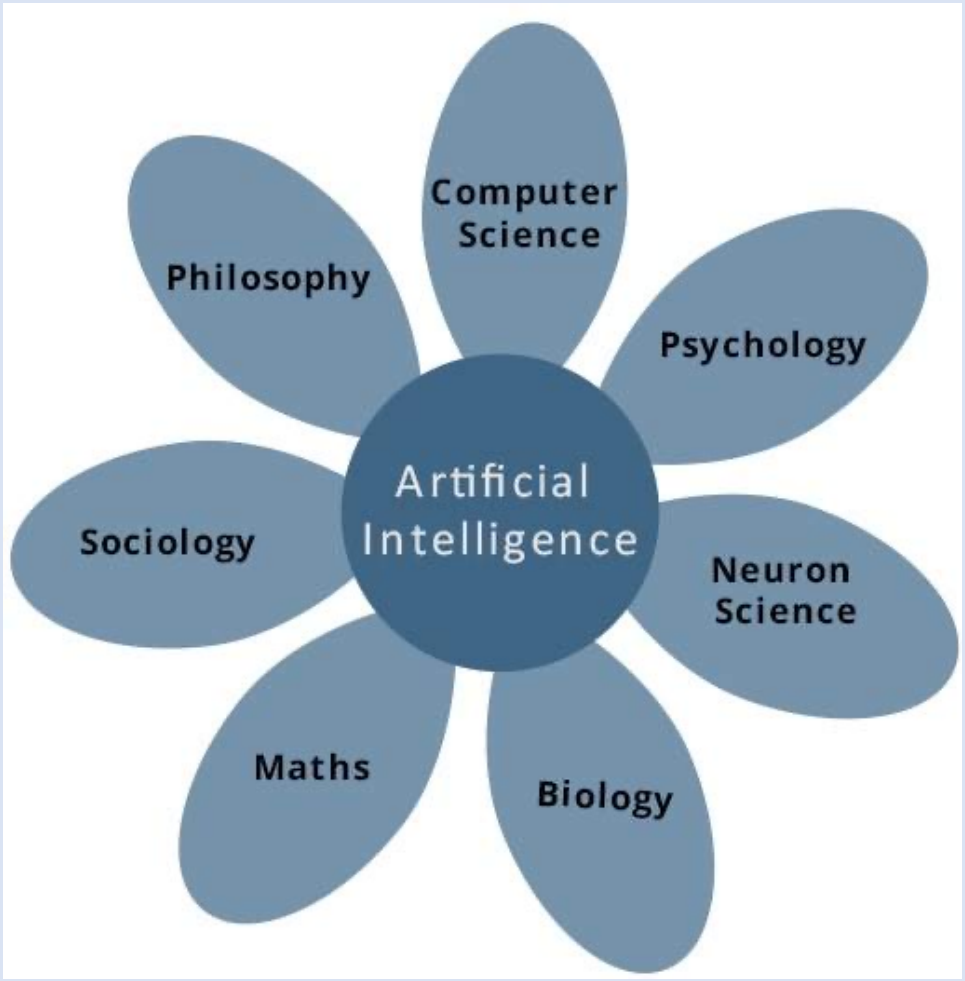
Artificial Intelligence

Multi-Agent Systems (MAS)

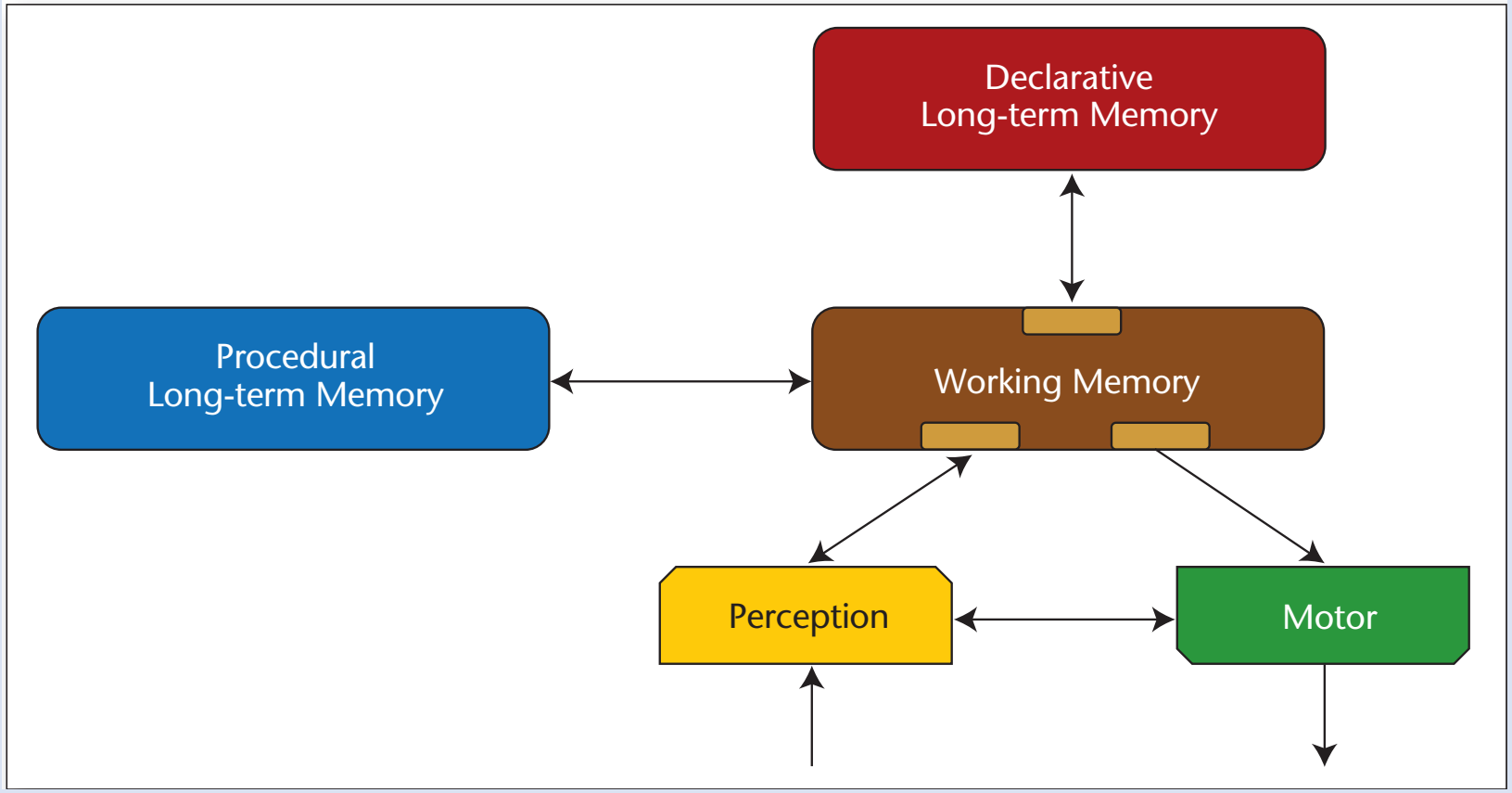
Dr. Bilgin Avenoğlu

AI Disciplines



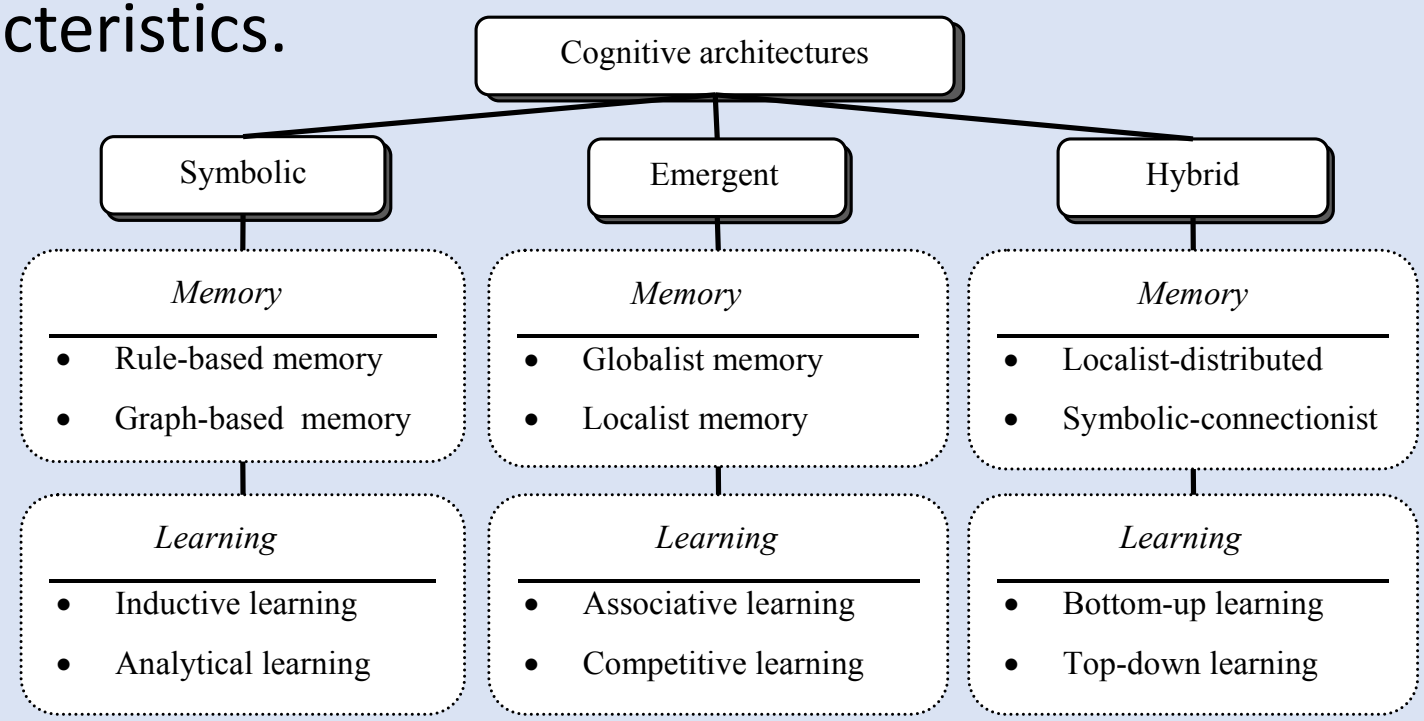
A Standard Model of the Mind

- The mind is not an undifferentiated pool of information and processing, but is built of independent modules that have distinct functionalities.



Cognitive Architectures

- Cognitive architecture (CA) research is a theme in AI that models the main factors participated in our thinking and decision and concentrates on the relationships among them.
- CA mostly refers to the computational model simulating human's cognitive and behavioral characteristics.



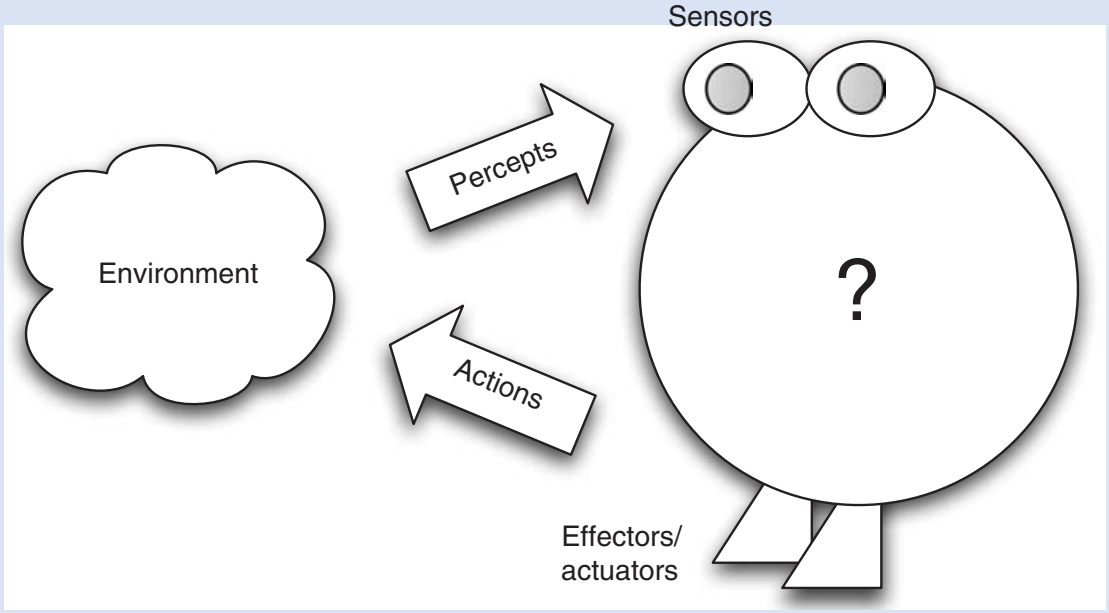
Abbreviation	Perception	Actuation	Memory	Reasoning	Learning	Attention	Emotion	Planning	Interaction	Motivation	Proposed year
4CAPS	√		√	√	√						2006
4D/RCS	√	√						√			2000
ACT-R	√	√	√	√	√	√					2004
ADAPT	√	√						√			2004
ARCADIA	√	√	√			√					2015
ARDIS	√	√									2009
ASMO	√	√	√			√					2010
BBD	√	√	√					√			2005
BECCA	√	√			√			√			2009
BDI	√	√	√	√							1998
CARACaS	√	√		√					√		2011
Casimir	√		√	√							2011
CELTS	√	√	√		√	√	√				2011
CERA-CRANIUM	√	√	√	√			√				2009
Cerebus	√	√	√					√			2000
CHARISMA	√	√	√		√				√	√	2011
CHREST	√		√		√	√					1992
CLARION	√	√	√	√	√						1998
COGNET/iGEN	√	√	√		√	√					2000
Cognitive Symmetry Engine	√	√	√								2011
CoJACK	√	√	√	√			√	√			2009
Companions	√		√	√	√				√		2006
Cosy	√	√	√		√			√	√		2004
CORTEX	√	√	√	√					√		2014
DAC	√	√	√		√			√		√	2012
DIARC	√	√	√	√			√		√		2006
DiPRA	√	√	√	√				√			2005
DSO-CA	√	√	√	√							2011
EMILE	√	√	√	√			√	√			2000
EPAM	√	√	√		√						1995
EPIC	√	√	√		√						1994
GLAIR	√	√		√				√			2009
GMU-BICA	√		√	√							2005
HTM	√	√	√		√						2007
ICARUS	√	√	√	√							2004
iCub	√	√	√			√	√			√	2004
Ikon Flux	√		√	√	√						2009
IMPRINT	√		√	√							2002
Leabra	√		√								2012
LIDA	√	√	√			√					2006
MAMID	√	√	√	√		√	√				2000
MDB	√	√	√		√						2006
MicroPsi	√	√	√		√		√			√	2007
MusiCog	√		√		√						2012
NARS			√	√	√						2013
Novamente Engine	√	√	√	√							2007
OpenCogPrime	√	√	√	√	√	√	√	√	√	√	2013
PMFserv			√				√		√		2001
Pogamut	√	√	√								2009
Polyscheme	√		√	√		√					2005
R-CAST	√	√	√	√					√		2008
REAPER	√	√	√						√		2001
REM	√	√	√	√							2001
SAL	√	√	√	√							2008
SEMLC	√	√	√		√			√		√	2011
Shruti	√	√	√	√							2000
SiMA	√	√	√	√						√	2013
Soar			√	√	√						1987

MAS Definition

- Types of software:
  - functional programs: takes some input, chews over this input, and then on the basis of this, produces some output and halts.
    - 'read a list of numbers and print the average'
- Unfortunately, many programs do not have this simple *input-compute-output* operational structure.
  - many of the systems we need to build in practice have a 'reactive' flavour, in the sense that they have to maintain a long- term, ongoing interaction with their environment;
    - operating systems, process control systems, online banking systems, web servers
- Agent-based systems: active, purposeful producers of actions
  - they are sent out into their environment to achieve goals for us, and we want them to actively pursue these goals, figuring out for themselves how best to accomplish these goals, rather than having to be told in low-level detail how to do it.

MAS Definition

- Agent systems are *situated* in some *environment*.
- Agents are capable of *sensing* their environment (*via sensors*),
- Agents have a repertoire of possible *actions* that they can perform (via *effectors* or *actuators*) in order to modify their *environment*.



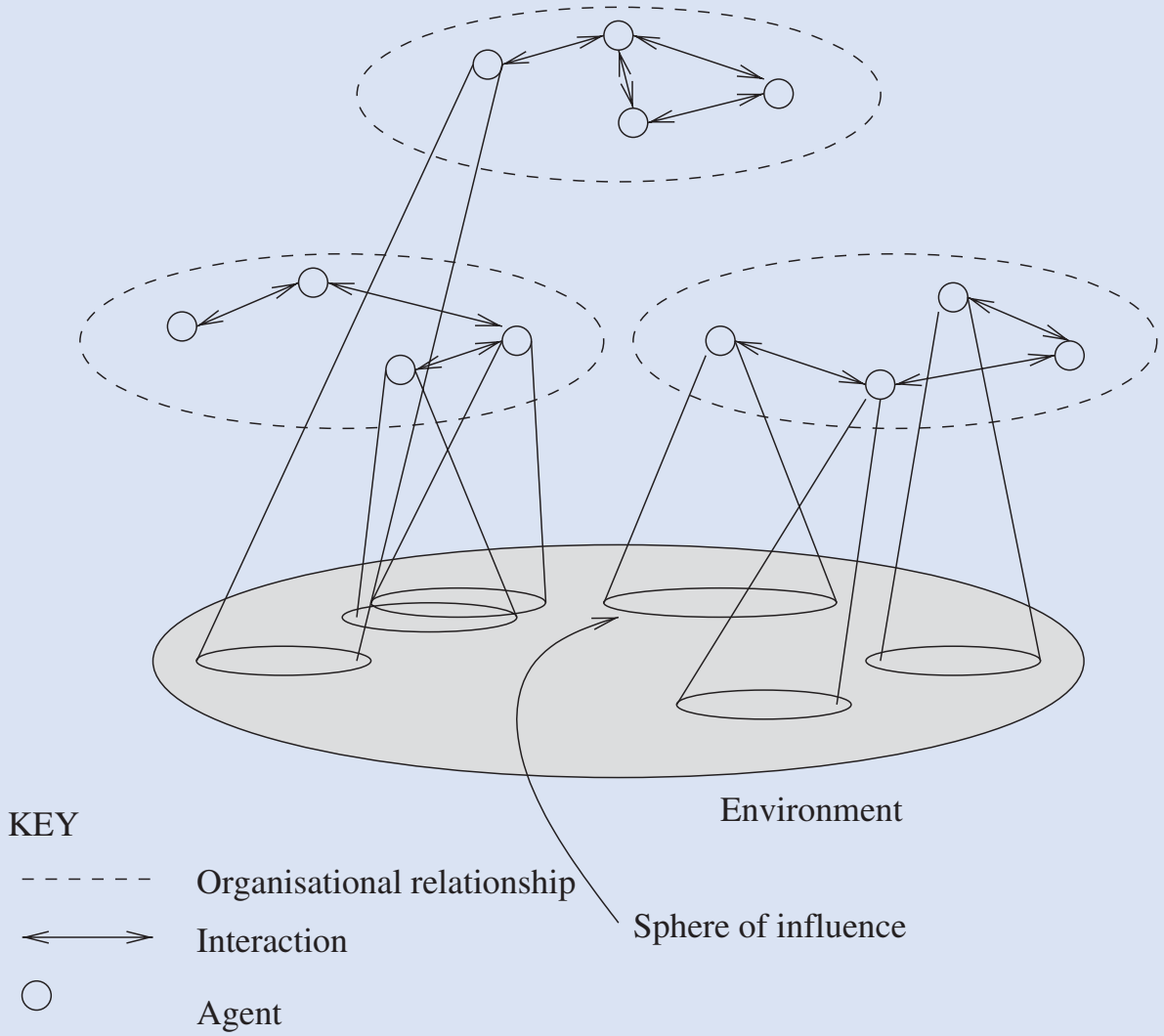
MAS Definition

- agents should have the following properties:
  - autonomy;
  - proactiveness;
    - generating and attempting to achieve goals; not driven solely by events; taking the initiative.
  - reactivity;
    - If a program's environment is guaranteed to be fixed, a program can just execute blindly.
  - social ability.
    - Social ability in agents is the ability to interact with other agents (and possibly humans) via cooperation, coordination, and negotiation.

Level	Definition
1	The computer offers no assistance, human must do it all
2	The computer offers a complete set of action alternatives, and
3	Narrows the selection down to a few, or
4	Suggests one, and
5	Executes that suggestion if the human approves, or
6	Allows the human a restricted time to veto before automatic execution, or
7	Executes automatically, then necessarily informs the human, or
8	Informs him after execution only if he asks, or
9	Informs him after execution if it, the computer, decides to
10	The computer decides everything and acts autonomously, ignoring the human

MAS

- We have talked about agents occupying an environment in isolation.
- In practice, ‘single agent systems’ are rare.
- The more common case is for agents to inhabit an environment which contains other agents, giving a multi-agent system.



The Belief-Desire-Intention (BDI) Agent Model

- The idea that we can talk about computer programs as if they have a ‘mental state’.
  - Beliefs are information the agent has about the world.
  - Desires are all the possible states of affairs that the agent might like to accomplish.
  - Intentions are the states of affairs that the agent has decided to work towards.
    - Intentions may be goals that are delegated to the agent, or may result from considering options: we think of an agent looking at its options and choosing between them.

- We can imagine our agent starting with some delegated goal, and then considering the possible options that are compatible with this delegated goal; the options that it chooses are then intentions, which the agent is committed to.

Practical Reasoning

- How does an agent with beliefs, desires and intentions go from these to its actions?
  - a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes.
- Human practical reasoning seems to consist of two distinct activities: deliberation (fixing upon states of affairs that we want to achieve, i.e. our intentions); and means-ends reasoning (deciding how to act so as to bring about our intentions).

A Computational Model of BDI Practical Reasoning

```
1.  B ← B0;        /* B0 are initial beliefs */
2.  I ← I0;        /* I0 are initial intentions */
3.  while true do
4.    get next percept ρ via sensors;
5.    B ← brf(B, ρ);
6.    D ← options(B, I);
7.    I ← filter(B, D, I);
8.    π ← plan(B, I, Ac); /* Ac is the set of actions */
9.    while not (empty(π) or succeeded(I, B) or impossible(I, B)) do
10.     α ← first element of π;
11.     execute(α);
12.     π ← tail of π;
13.     observe environment to get next percept ρ;
14.     B ← brf(B, ρ);
15.     if reconsider(I, B) then
16.       D ← options(B, I);
17.       I ← filter(B, D, I);
18.     end-if
19.     if not sound(π, I, B) then
20.       π ← plan(B, I, Ac)
21.     end-if
22.   end-while
23. end-while
```

Figure 2.1 Overall control loop for a BDI practical reasoning agent.

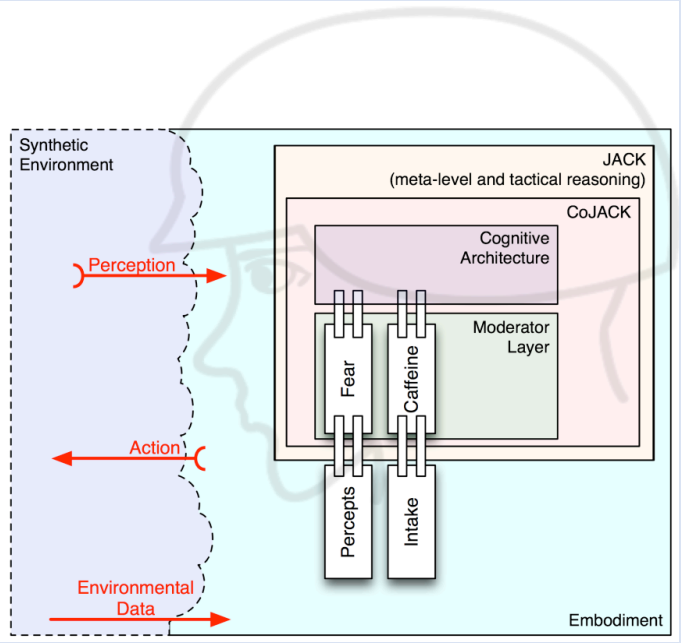
The Procedural Reasoning System (PRS)

- PRS is originally developed at Stanford Research Institute by Michael Georgeff and Amy Lansky, was perhaps the first agent architecture to explicitly embody the BDI paradigm, and has proved to be one of the most durable approaches to developing agents to date.
- PRS has been re-implemented several times since the mid-1980s,
  - JACK is a commercially available programming language of PRS ,
  - Agent Oriented Software – the Australian AI company since 1997.

CoJACK

- CoJACK™ is cognitive architecture used for modelling the variation in human behaviour.
- It is used in simulation systems to underpin virtual actors.
- Humans share a common cognitive architecture and physiology, and that variation results from individual differences in knowledge and the values of the architecture's and physiology's parameters.

differences across individuals, differences across time on a task within an individual, and the effects of context on a task, such as time pressure, heat, sleep patterns, and food, drink, and psychoactive substances such as caffeine.



AgentSpeak and Jason

- In 1996, Anand Rao created a logic-based agent programming language based on the BDI architecture and named it AgentSpeak(L).
- This became a highly cited paper in the multi-agent systems literature.
- AgentSpeak was an abstract agent programming language aimed to help the understanding of the relation between practical implementations of the BDI architecture such as PRS and the formalisation of the ideas behind the BDI architecture using modal logics.
- Jason is an interpreter for an extended version of AgentSpeak.
  - It implements the operational semantics of that language, and provides a platform for the development of multi-agent systems, with many user-customisable features.

Examples

Jason .mas2j project

JASONPROJECT

```
MAS bdi_hw {

    agents: bob;
    // file bob.asl at project directory

}
```

```
happy(bob).

!say(hello).

+!say(X) : happy(bob) <- .print(X).
```

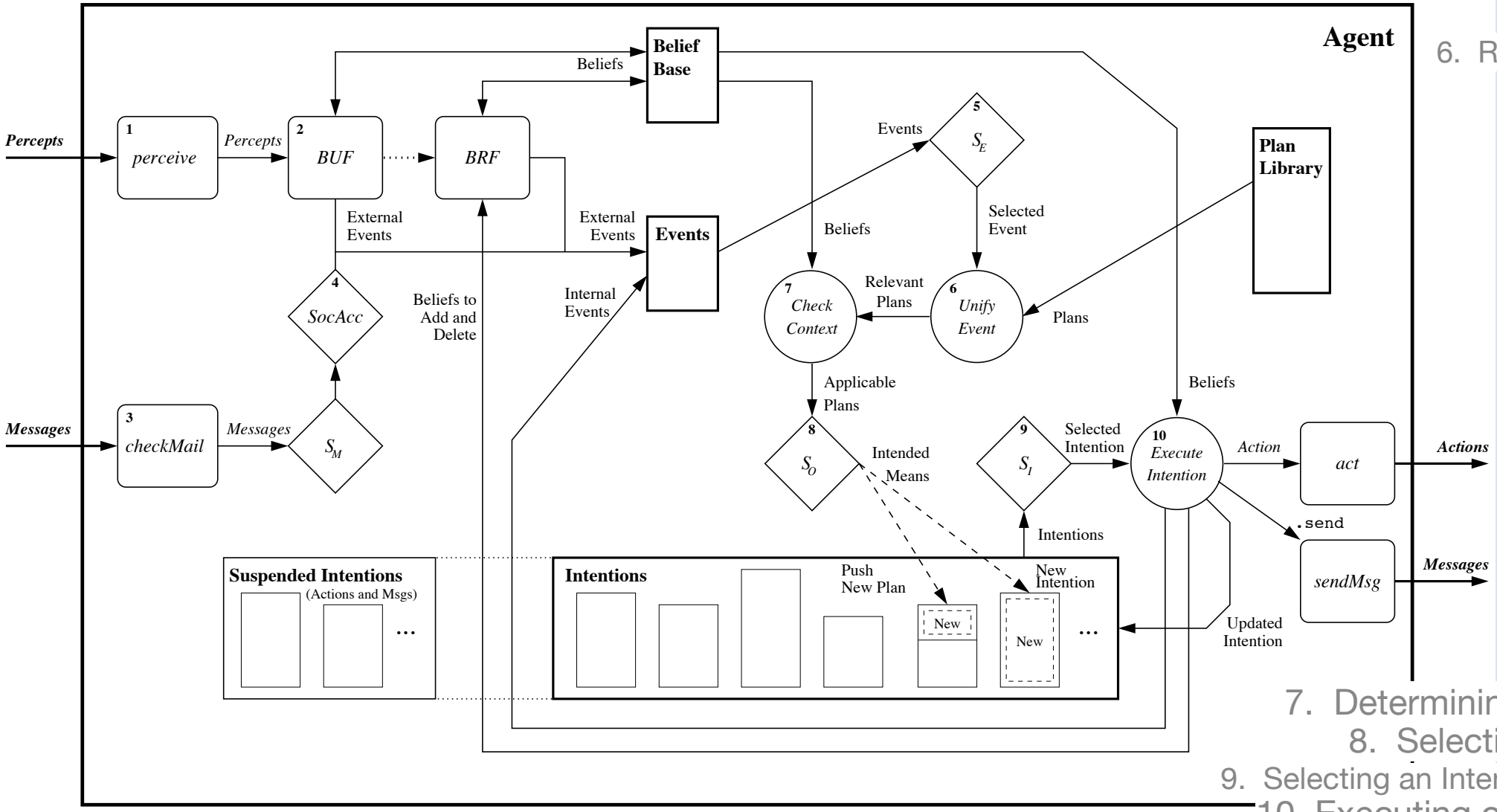
```
+green_patch(Rock)
   :  not battery_charge(low)
   <- ?location(Rock,Coordinates);
      !at(Coordinates);
      !examine(Rock).

+!at(Coords)
   :  not at(Coords)
      & safe_path(Coords)
   <- move_towards(Coords);
      !at(Coords).

+!at(Coords) ...
```

Jason Reasoning Cycle

1. Perceiving the Environment
2. Updating the Belief Base
3. Receiving Communication from Other Agents
4. Selecting 'Socially Acceptable' Messages
5. Selecting an Event
6. Retrieving all Relevant Plans
7. Determining the Applicable Plans
8. Selecting one Applicable Plan
9. Selecting an Intention for Further Execution
10. Executing one step of an Intention



Agent Communication

- Agent communication in multi-agent systems is typically based on the speech-act theory
- Speech-act theory starts from the principle that language is action:
  - a rational agent makes an utterance in an attempt to change the state of the world, in the same way that an agent performs 'physical' actions to change the state of the world.
  - what distinguishes speech acts from other ('non-speech') actions is that the domain of a speech act –the part of the world that the agent wishes to modify through the performance of the act – is typically limited the mental state(s) of the hearer(s) of the utterance.
  - thus examples of speech acts might be to change your beliefs, desires or intentions.

Agent Communication

- The various types of speech acts are generally referred to as ‘performatives’ in the context of agent communication.
  - representatives – such as informing, e.g. ‘It is raining’;
  - directives – attempts to get the hearer to do something, e.g. ‘Please make the tea’;
  - commisives – which commit the speaker to doing something, e.g. ‘I promise to…’;
  - expressives – whereby a speaker expresses a mental state, e.g. ‘Thank you!’;
  - declarations – such as declaring war or christening.

- performative = request
  content = ‘the door is closed’
  speech act = ‘please close the door’
- performative = inform
  content = ‘the door is closed’
  speech act = ‘the door is closed!’
- performative = inquire
  content = ‘the door is closed’
  speech act = ‘is the door closed?’

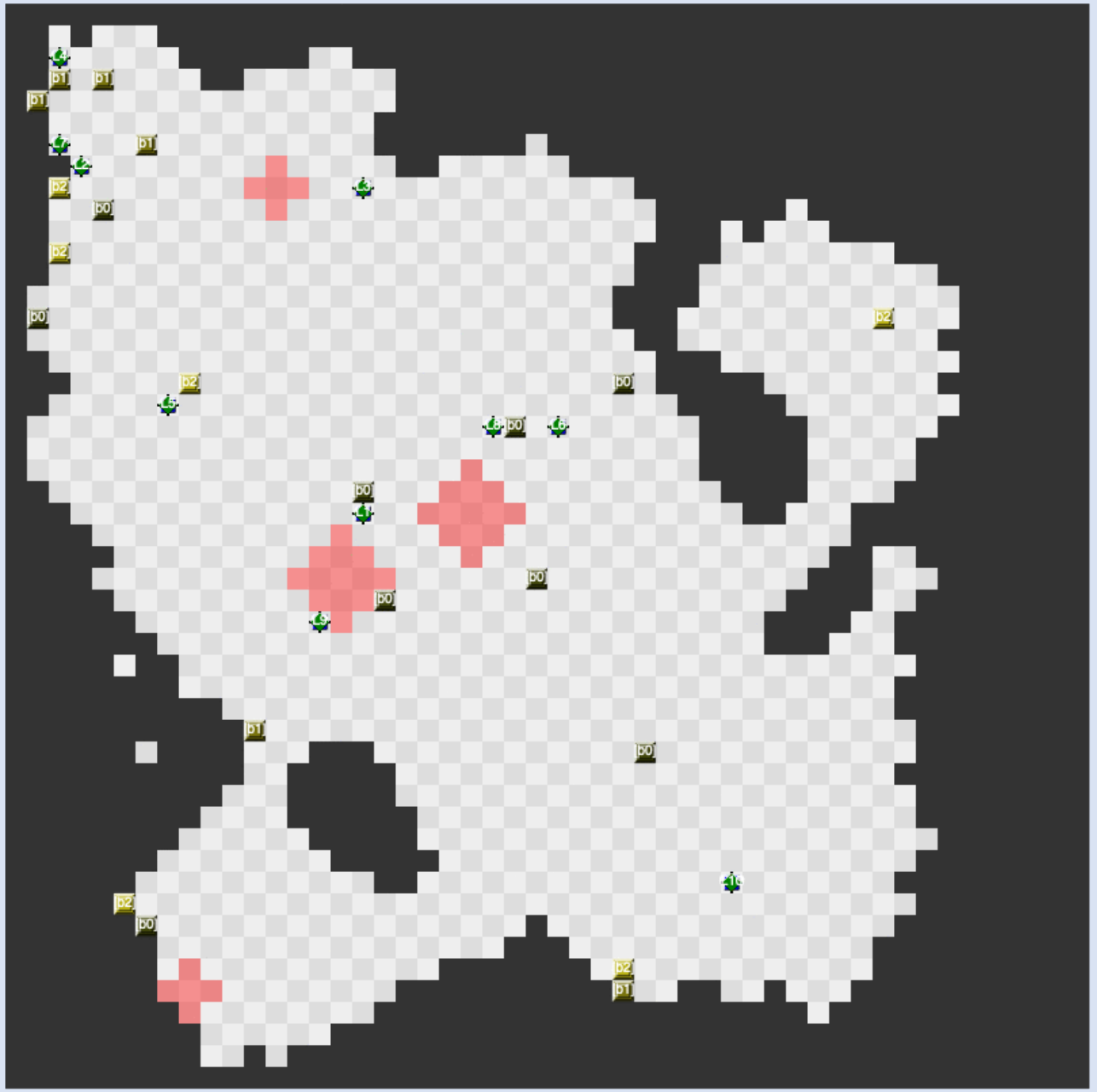
Agent Communication

- The Knowledge Query and Manipulation Language (KQML) was the first attempt to define a practical agent communication language that included high-level (speech-act based) communication

```
(ask-one
   :content  (PRICE IBM ?price)
   :receiver stock-server
   :language LProlog
   :ontology NYSE-TICKS
)
```

- The FIPA standard for agent communication (ACL) was released in 2002.
  - The goal of the FIPA organisation was to develop a coherent collection of standards relating to agent communication, management and use, and the agent communication standard was just one component of the FIPA suite.

The Multi-Agent Programming Contest



The MASSim Platform

Server: Unified scenario implementation

Agents: Connect remotely (sockets)

- Receive percepts
  (state of the visible environment)
- Deliberate and send actions back

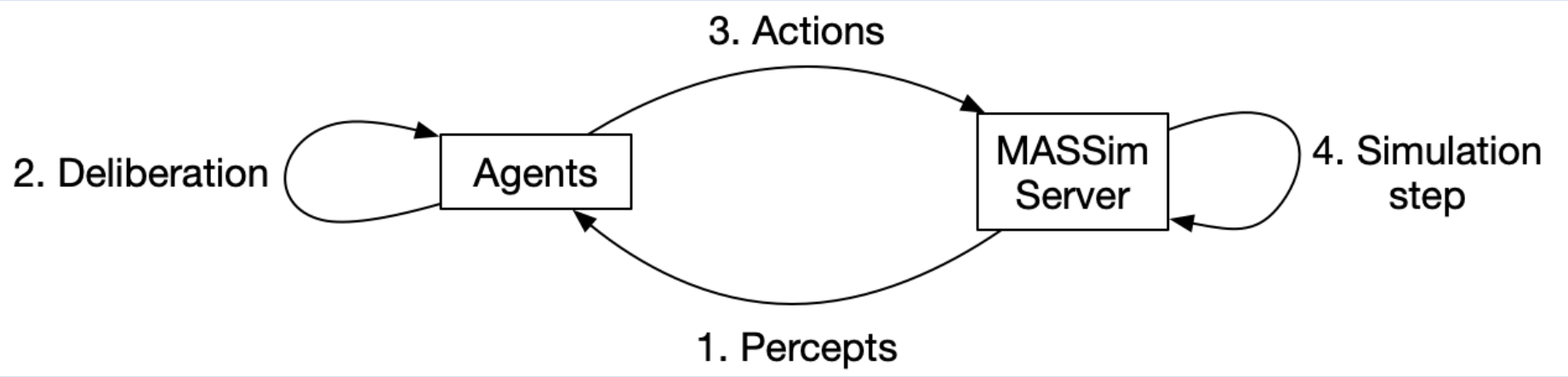
Simulation: Discrete steps

- Actions of all agents processed after each step

Tournament

- Games between 2 teams of agents (as of now)
- Each team plays against each other
- Deliberation: 4 seconds per step for network roundtrip and action computation
- 3 simulations with different parameters per match
  - environment characteristics
  - agent characteristics (features, number, . . .)

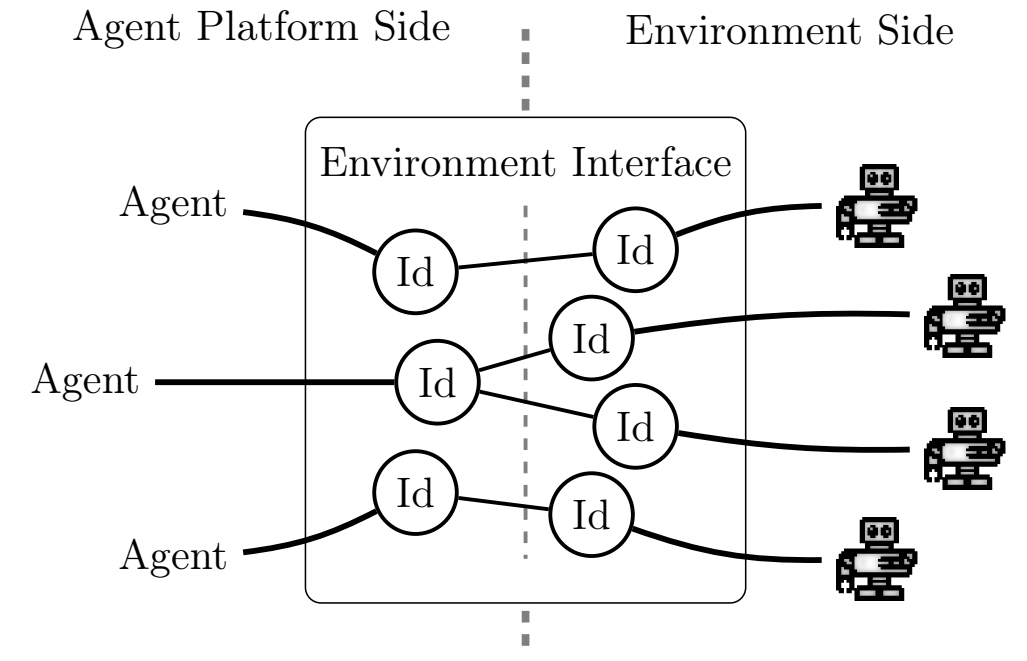
The MASSim Platform

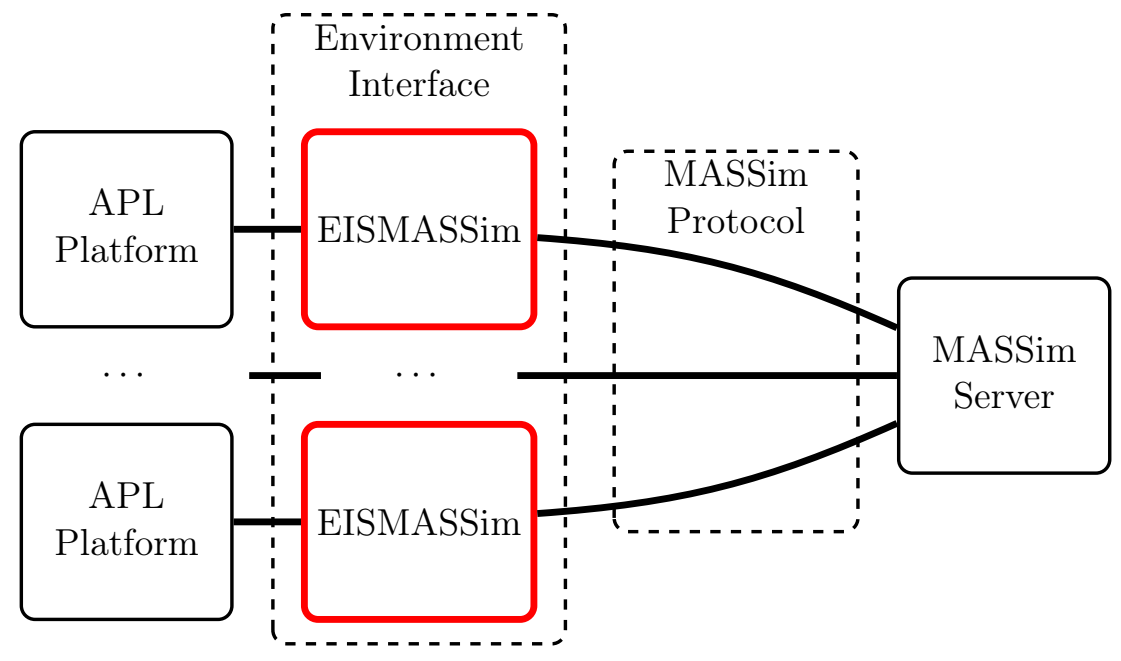


Features in the MASSim

- The MASSim server is implemented in Java
- Highly configurable
  - Simulations
  - Steps
  - Teams
  - ...
- Also, it provides an implementation of EIS [2, 3] (EISMASSim) to Agent Platforms in Java
  - Abstraction to model the agent-environment interaction[1]
  - Handles the communication with the MASSim server

Mapping Entities to Agents





Scenario

The End!