong data typing orientation.	Inherent graph structure, supporting of linkage and connectivity analysis. Datatypes are inherently loose, though axioms can add strong types. Datatypes treated in the same way as classes, and datatype values are treated in the same way as individual identiers (*i.e.*, a data value is treated as referring to an object).
Querying and Tooling	**Querying and Tooling**
SQL and query optimizers well developed. Tooling well developed. Disjunction not supported; negation must be accommodated through approaches such as NAF. Sums and counts are easier due to unique name premise. Answer closure (one answer passable to a next calculation) is easier than OWA. Most tools are not suitable for any arbitrary schema.	SPARQL and emerging rule languages used for querying; performance at scale and with broad distribution a concern. Queries require contextual information for proper set selection. Negation and disjunction are allowed and are powerful constructs. Tools generally less developed. Exciting opportunities for *ontology-driven applications* working against a small set of generic tools.

http://www.mkbergman.com/852/

Characteristics of OWA-based knowledge systems(1)

Knowledge is never complete — gaining and using knowledge is a process, and is never complete. A completeness assumption around knowledge is by definition inappropriate;

Knowledge is found in structured, semi-structured and unstructured forms — structured databases represent only a portion of structured information in the enterprise (spreadsheets and other non-relational data-stores provide the remainder). Further, general estimates are that 80% of information available to enterprises reside in documents, with a growing importance to metadata, Web pages, markup documents and other semi-structured sources. A proper data model for knowledge representation should be equally applicable to these various information forms; the open semantic language of RDF is specifically designed for this purpose;

Knowledge can be found anywhere — the open world assumption does not imply open information only. However, it is also just as true that relevant information about customers, products, competitors, the environment or virtually any knowledge-based topic can also not be gained via internal information alone. The emergence of the Internet and the universal availability and access to mountains of public and shared information demands its thoughtful incorporation into KM systems. This requirement, in turn, demands OWA data models;

Knowledge structure evolves with the incorporation of more information — our ability to describe and understand the world or our problems at hand requires inspection, description and definition. Birdwatchers, botanists and experts in all domains know well how inspection and study of specific domains leads to more discerning understanding and "seeing" of that domain. Before learning, everything is just a shade of green or a herb, shrub or tree to the incipient botanist; eventually, she learns how to discern entire families and individual plant species, all accompanied by a rich domain language. This truth of how increased knowledge leads to more structure and more vocabulary needs to be explicitly reflected in our KM systems;

http://www.mkbergman.com/852/

Characteristics of OWA-based knowledge systems(2)

Knowledge is contextual — the importance or meaning of given information changes by perspective and context. Further, exactly the same information may be used differently or given different importance depending on circumstance. Still further, what is important to describe (the "attributes") about certain information also varies by context and perspective. Large knowledge management initiatives that attempt to use the relational model and single perspectives or schema to capture this information are doomed in one of two ways: either they fail to capture the relevant perspectives of some users; or they take forever and massive dollars and effort to embrace all relevant stakeholders' contexts;

Knowledge should be coherent — (i.e., internally logically consistent). Because of the power of OWA logics in inferencing and entailments, whatever "world" is chosen for a given knowledge representation should be coherent. Various fantasies, even though not real, can be made believable and compelling by virtue of their coherence;

Knowledge is about connections — knowledge makes the connections between disparate pieces of relevant information. As these relationships accrete, the knowledge base grows. Again, RDF and the open world approach are essentially connective in nature. New connections and relationships tend to break brittle relational models, and …;

Knowledge is about its users defining its structure and use — since knowledge is a state of understanding by practitioners and experts in a given domain, it is also important that those very same users be active in its gathering, organization (structure) and use. Data models that allow more direct involvement and authoring and modification by users — as is inherently the case with RDF and OWA approaches — bring the knowledge process closer to hand. Besides this ability to manipulate the model directly, there are also the immediacy advantages of incremental changes, tests and tweaks of the OWA model. The schema consensus and delays from single-world views inherent to CWA remove this immediacy, and often result in delays of months or years before knowledge structures can actually be used and tested.

Characteristics of OWA-based knowledge systems(3)

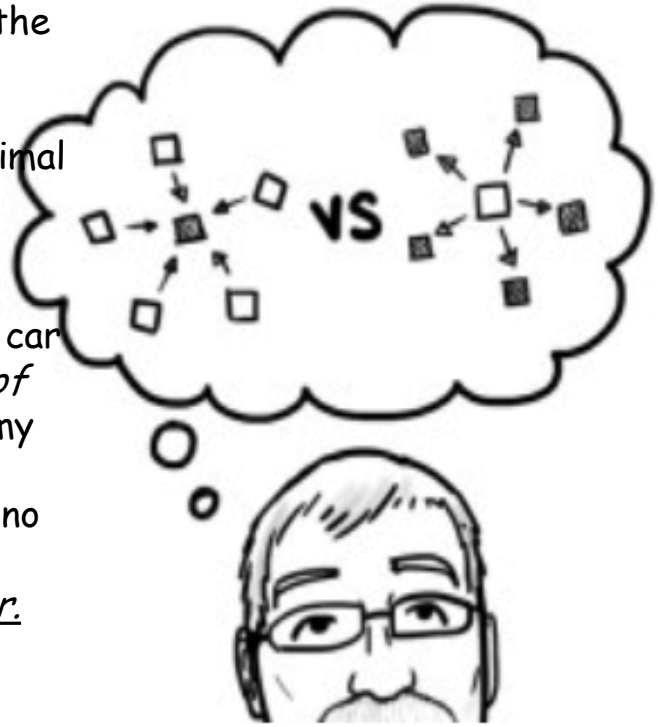
- Domains can be analyzed and inspected incrementally;
- Schema can be incomplete and developed and refined incrementally;
- The data and the structures within these open world frameworks can be used and expressed in a piecemeal or incomplete manner;
- We can readily combine data with partial characterizations with other data having complete characterizations;
- Systems built with open world frameworks are flexible and robust; as new information or structure is gained, it can be incorporated without negating the information already resident, and …;
- Open world systems can readily bridge or embrace closed world subsystems.

http://www.mkbergman.com/852/

CWA vs. OWA: Convergent vs. Divergent Reasoning

Convergent Reasoning is the practice of trying to solve a discrete challenge quickly and efficiently by selecting the optimal solution from a finite set.

Convergent example:
I live four miles from work. My car gets 30 MPG (*Miles Per Gallon of gas*). I want to use less fuel in my commute for financial and conservation reasons. Money is no object. *Find the three best replacement vehicles for my car.*

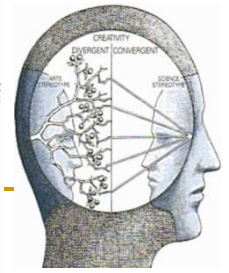


Divergent Reasoning takes a challenge and attempts to identify all of the possible drivers of that challenge, then lists all of the ways those drivers can be addressed (it's more than just brainstorming).

Divergent Example:
I live four miles from work. My car gets 30 MPG. I want to use less fuel in my commute for financial and conservation reasons. Money is no object. *What options do I have to reduce my fuel consumption?*

Both examples will produce valuable results. The convergent example may be driven by another issue – perhaps my current car was totaled and I only have a weekend to solve the problem. The divergent example may take more time to investigate – but you may discover an option that is completely different than what the user has asked you to do – like start your own company from home or invent a car th

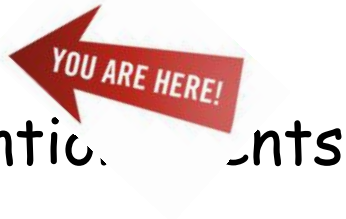
By Phil Charron
VICE PRESIDENT, EXPERIENCE DESIGN



http://creativegibberish.org/439/divergent-thinking/

Types of Agents

- Logic-based agents
- Reactive agents
- Belief-desire-intentio[illegible]nts
- Layered architectures

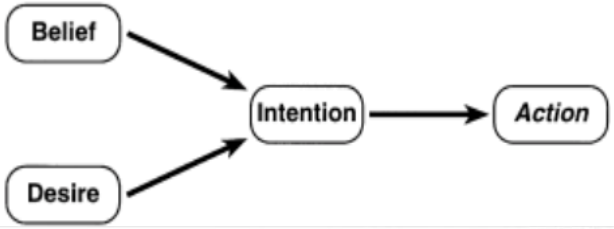


Types of Agents

- Logic-based agents
- Reactive agents
- Belief-desire-intention agents
- Layered architectures

Belief-Desire-Intention (BDI) architectures

- They have their roots in understanding ***practical reasoning***.
- It involves two processes:
  - ***Deliberation***: deciding ***which*** goals we want to achieve.
  - ***Means-ends reasoning*** ("planning"): deciding ***how*** we are going to achieve these goals.



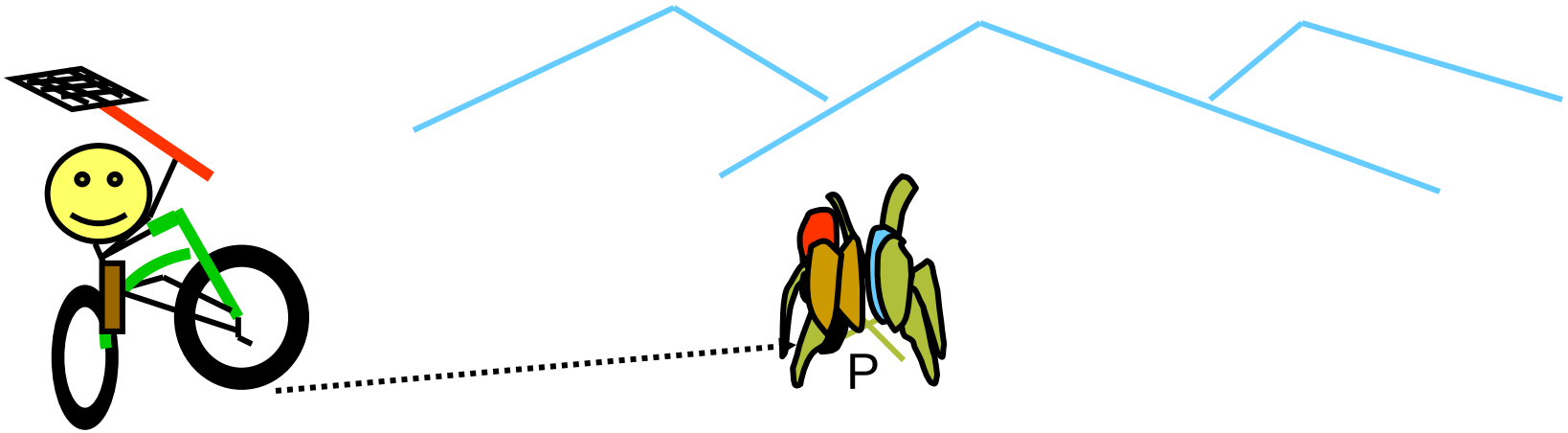
BDI architectures

- First: try to understand what ***options*** are available.
- Then: ***choose*** between them, and ***commit*** to some.
- Intentions influence beliefs upon which future reasoning is based

These chosen options become intentions, which then determine the ***agent's actions***.

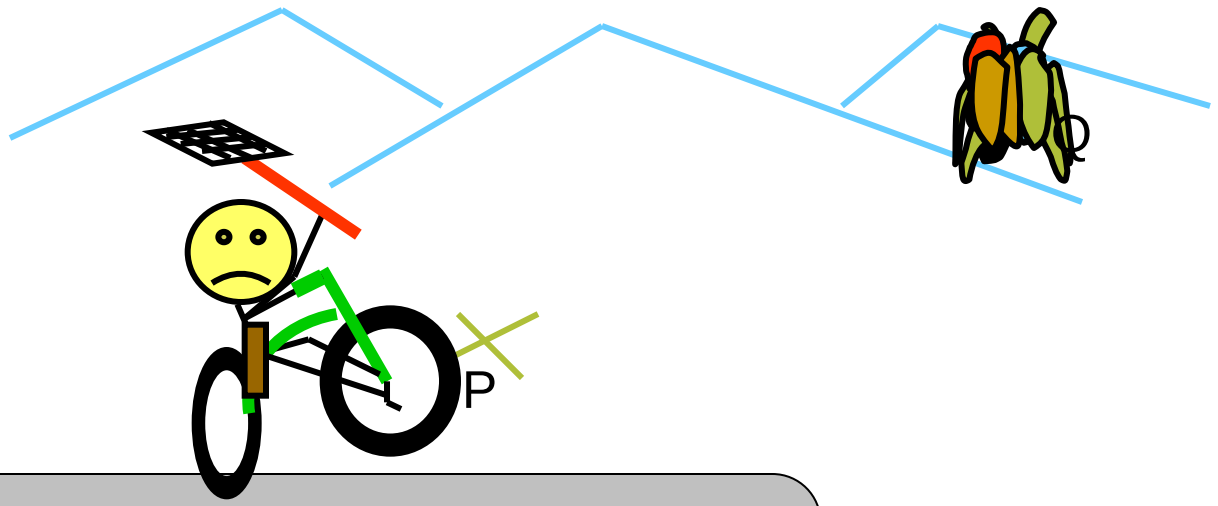
BDI architectures: reconsideration of intentions

- Example




```
Time t = 0
Desire: Kill the alien
Intention: Reach point P
Belief: The alien is at P
```

BDI architectures: reconsideration of intentions



Time t = 1
Desire: Kill the alien
Intention: Kill the alien
Belief: The alien is at P *Wrong!*

BDI Architecture

- **Beliefs**: Beliefs represent the informational state of the agent, in other words its beliefs about the world (including itself and other agents). Beliefs can also include inference rules, allowing forward chaining to lead to new beliefs. Using the term *belief* rather than *knowledge* recognizes that what an agent believes may not necessarily be true (and in fact may change in the future).
- **Desires**: Desires represent the motivational state of the agent. They represent objectives or situations that the agent *would like* to accomplish or bring about. Examples of desires might be: *find the best price*, *go to the party* or *become rich*.
  - **Goals**: A goal is a desire that has been adopted for active pursuit by the agent. Usage of the term *goals* adds the further restriction that the set of active desires must be consistent. For example, one should not have concurrent goals to go to a party and to stay at home even though they could both be desirable.

Source : Wikipedia

BDI Architecture

- **Intentions**: Intentions represent the deliberative state of the agent – what the agent *has chosen* to do. Intentions are desires to which the agent has to some extent committed. In implemented systems, this means the agent has begun executing a plan.
  - **Plans**: Plans are sequences of actions (recipes or knowledge areas) that an agent can perform to achieve one or more of its intentions. Plans may include other plans: my plan to go for a drive may include a plan to find my car keys.
  - **Events**: These are triggers for reactive activity by the agent. An event may update beliefs, trigger plans or modify goals. Events may be generated externally and received by sensors or integrated systems. Additionally, events may be generated internally to trigger decoupled updates or plans of activity.

Types of Agents

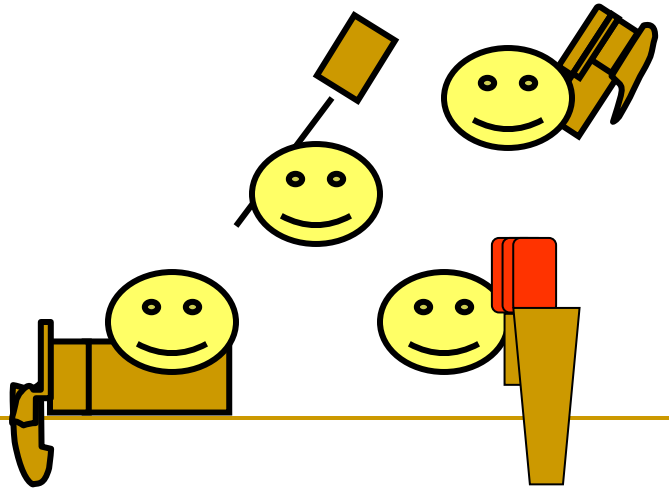
- Logic-based agents
- Reactive agents
- Belief-desire-intention agents
- Layered architectures

Today

- Web of knowledge
- Why Agent?
- What is an agent
- Features of Intelligent Agent
- Types of Agents
  - Logic-based agents
  - Reactive agents
  - Belief-desire-intention agents
  - Layered architectures
- Multiagent Systems

Multi-Agent Systems (MAS) Main idea

- ***Cooperative*** working environment comprising synergistic software components can cope with complex problems.

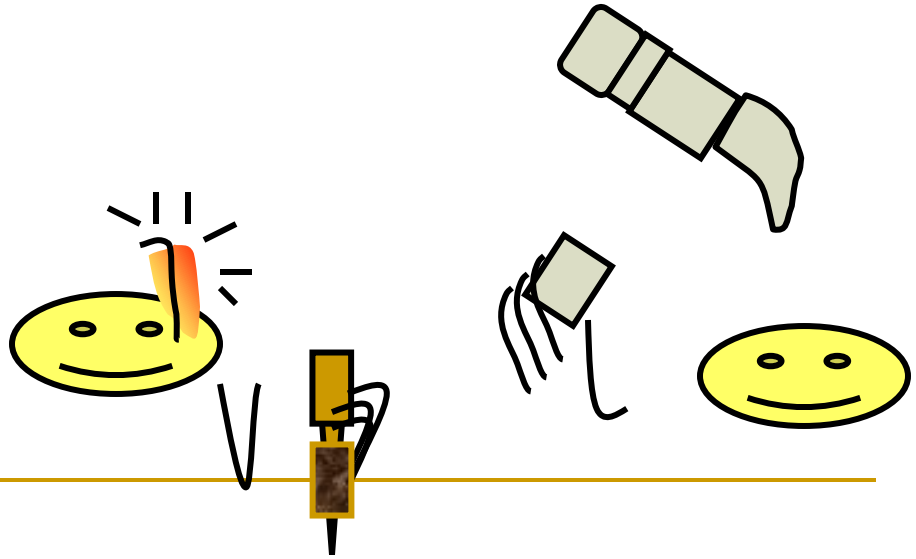


Rationality

- Principle of ***social rationality*** by Hogg et al.:
  "Within an agent-based society, if a socially rational agent can perform an action so that agents' join benefit is greater than their joint loss then it may select that action."

  EU(a) = f(IU(a), SU(a))

  where:
  EU(a): expected utility of action a
  IU(a): individual utility
  SU(a): social utility



Agent platform

- A platform is a *place* which provides services to an Agent

  - Services: Communications, Resource Access, Migration, Security, Contact Address Management, Persistence, Storage, Creation etc.

  - Middleware
    - Fat AOM (Agent Oriented Middleware): lots of services and lightweight agents
    - Thin AOM: few services and very capable agents

Mobile Agent

- The Mobile Agent is the entity that moves between platforms
  - Includes the state and the code where appropriate
  - Includes the responsibilities and the social role if appropriate (I.e. the agent does not usually become a new agent just because it moved.)

Conclusion

- The concept of ***agent*** is associated with many different kinds of software and hardware systems. Still, we found that there are similarities in many different definitions of agents.
  - Unfortunately, still, the meaning of the word "agent" depends heavily on who is speaking.
- There is no consensus on what an agent is, but several key concepts are fundamental to this paradigm. We have seen:
  - The main characteristics upon which our agent definition relies
  - Several types of software agents
  - In what an agent differs from other software paradigms

Discussion

- Who is legally responsible for the actions or agents?
- How many tasks and which tasks the users want to delegate to agents?
- How much can we trust in agents?
- How to protect ourselves of erroneously working agents?

Today

- Web of knowledge
- Why Agent?
- What is an agent
- Features of Intelligent Agent
- Types of Agents
  - Logic-based agents
  - Reactive agents
  - Belief-desire-intention agents
  - Layered architectures
- Multiagent Systems