


Agent-Based Modelling and Simulation

Universiteit Antwerpen

Romain Franceschini, Hans Vangheluwe

MoSIS

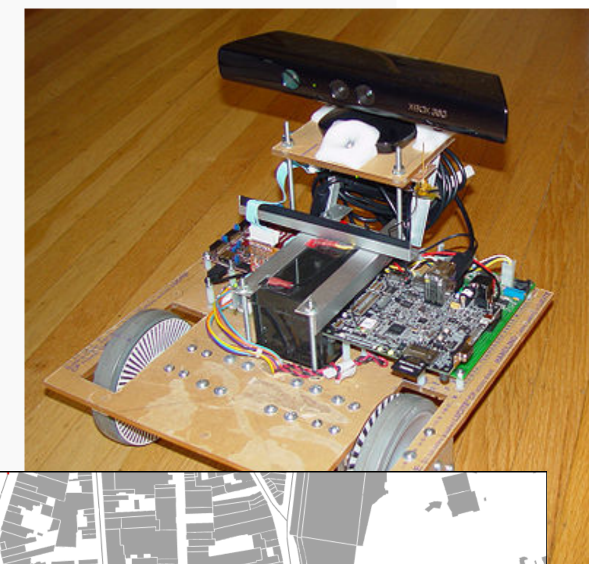
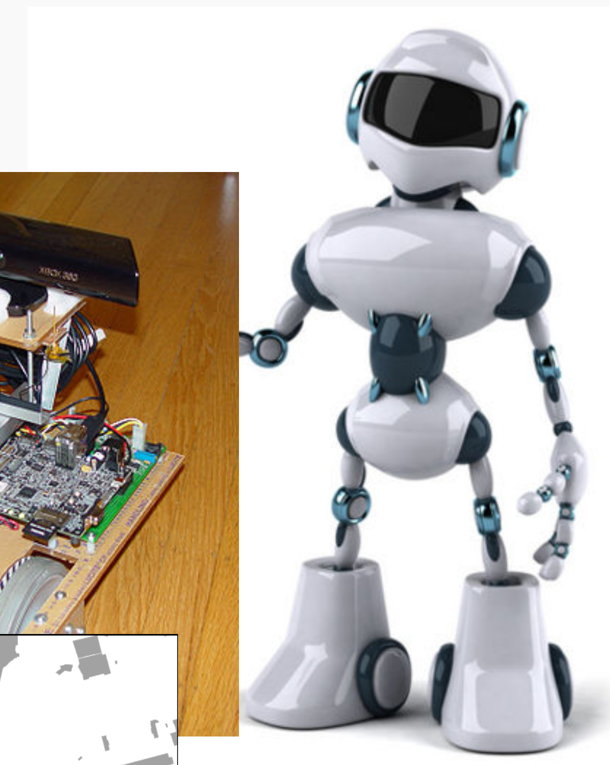
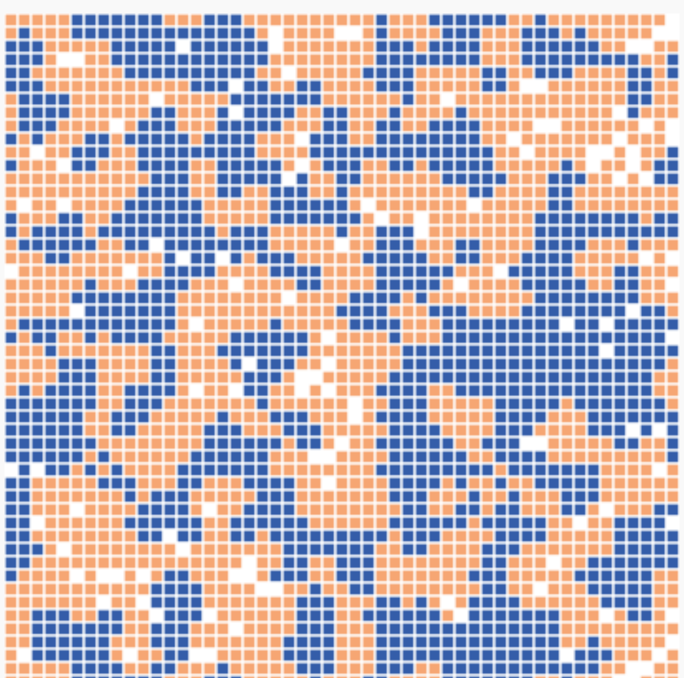
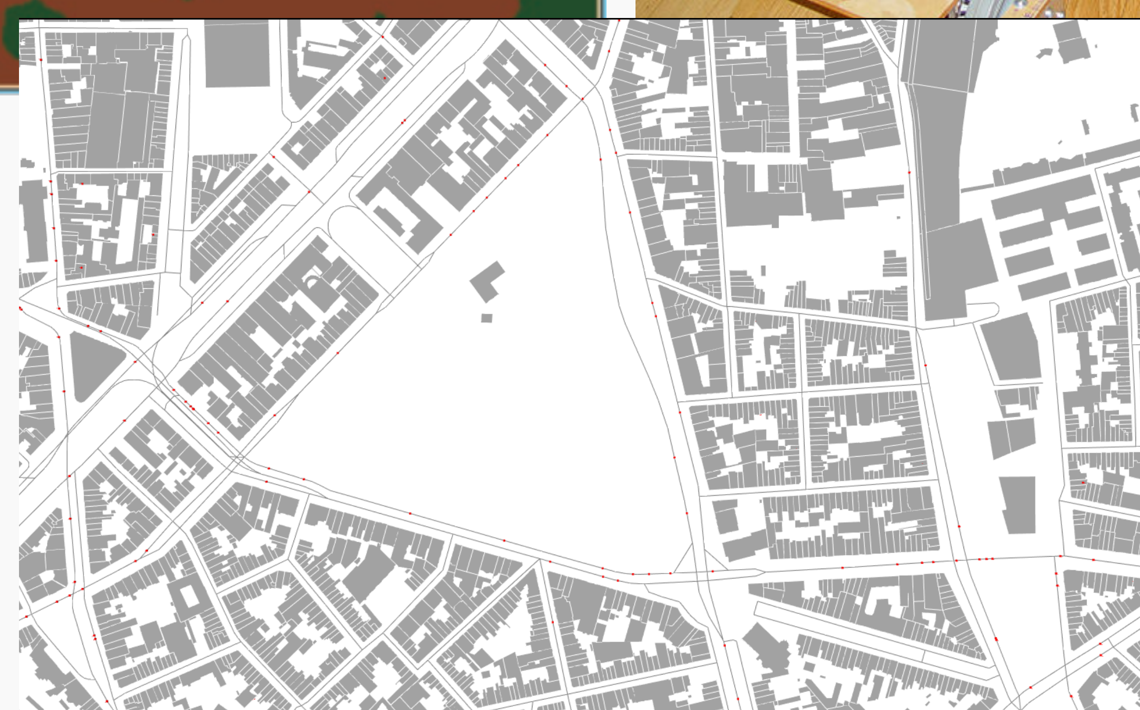
Introduction

Agent Paradigm

The **agent paradigm** is a collection of *concepts* used to tackle *behaviour* of

Distributed, Situated, Interacting, Autonomous and Reactive Systems (agents)

with Dynamic structure

Agent views

Views over agent concepts:

- Programming paradigm (Agent-Oriented Programming)
- Modelling paradigm
  - Multi-Agent System (executed on middleware)
  - Agent-Based Modelling (simulation)

Origins

Distributed Artificial Intelligence

- Collective problem solving
- Communication via information sharing

Artificial Life

- Understanding living systems
- Interactions with environment
- Evolution, survival, adaptation, reproduction, learning processes

Multi-Agent Systems

- Design *autonomous* and *adaptive* agents

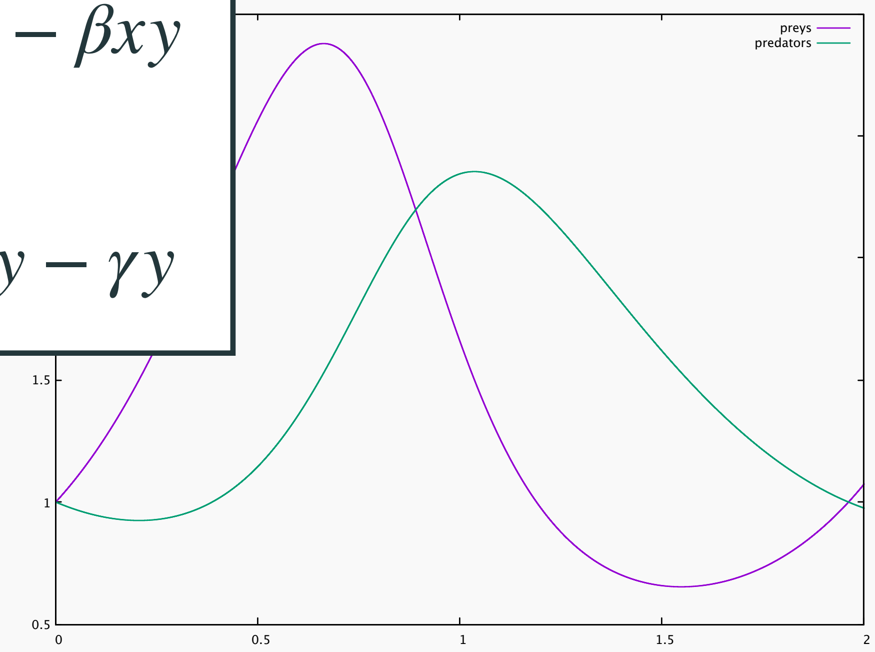
Maes, Pattie. 1995. "Artificial Life Meets Entertainment: Lifelike Autonomous Agents." Communications of the ACM 38 (November): 108–114.

Macroscopic models

- ODEs
- Monte Carlo simulation
- System Dynamics

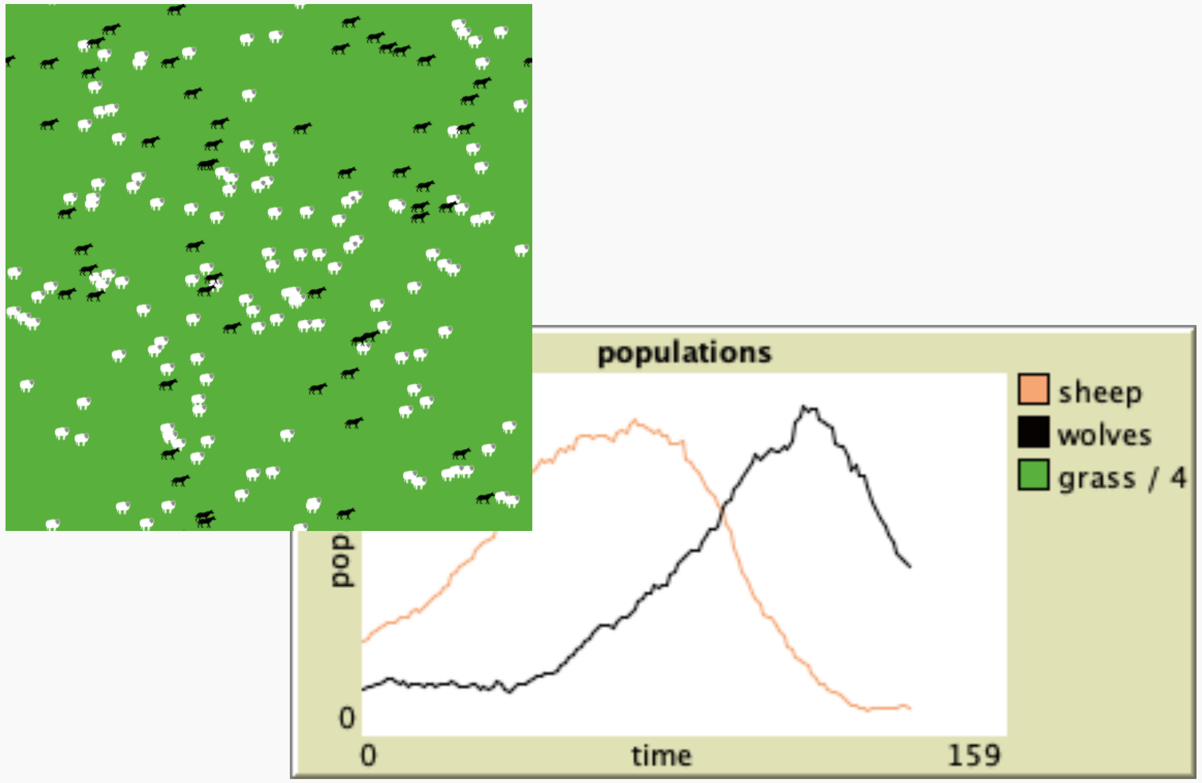
$$\frac{dx}{dt} = \alpha x - \beta xy$$

$$\frac{dy}{dt} = \delta xy - \gamma y$$



Microscopic models

- Cellular Automata
- Individual-Based Models
- Agent-Based Models

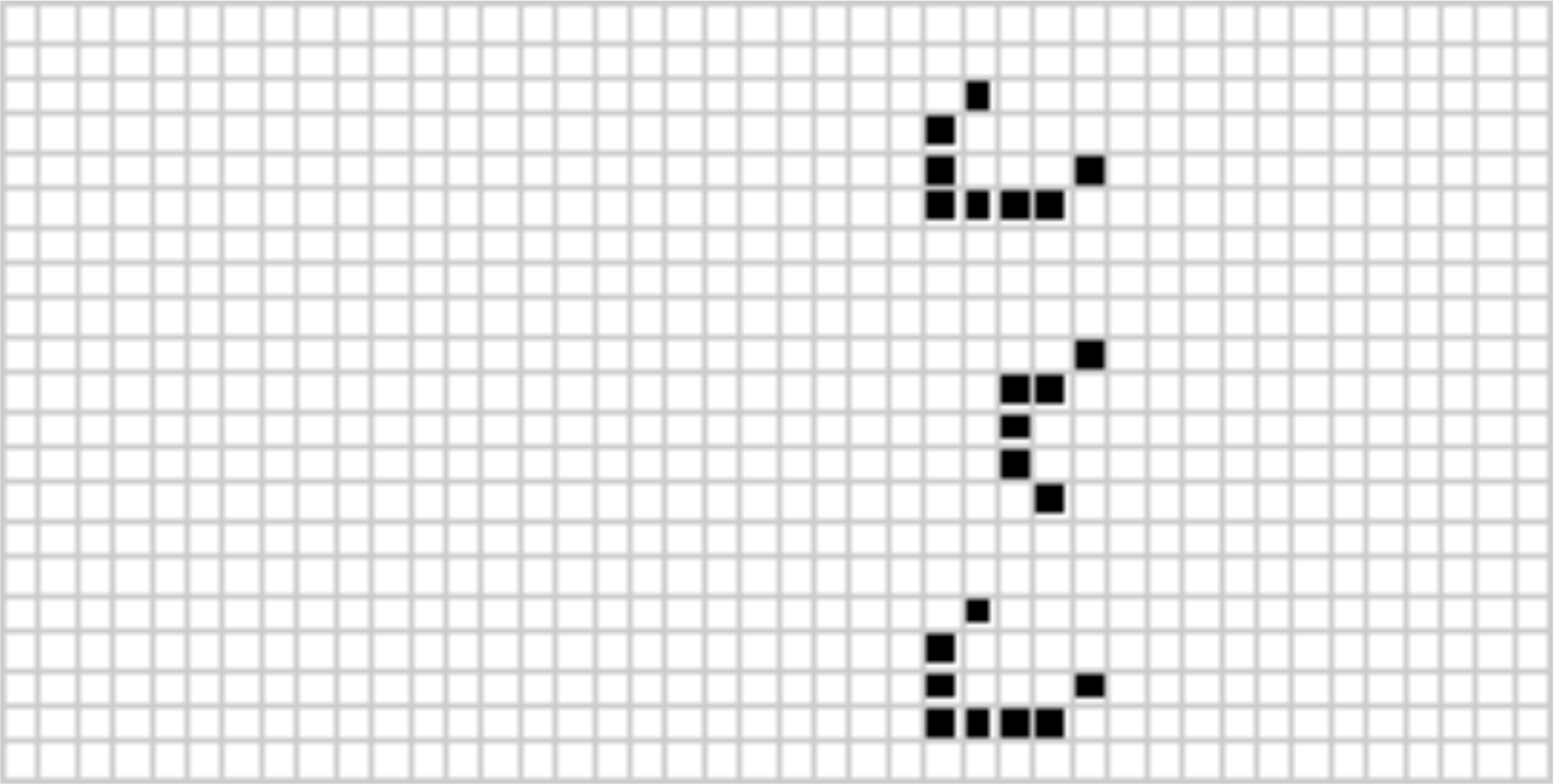


When to use ABM?

- Medium Numbers
- Heterogeneity
- Complex but Local Interactions
- Rich Environments
- Time
- Adaptation

Related formalisms

Idiomatic example: John Conway's Game of Life



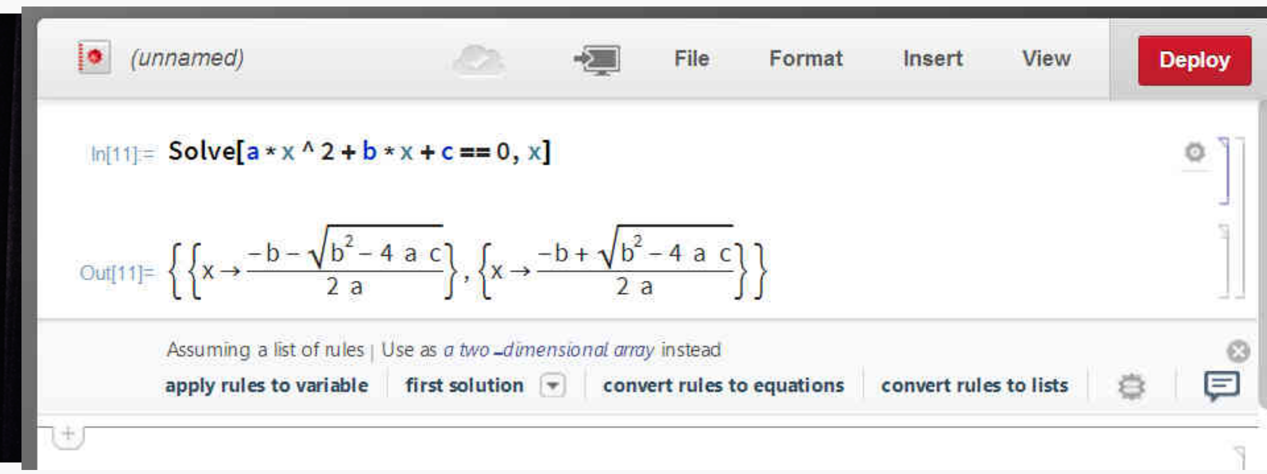
$CA = (T, X, Y, \Omega, S, \delta, \lambda)$, where:

- $T = \mathbb{N}$ the discrete time base.
- X and Y the input and output sets, respectively.
- $\Omega = \{\ldots, \omega : T \to X, \ldots\}$ the set of input segments (ω domain can be $\subseteq T$).
- $S = \times_{i \in C} V_i$ the state set, with:
  - $C = I^D$ the *cell index set* of a D-dimensional grid indexed by I, and
  - V an *homogeneous* value set, such that $\forall i \in C, V_i = V$.
- $\delta : \Omega \times S \to S$ the total transition function

  $(\omega_{]n,n+1]}, \times_{i \in C} v(i)) \mapsto \times_{i \in C} \delta_i(i)$
- $\lambda : S \to Y$ the output function, where Y has a similar structure to S.

Hans Vangheluwe. 2000. "Multi-Formalism Modelling and Simulation.", 82–85.

Universal Cellular Automata



Physica 10D (1984) 1–35
North-Holland, Amsterdam

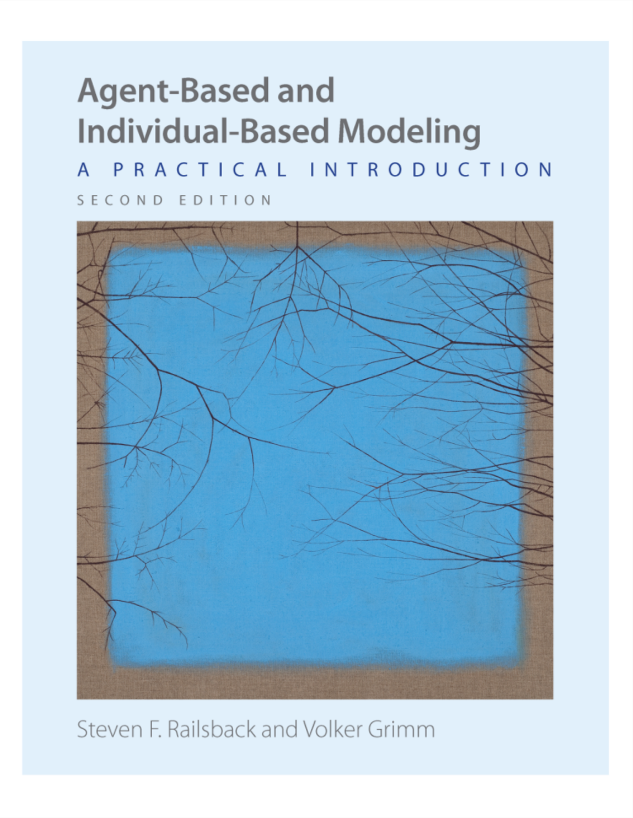
UNIVERSALITY AND COMPLEXITY IN CELLULAR AUTOMATA

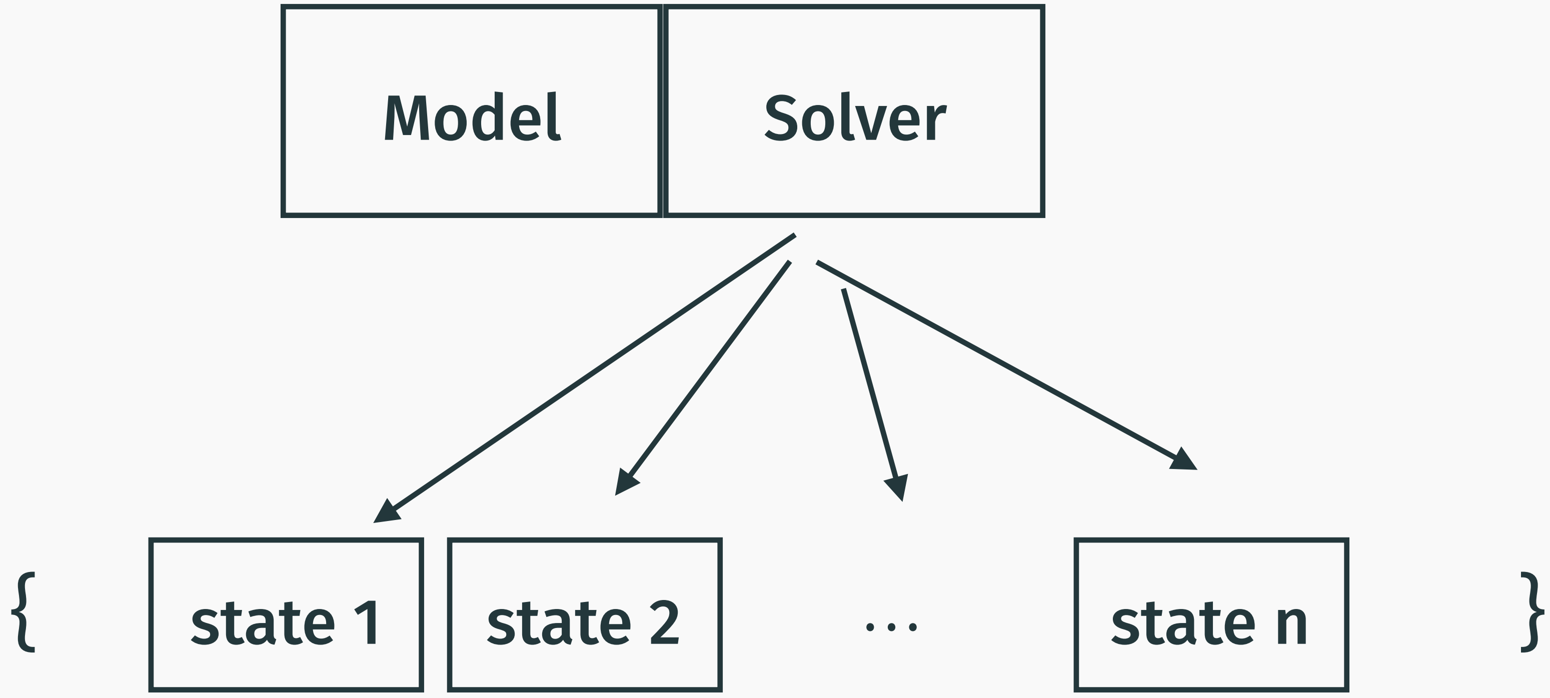
Stephen WOLFRAM*
The Institute for Advanced Study, Princeton NJ 08540, USA

Cellular automata are discrete dynamical systems with simple construction but complex self-organizing behaviour. Evidence is presented that all one-dimensional cellular automata fall into four distinct universality classes. Characterizations of the structures generated in these classes are discussed. Three classes exhibit behaviour analogous to limit points, limit cycles and chaotic attractors. The fourth class is probably capable of universal computation, so that properties of its infinite time behaviour are undecidable.

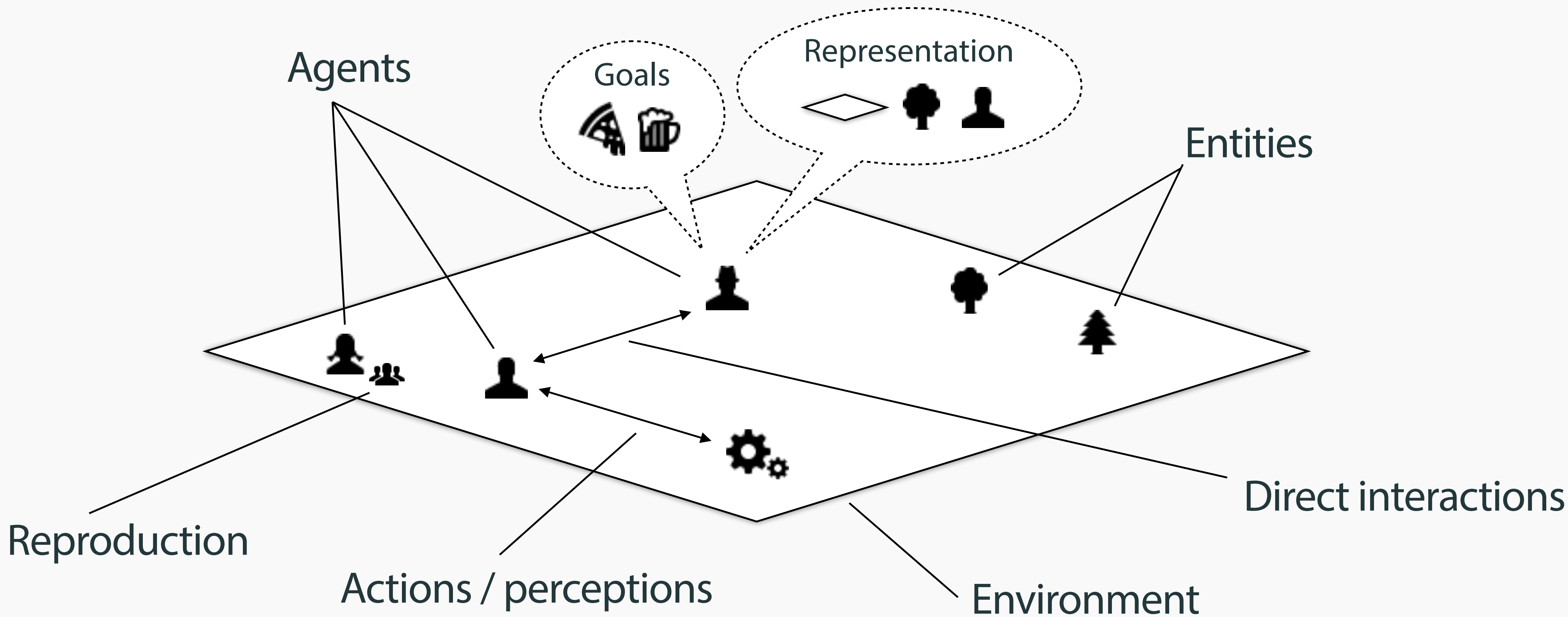
Individual as the main modelling entity

- Set of equations modelling behaviour
- 1 state = 1 entity
- Allow variability in the population
- Evolved over time to ABM-like





Agent-Based Modelling



Modelling Tools

NetLogo

GAMA

repast

cormas

COmmon pool Resources
and Multi-Agent Simulations

MaDKit

The Multiagent Development Kit

TurtleKit

M A S O N

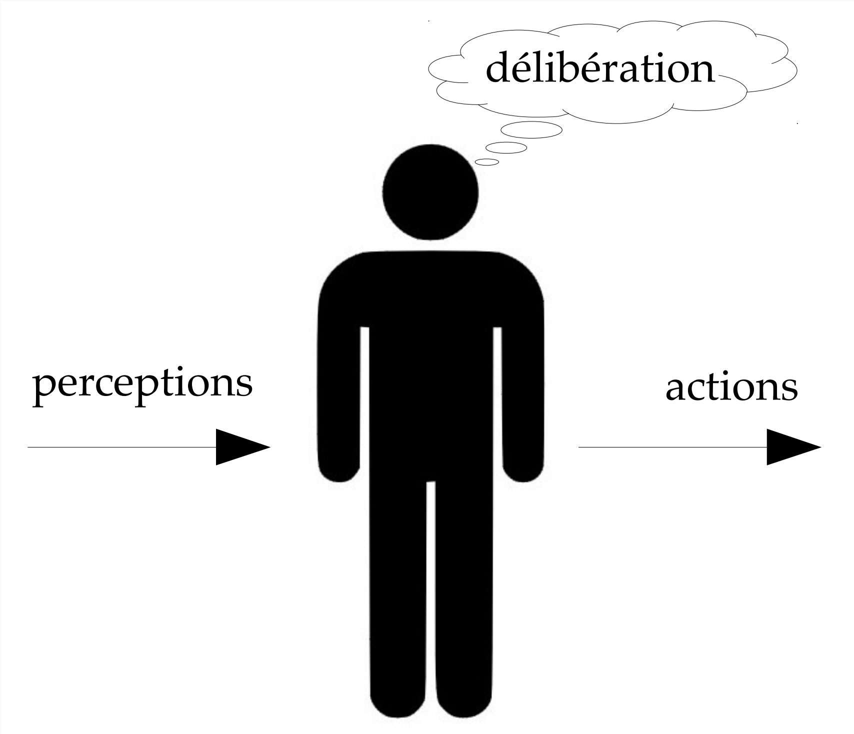
Agent

Properties

- Autonomous
- Social
- Reactive
- Proactive

Two visions of intelligence:

- Cognitive
- Reactive



Wooldridge, Michael J, and Nicholas R Jennings. 1995. “Intelligent Agents: Theory and Practice.” The Knowledge Engineering Review 10 (02): 115–152.

Agent

Agent type	**Properties**
Entity	*Acts upon the environment*
Tropistic (purely reactive)	Perceive, acts
Hysteretic (reactive with state)	Perceive, memorise, acts
Reasoning	Perceive, memorise, reasons, acts

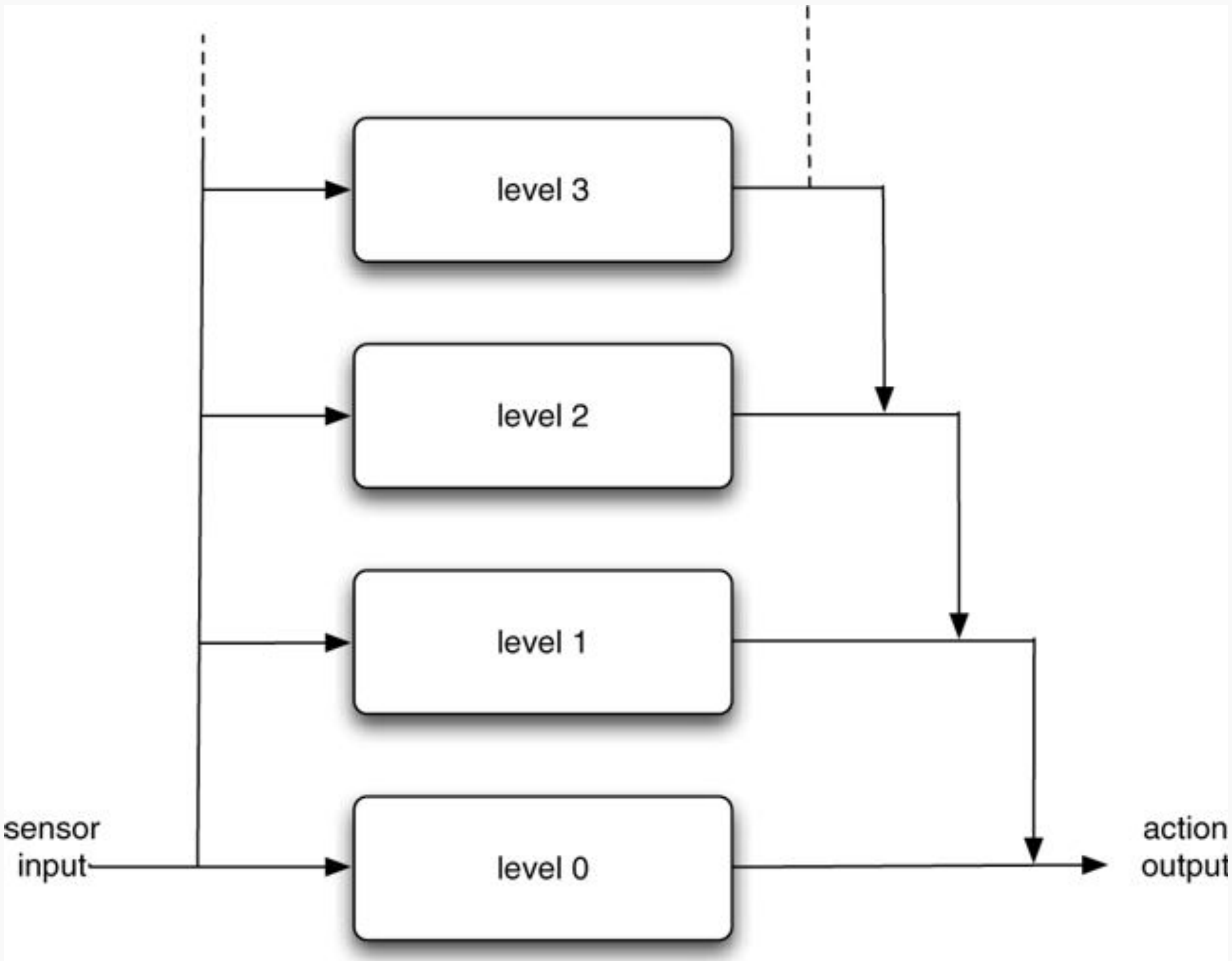
Reactive agents (tropistic and hysteretic) architectures :

- Subsumption
- Situated automata
- Agent network architecture

Reasoning agents :

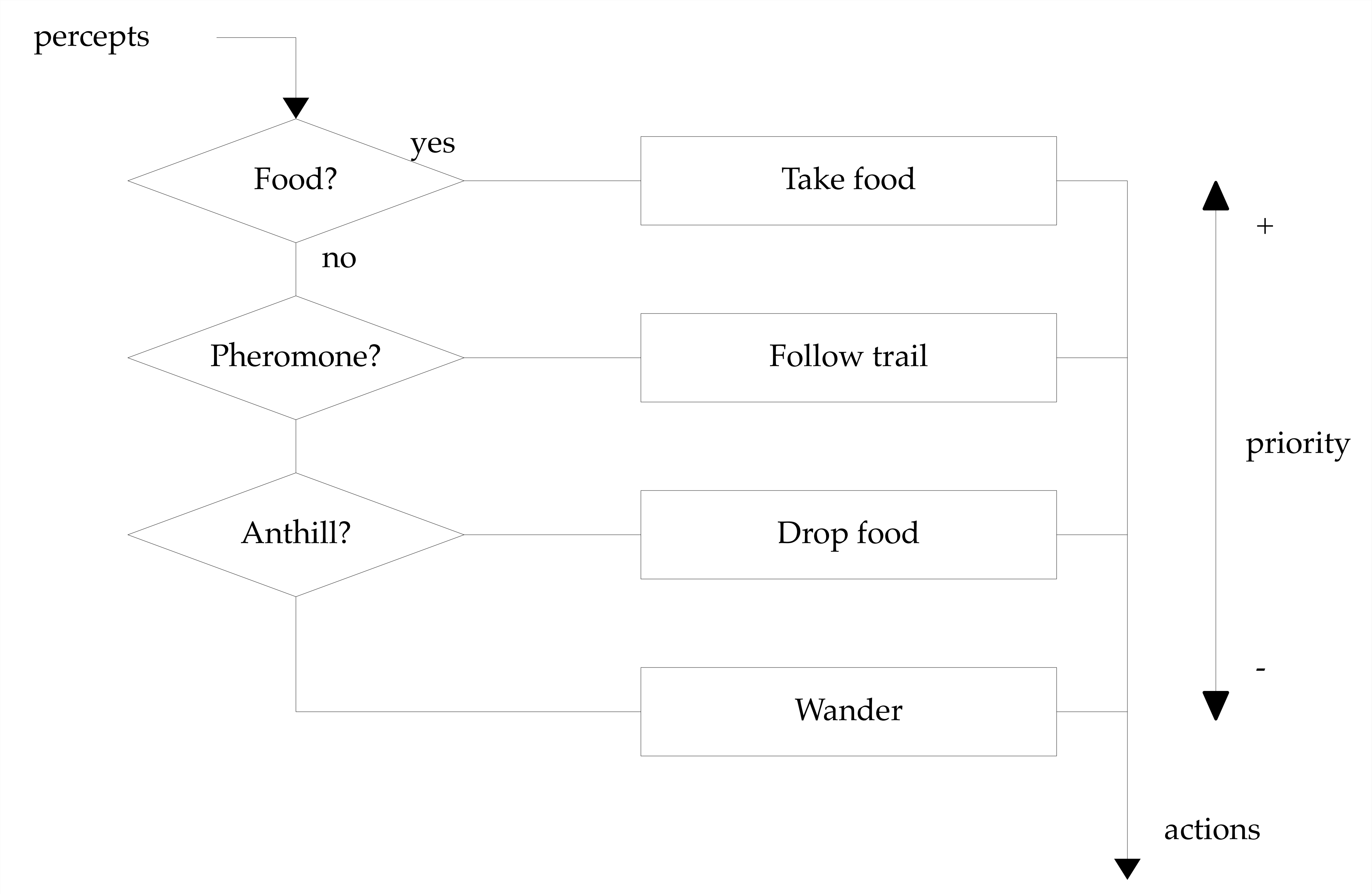
- Logical deduction
- Belief - Desire - Intention

Subsumption architecture



Brooks, Rodney A. 1991. "Intelligence without Representation." Artificial Intelligence 47 (1–3): 139–59. `https://doi.org/10.1016/0004-3702(91)90053-M`.

Subsumption architecture

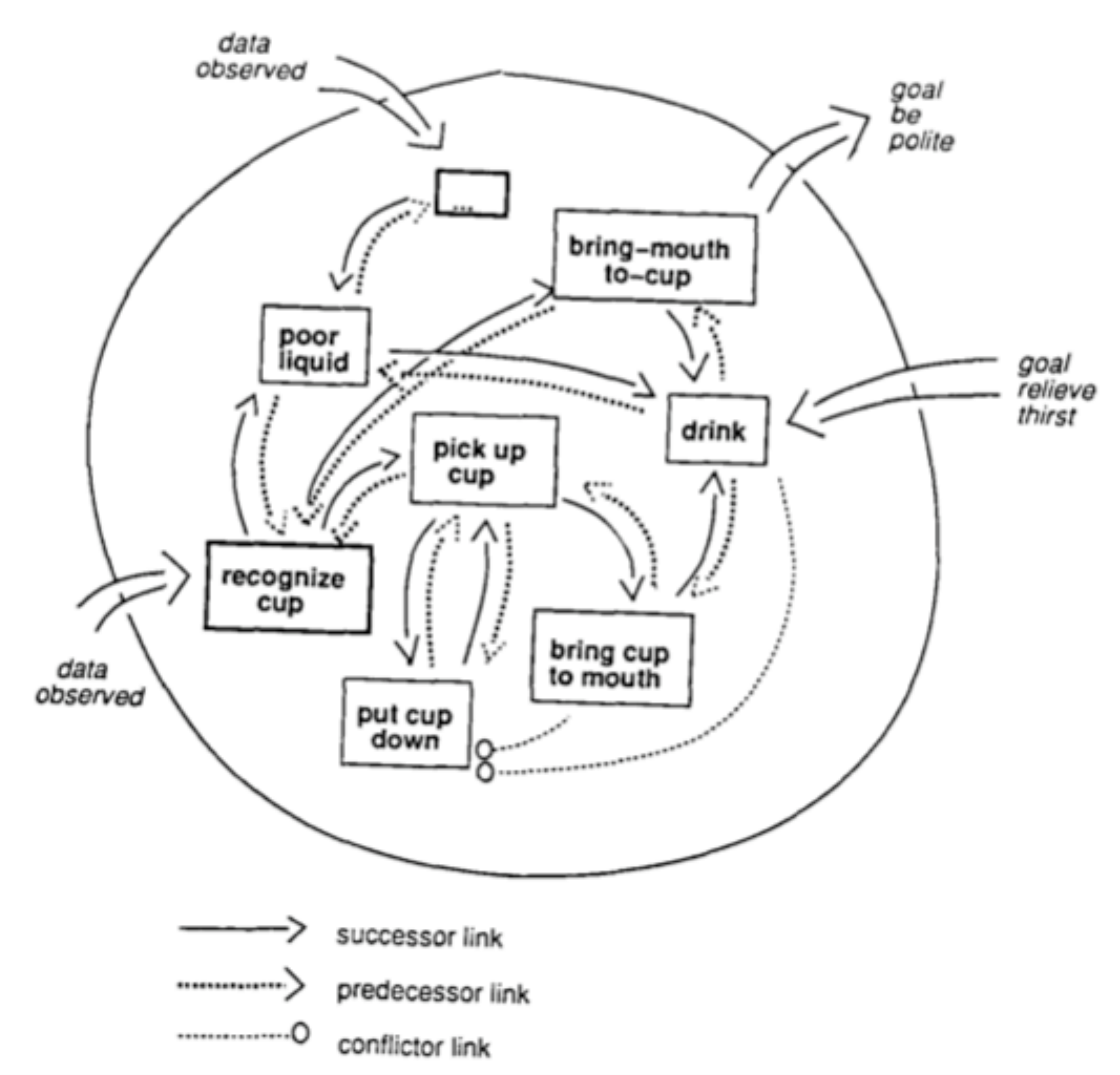


Brooks, Rodney A. 1991. "Intelligence without Representation." Artificial Intelligence 47 (1–3): 139–59. `https://doi.org/10.1016/0004-3702(91)90053-M`.

Agent network architecture

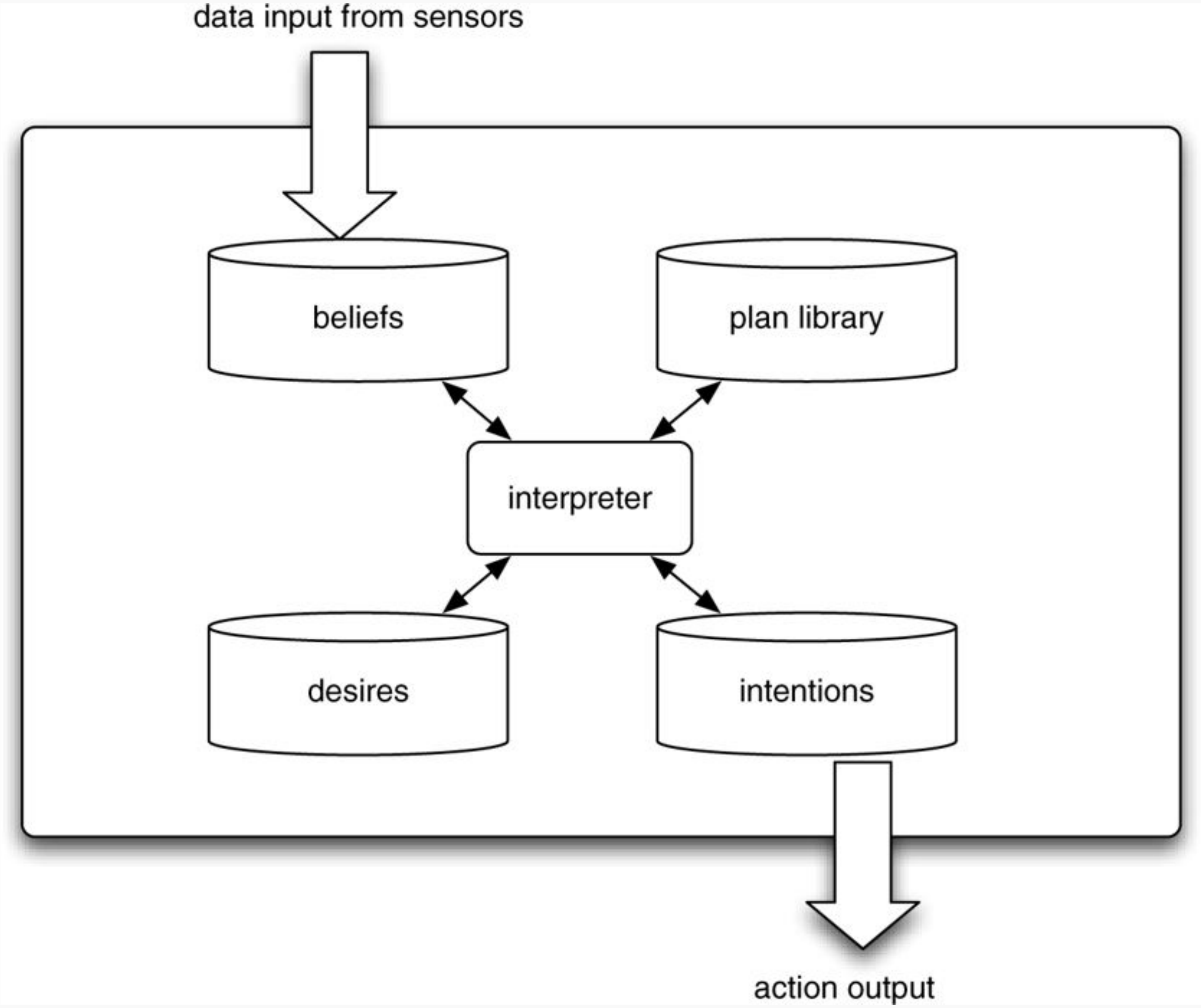
```
defmodule RECOGNIZE CUP
  condition-list: object-observed
  add-list: cup-observed
  delete-list: object-observed
  activation-level: 53
  implementation: <some processes>

defmodule PICK UP CUP
  condition-list: cup-observed hand-empty
  add-list: cup-in-hand
  delete-list: hand-empty
  activation-level: 65
  implementation: <some processes>
```



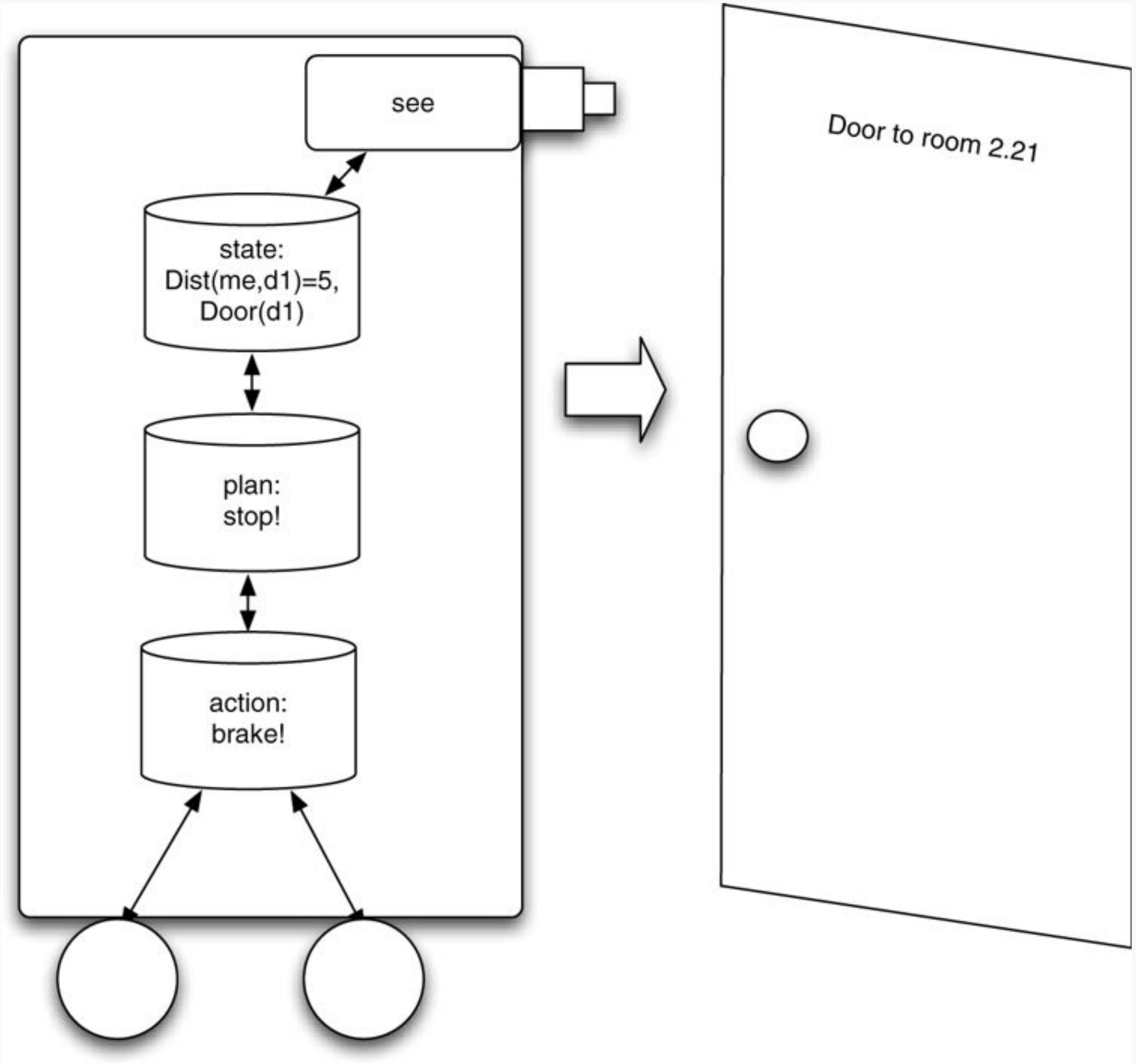
Maes, Pattie. 1991. “The Agent Network Architecture (ANA).” ACM SIGART Bulletin 2 (4): 115–20. `https://doi.org/10.1145/122344.122367`.

Beliefs-Desires-Intentions



Rao, Anand S, and Michael P Georgeff. 1992. “An Abstract Architecture for Rational Agents.” In Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, 439–449. Cambridge, MA, USA.

Logical deduction



Example from: Wooldridge, Michael J. 2009. An Introduction to MultiAgent Systems, 2nd Edition. John Wiley & Sons Ltd.

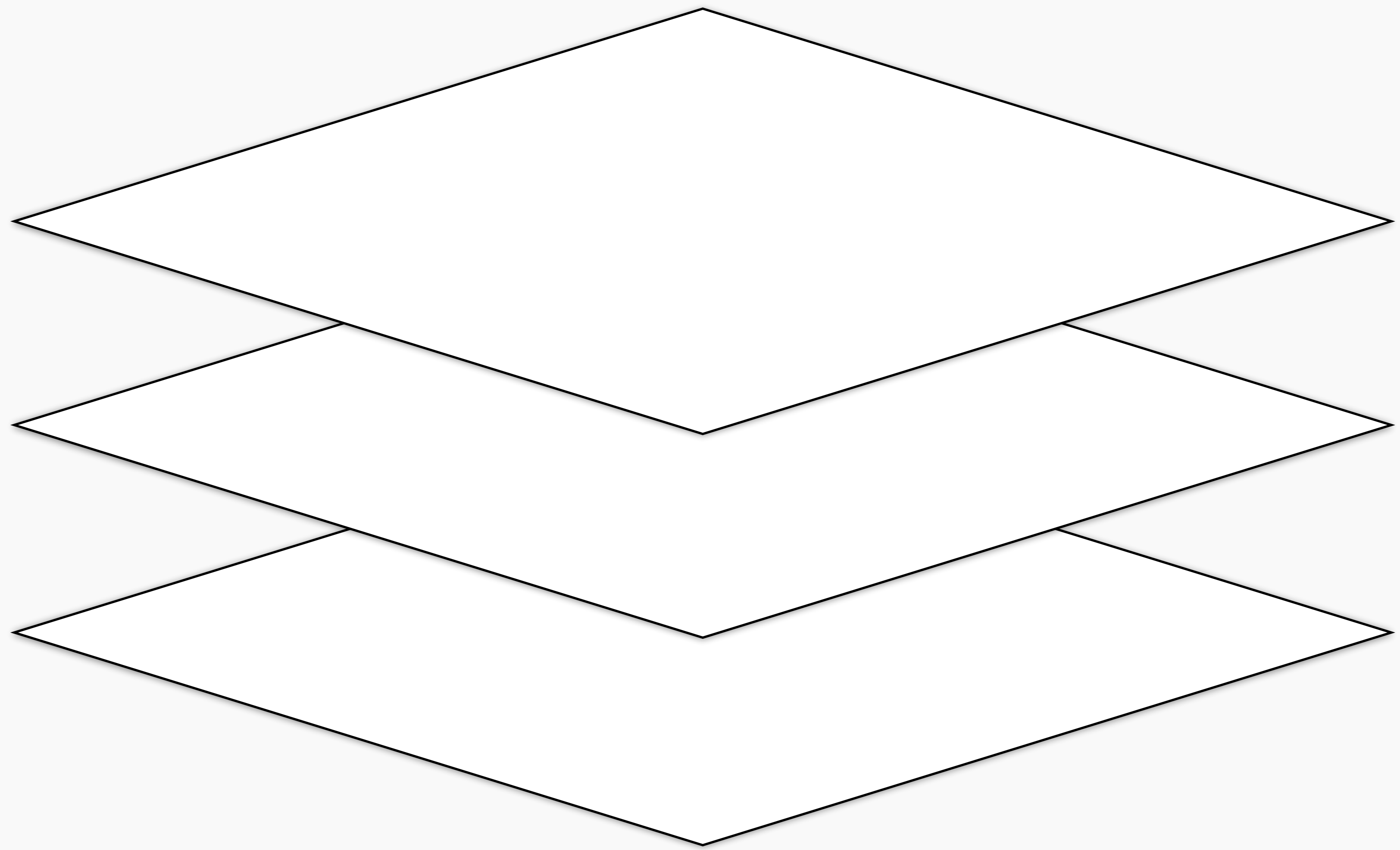
Environment

3-Tier model

MAS Application environment

Execution platform (OS, VM, middleware)

Physical infrastructure (hardware, network)



Weyns, Danny, H Van Dyke Parunak, Fabien Michel, Tom Holvoet, and Jacques Ferber. 2005. “Environments for Multiagent Systems. State-of-the-Art and Research Challenges.” In Environments for Multi-Agent Systems, 1–47. Berlin, Heidelberg: Springer.

Environment

> *The environment is a first-class abstraction that provides the surrounding conditions for agents to exist and that mediates both the interaction among agents and the access to resources*

Weyns, Danny, Andrea Omicini, and James J Odell. 2006. “Environment as a First Class Abstraction in Multiagent Systems.” Autonomous Agents and Multi-Agent Systems 14 (1): 5–30.

> *Agents are* ***situated*** *in an environment that provides the conditions under which an entity (agent or objects) exists. (Odell)*

Odell, James J, H Van Dyke Parunak, Mitch Fleischer, et Sven Brueckner. 2003. « Modeling Agents and Their Environment ». In Agent-Oriented Software Engineering III, 16–31. Springer Berlin Heidelberg.

Environment

Properties:

- Partially vs. totally observable
- Deterministic vs. Stochastic
- Dynamic vs. Static
- Continuous vs. Discrete

Russel, Stuart J, et Peter Norvig. 2009. Artificial Intelligence: A Modern Approach (3rd edition). Prentice Hall.

Environment as a *topology*

$(P, dist)$ is a *quasimetric space*, where:

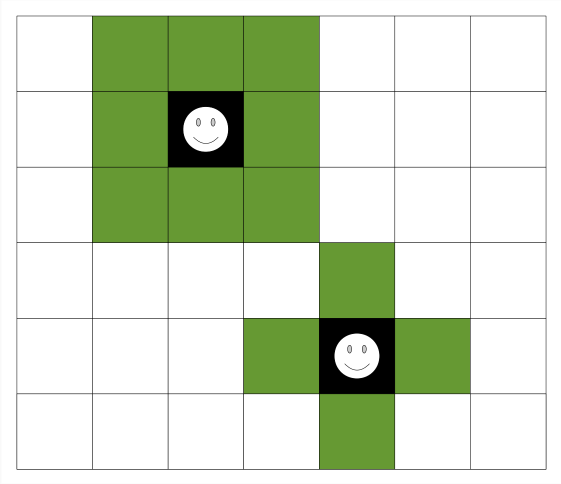
- P is the set of positions in the space
- $dist : P \times P \to \mathbb{R}^+_\infty$ is a metric

$\forall x, y, z \in P :$

$dist(x, x) = 0$	(reflexivity)
$dist(x, y) = 0 \iff x = y$	(identity of indiscernibles)
$dist(x, y) \geq 0$	(positivity)
$dist(x, z) \leq dist(x, y) + dist(y, z)$	(triangular inequality)
~~$dist(x, y) = dist(y, x)$~~	~~(symmetry)~~

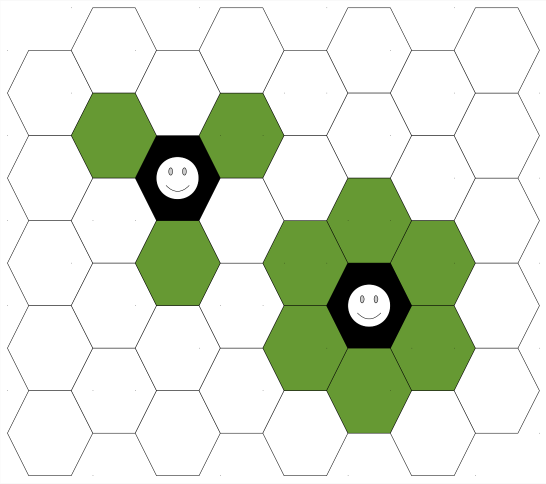
Mathieu, Philippe, Sébastien Picault, and Yann Secq. 2015. "Design Patterns for Environments in Multi-Agent Simulations." In PRIMA 2015: Principles and Practice of Multi-Agent Systems, 9387:678–86. Cham: Springer International Publishing. `https://doi.org/10.1007/978-3-319-25524-8_51`.

$P = \mathbb{Z}^2$

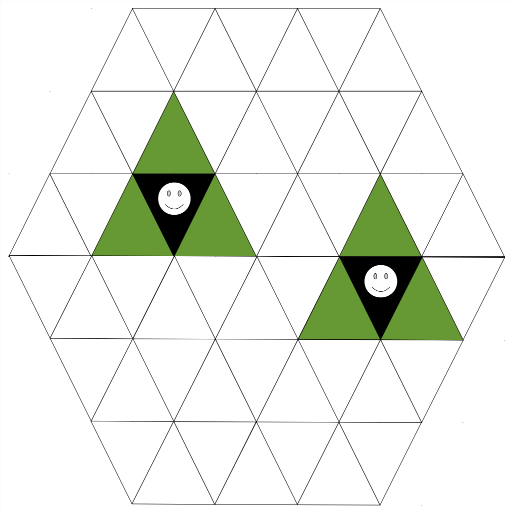


Chebychev distance (Moore)

Manhattan distance (von Neumann)

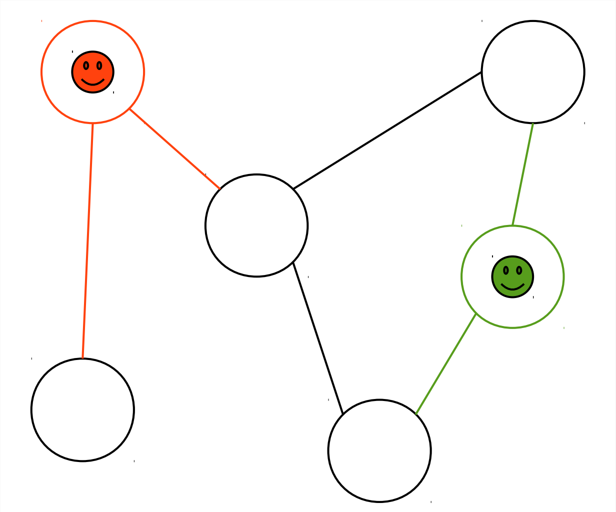


Hexagonal neighborhood



Triangular neighborhood

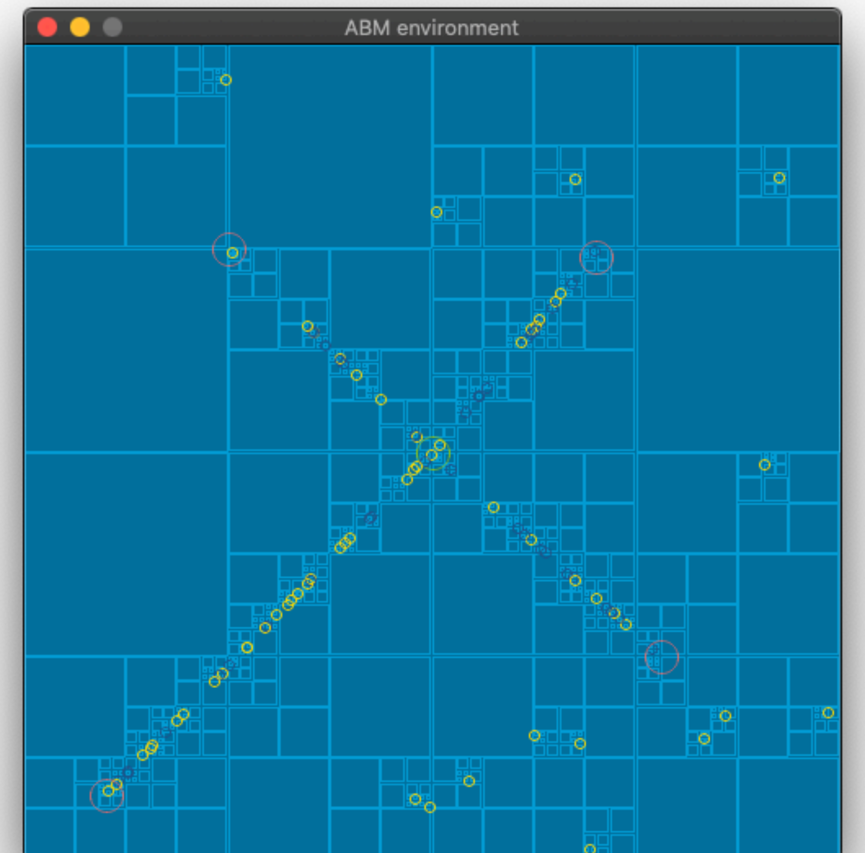
$P = Vertices$



Geodesic distance

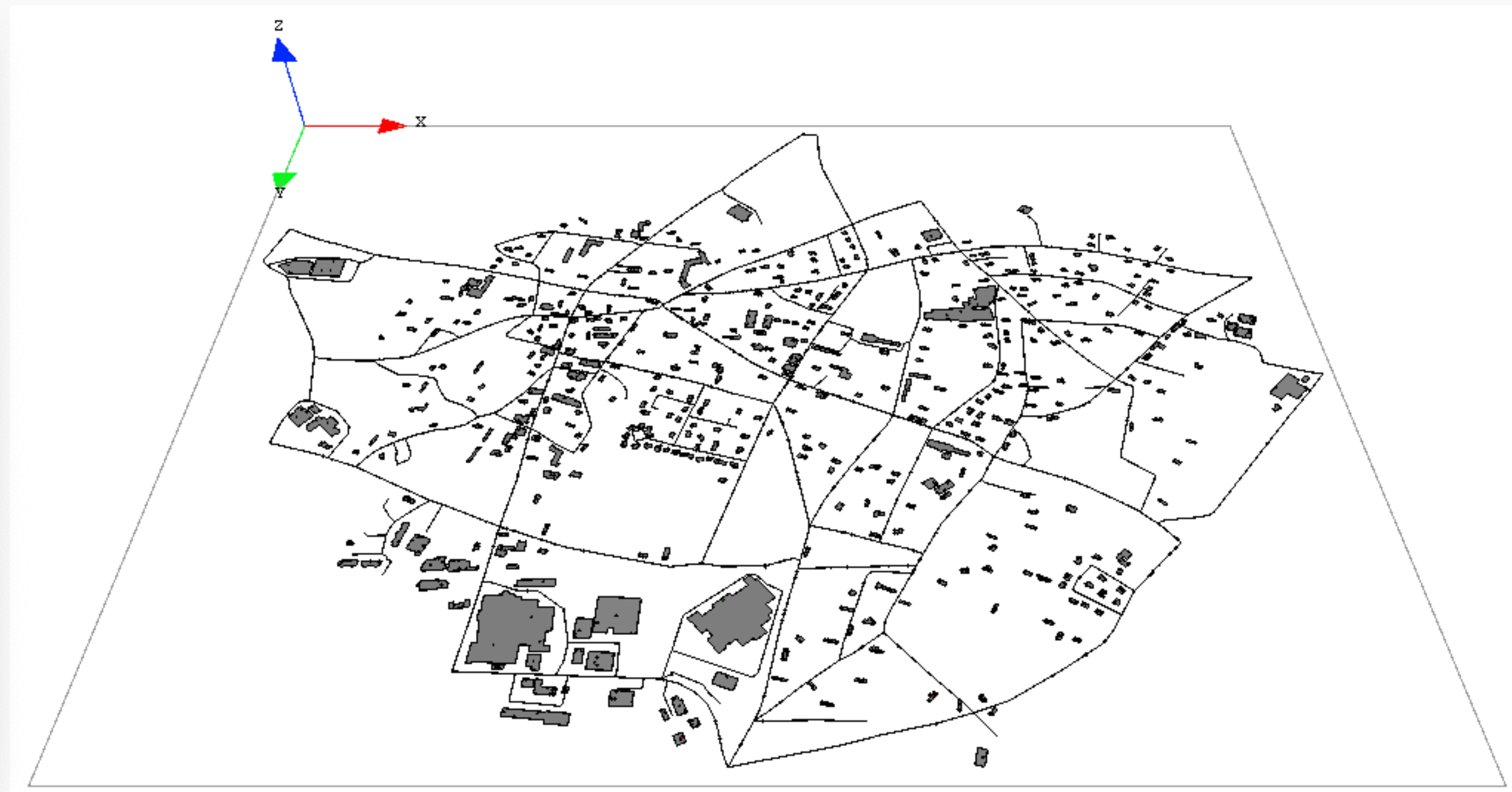
(shortest path)

Environment (continuous)



$P = \mathbb{R}^2$

Euclidean distance



$P = \mathbb{R}^3$

Euclidean distance

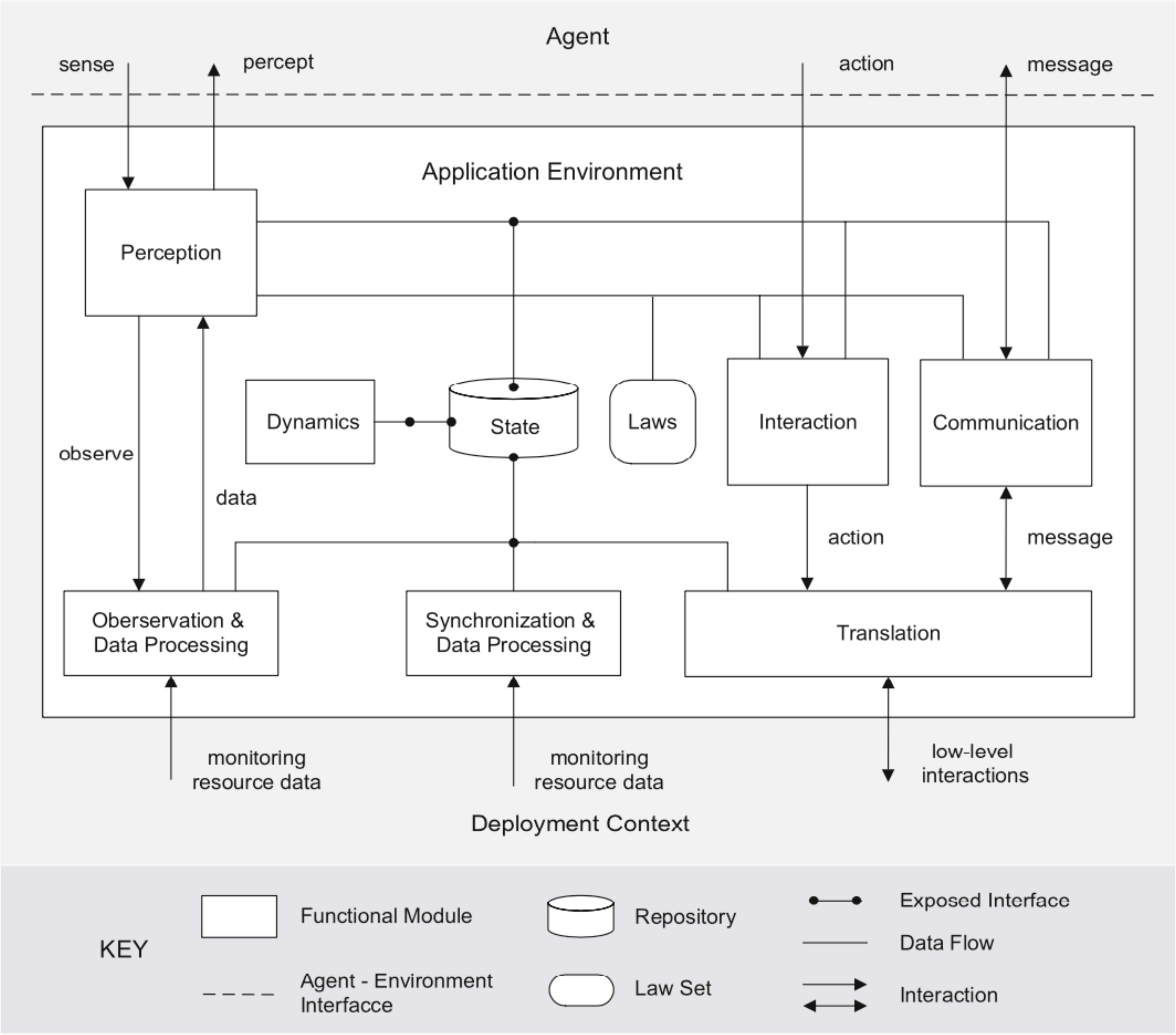
Environment

A *structuring* entity:

- *physical* structuring
- *communication* structuring
- *social* structuring

Weyns, Danny, Andrea Omicini, and James J Odell. 2006. "Environment as a First Class Abstraction in Multiagent Systems." Autonomous Agents and Multi-Agent Systems 14 (1): 5–30.

Environment



Weyns, Danny, Andrea Omicini, and James J Odell. 2006. “Environment as a First Class Abstraction in Multiagent Systems.” Autonomous Agents and Multi-Agent Systems 14 (1): 5–30.

Interaction

Interaction allows agents to *exchange information*, so they can *cooperate*, *negotiate*, or *solve* a conflict rather than just *compete*.

Enabler of **synergy** and **emergence.**

Two types of interaction generally distinguished:

- direct
- indirect

Indifference, Cooperation, Antagonism

Goals	Resources	Competence	Situation	
Complete	Ok	Ok	*Independence*	
	Ok	Insufficient	*Cooperation*	Simple collaboration
	Scarce	Ok		Congestion
	Scarce	Insufficient		Coordinated collaboration
Incomplete	Ok	Ok	*Antagonism*	Individual competition
	Ok	Insufficient		Collective competition
	Scarce	Ok		Individual conflicts for resources
	Scarce	Insufficient		Collective conflicts for resources

Ferber, Jacques. 1999. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. 1st éd. Addison-Wesley Longman Publishing Co., Inc.

Indirect Interaction

Agents interacts *through* the environment and are not necessarily aware of other agents.

Possible architectures:

- Blackboard systems
- Tuple spaces
- Stigmergy

Blackboard systems

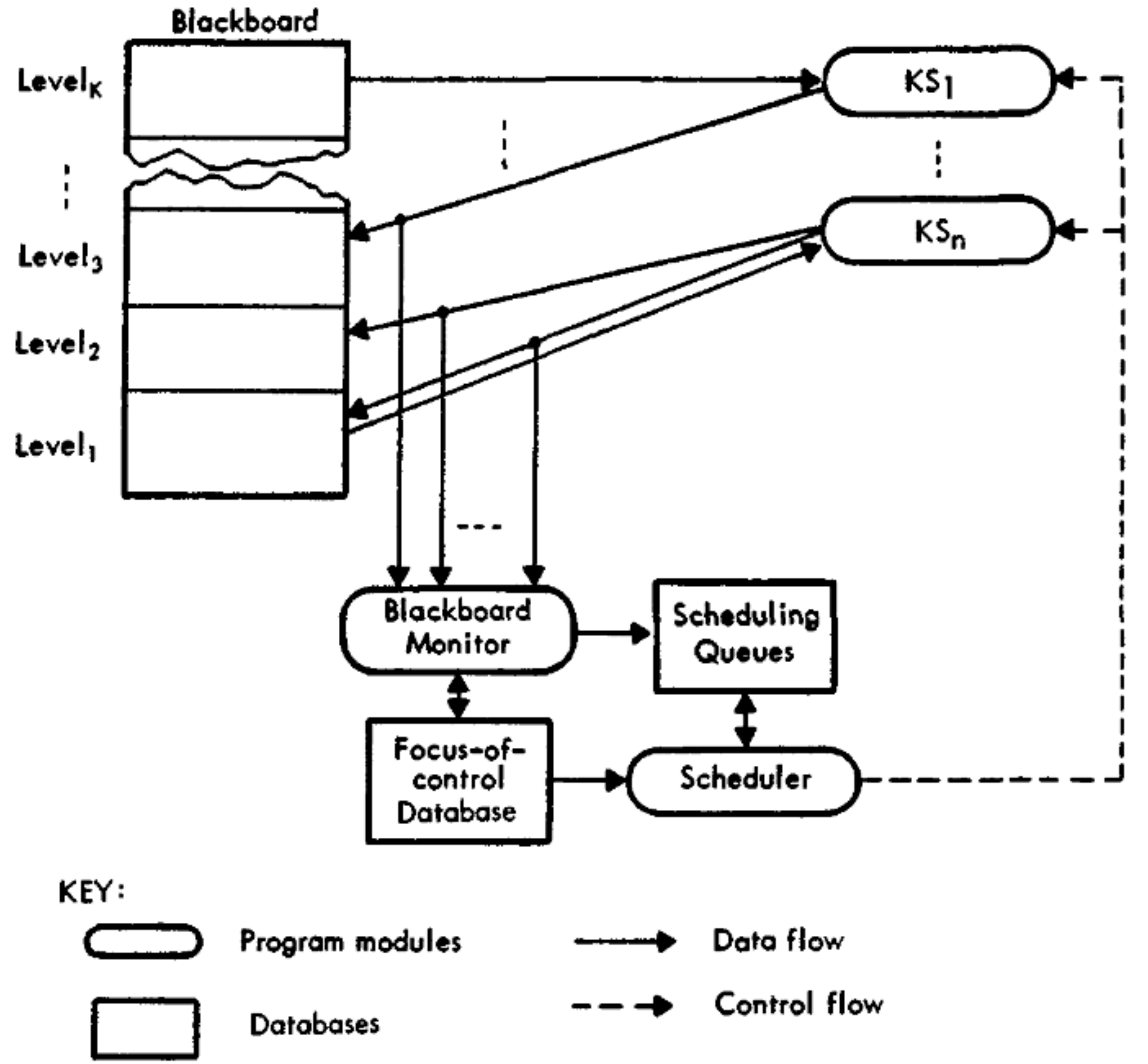


FIGURE 4. Schematic of the Hearsay-II architecture.

Erman, Lee D, Frederick Hayes-Roth, Victor R Lesser, and D Raj Reddy. 1980. "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty." ACM Computing Surveys 12 (2): 213–253.

Tuple spaces

Introduced by Linda :

- Coordination and communication languages
- Independent processes shares a tuple space (multiset)
- Tuples are stored and retrieved via 3 operations
  - in (atomic consume)
  - rd (read)
  - out (write)

Gelernter, David, and Nicholas Carriero. 1992. “Coordination Languages and Their Significance.” Communications of the ACM 35 (2): 96.

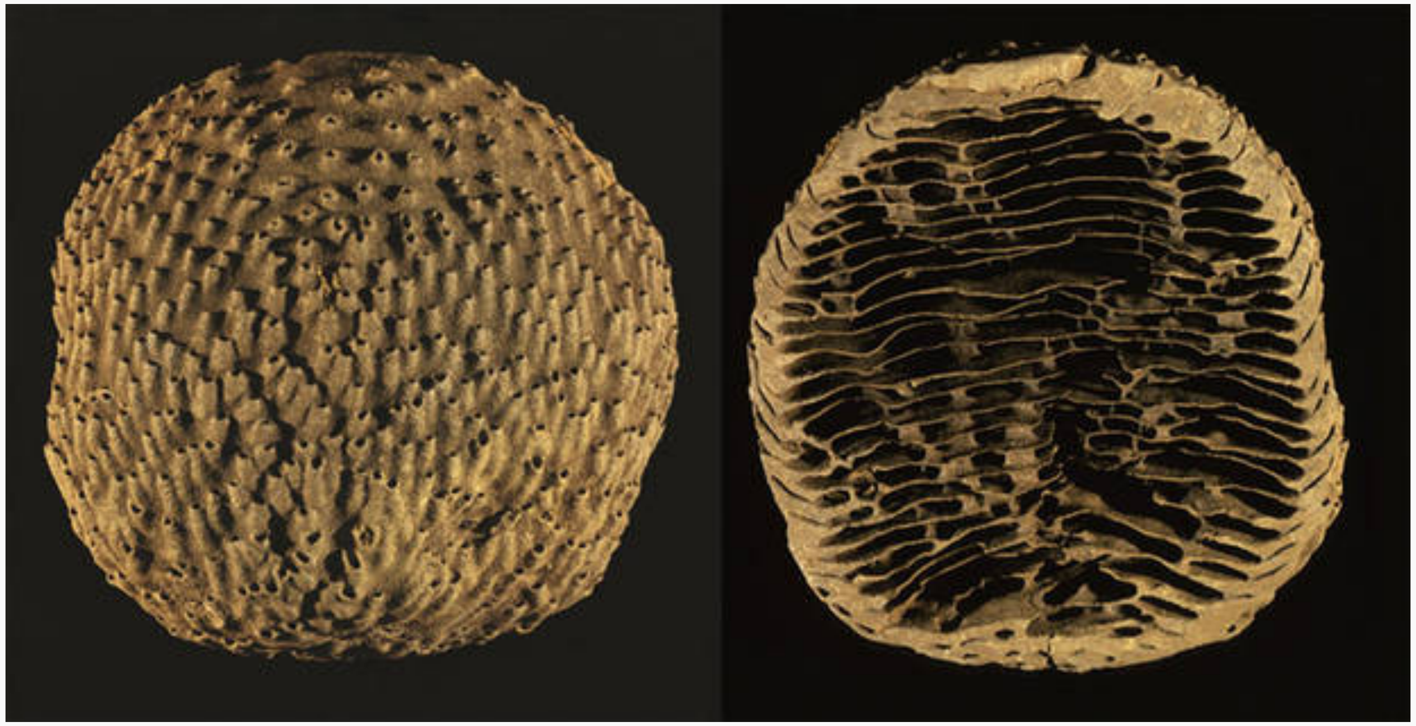
LIME (Linda in a Mobile Environment) :

- 1 agent, 1 tuple space
- Tuple spaces merged when agents are on the same host

Murphy, A., Picco, G.P., Roman, G.C.: LIME: a Middleware for Physical and Logical Mobility. 21th International Conference on Distributed Computing Systems (2001)

Stigmergy, coined by P. Grassé

A *spontaneous* phenomenon *emerges* from the set of individual actions leaving traces in the environment



In practice, depends on :

- Gradient fields (attractive/repulsive)
- Resources (objects that agents can produce/manipulate)

Grassé, PIerre-P. 1959. “La Reconstruction Du Nid et Les Coordinations Interindividuelles Chez Bellicositermes Natalensis et Cubitermes Sp. La Théorie de La Stigmergie: Essai d’interprétation Du Comportement Des Termites Constructeurs.” Insectes Sociaux 6 (1): 41–80.

Direct Interaction

Agents communicate through message passing using dedicated channels.

Requires a shared *communication language:*

- FIPA-ACL
- KQML

Influenced by the *speech act* theory (John R. Searle, 1960s):

- *Fact* vs. *performative* statements
- Explicitly model the *intention* as well as the content of a message

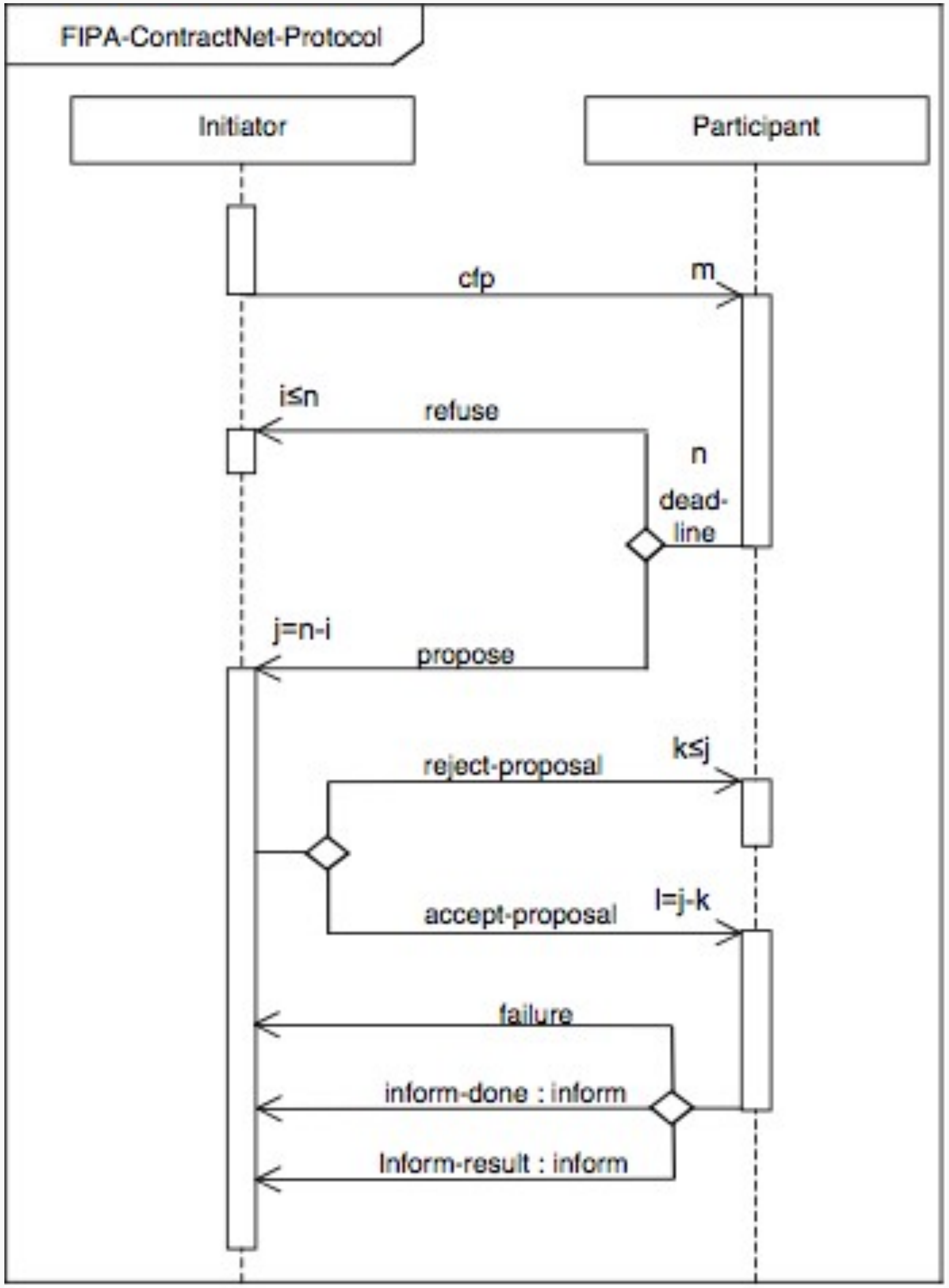
FIPA-ACL

Parameter	Category of Parameters
`performative`	Type of communicative acts
`sender`	Participant in communication
`receiver`	Participant in communication
`reply-to`	Participant in communication
`content`	Content of message
`language`	Description of Content
`encoding`	Description of Content
`ontology`	Description of Content
`protocol`	Control of conversation
`conversation-id`	Control of conversation
`reply-with`	Control of conversation
`in-reply-to`	Control of conversation
`reply-by`	Control of conversation

Table 1: FIPA ACL Message Parameters

Foundation for Intelligent Physical Agents. 2002. FIPA ACL Message Structure Specification.

FIPA-ContractNet



Foundation for Intelligent Physical Agents. 2002. FIPA Contract Net Interaction Protocol Specification.

Organisation

Organisation

Organisation is about forming virtual societies of agents in terms of:

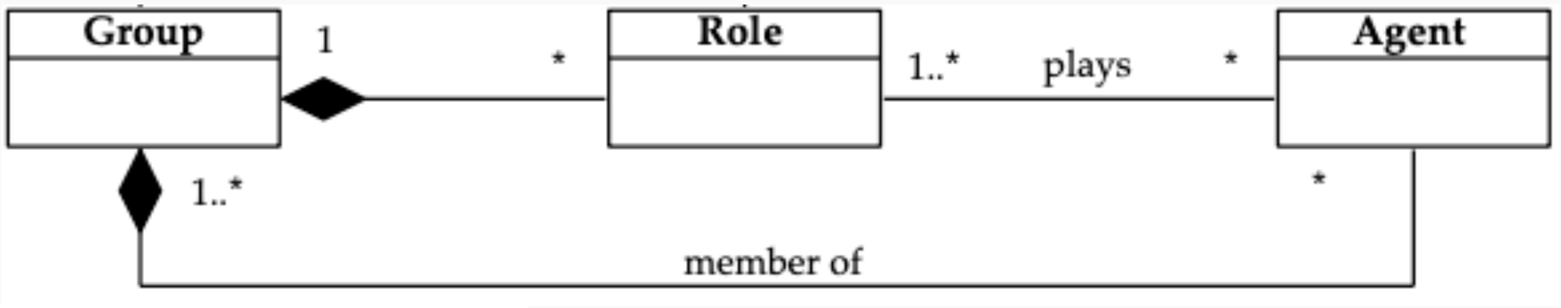
- Structure (groups, roles)
- Behaviour (norms, sanctions)
- Collective knowledge (institutions, culture)

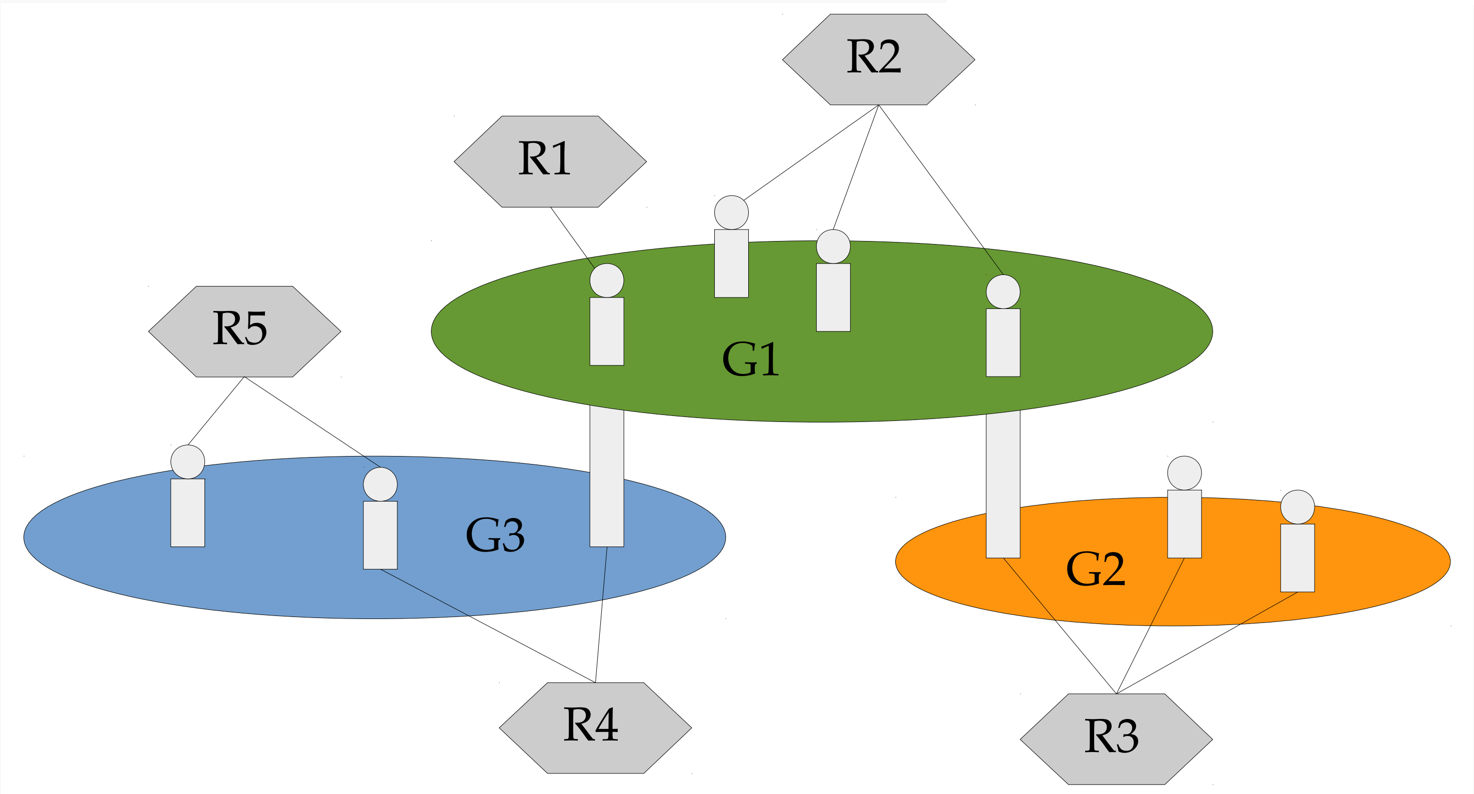
Structural organisation

Establish the links that unite (or oppose) agents (OCMAS)

- Helps managing complexity (who to interacts with)
- Hierarchy between agents, roles, groups
- Part of the environment responsibilities

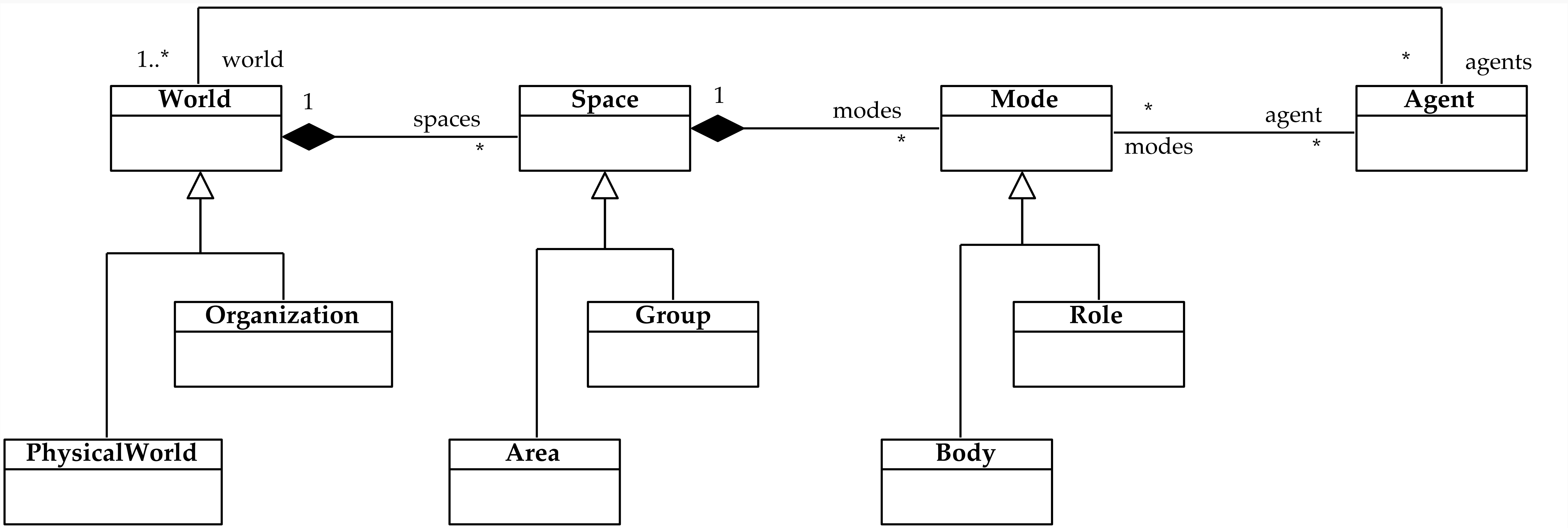
Example: AGR metamodel





Ferber, Jacques, Fabien Michel, and Olivier Gutknecht. 2003. "Agent/Group/Roles: Simulating with Organizations." In ABS'03: Agent Based Simulation. Montpellier (France).

AGRE (Agent-Group-Role-Environment)



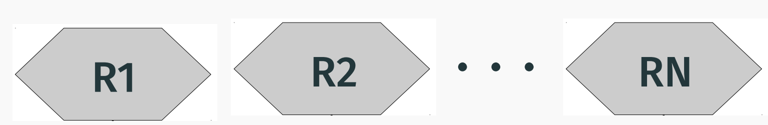
Ferber, Jacques, Fabien Michel, and José-Antonio Báez-Barranco. 2005. “AGRE: Integrating Environments with Organizations.” In Environments for Multi-Agent Systems, 48–56. Berlin, Heidelberg: Springer.

$(P, dist)$ is a ~~quasimetric~~ *hemimetric space*, where:

- P is the set of positions in the space
- $dist : P \times P \to \mathbb{R}^{+}_{\infty}$ is a metric

$\forall x, y, z \in P :$

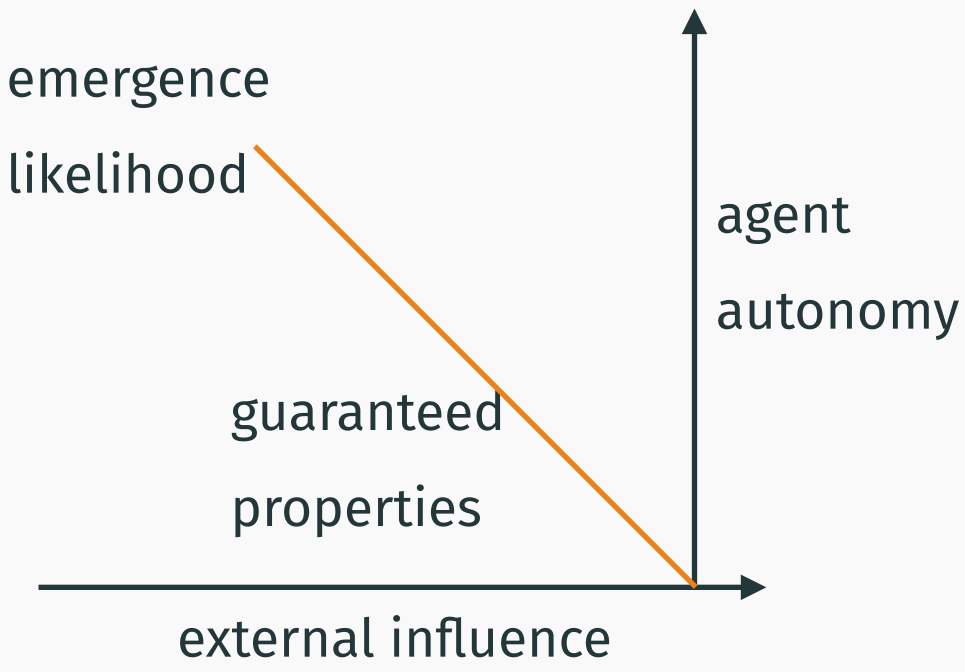
$dist(x, x) = 0$	(reflexivity)
~~$dist(x, y) = 0 \iff x = y$~~	~~(identity of indiscernibles)~~
$dist(x, y) \geq 0$	(positivity)
$dist(x, z) \leq dist(x, y) + dist(y, z)$	(triangular inequality)

$P = \{$ 

$\}$

Controlling agents behaviour

- Influence agents
- Contradicts autonomy property



Borrow concepts from social sciences

- Norms
- Social commitment
- Sanctions

Normative MAS

Norm = Principle of good deed

- Guides or regulates agent behavior
- Norms shared by a group
- Members can judge conformance or deviance
- Norms may evolve

From agent perspective

- Can choose conformance or deviance
- Can anticipate behavior of other agents
- Still fully autonomous

Boella, Guido, Leendert van der Torre, and Harko Verhagen. 2006. “Introduction to Normative Multiagent Systems.” Computational & Mathematical Organization Theory 12 (2–3): 71–79.

Social commitment is about modelling expectation.

A commitment is made by a *debtor* to a *creditor*

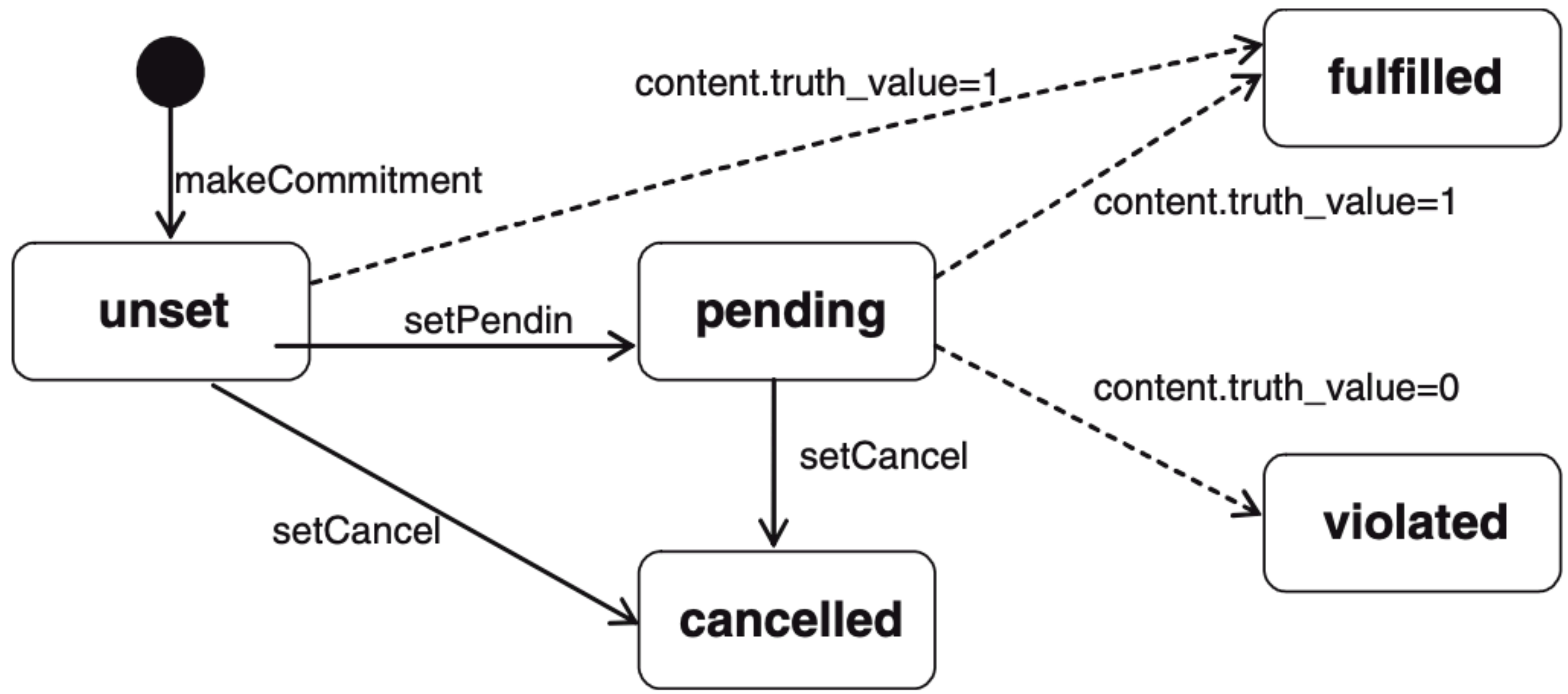


Fig. 2 The life-cycle of commitments

Fornara, Nicoletta, Francesco Vigan, and Marco Colombetti. 2006. "Agent Communication and Artificial Institutions." Autonomous Agents and Multi-Agent Systems 14 (2): 121–142.

Sanction (or reward)

Types:

- automatic (carried by action)
- material (e.g. violence/healing)
- social (e.g. reputation)
- psychological (emotions, e.g guilt)

Styles:

- implicit (self-inflicted)
- explicit (public)

Application policies:

- deterrence (severe immediate sanctions, reduces flexibility)
- retribution (revenge)
- invalidation (isolation)

Pasquier, Philippe, Roberto A Flores, and Brahim Chaib-draa. 2005. “Modelling Flexible Social Commitments and Their Enforcement.” In Engineering Societies in the Agents World V, 139–151. Berlin, Heidelberg: Springer.

Collective knowledge organisation

Norms as a regulation system

- Makes sense for a community
- What about distinct communities?

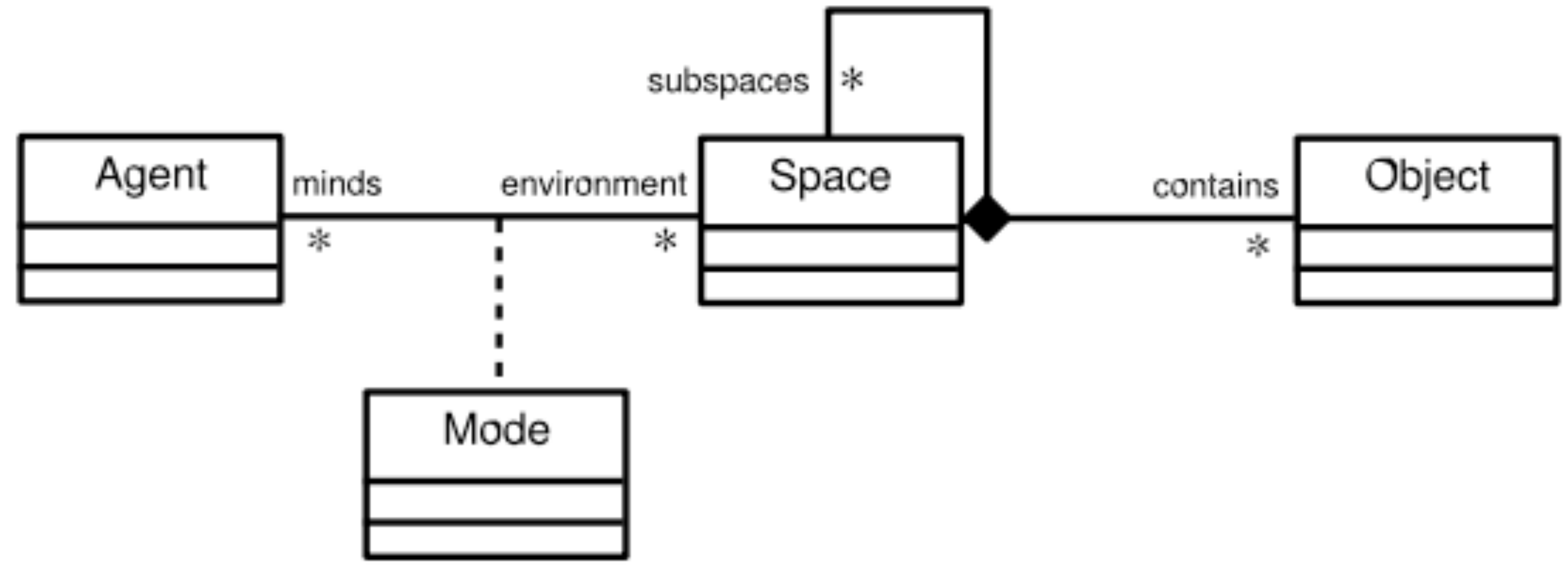


Fig. 1. Simplified UML representation of AGREEN

Institution

- Rule-based system
- Regulate interactions
- Institutional facts
- Assigns status to entities/agents
- Capability
- Relations between social and physical world
    - *count as* operator (X counts as Y in context C)

Báez-Barranco, José-Antonio, Tiberiu Stratulat, and Jacques Ferber. 2007. “A Unified Model for Physical and Social Environments.” In Environments for Multi-Agent Systems III, 41–50. Berlin, Heidelberg: Springer.

Culture

- Norms and ontologies relevant for a community
- MASQ (Multi-Agent Systems based on Quadrant) from Ken Wilber theory

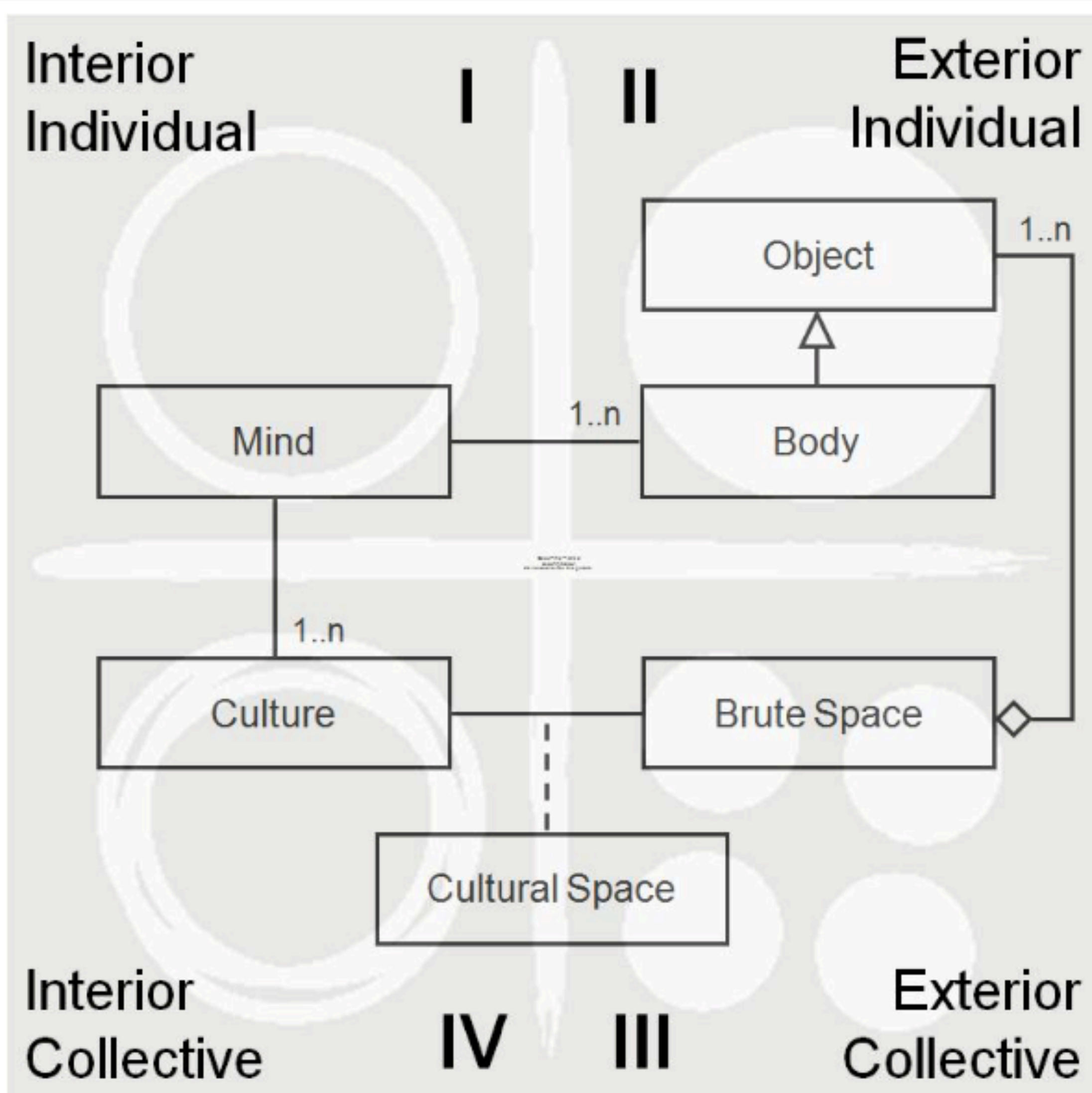
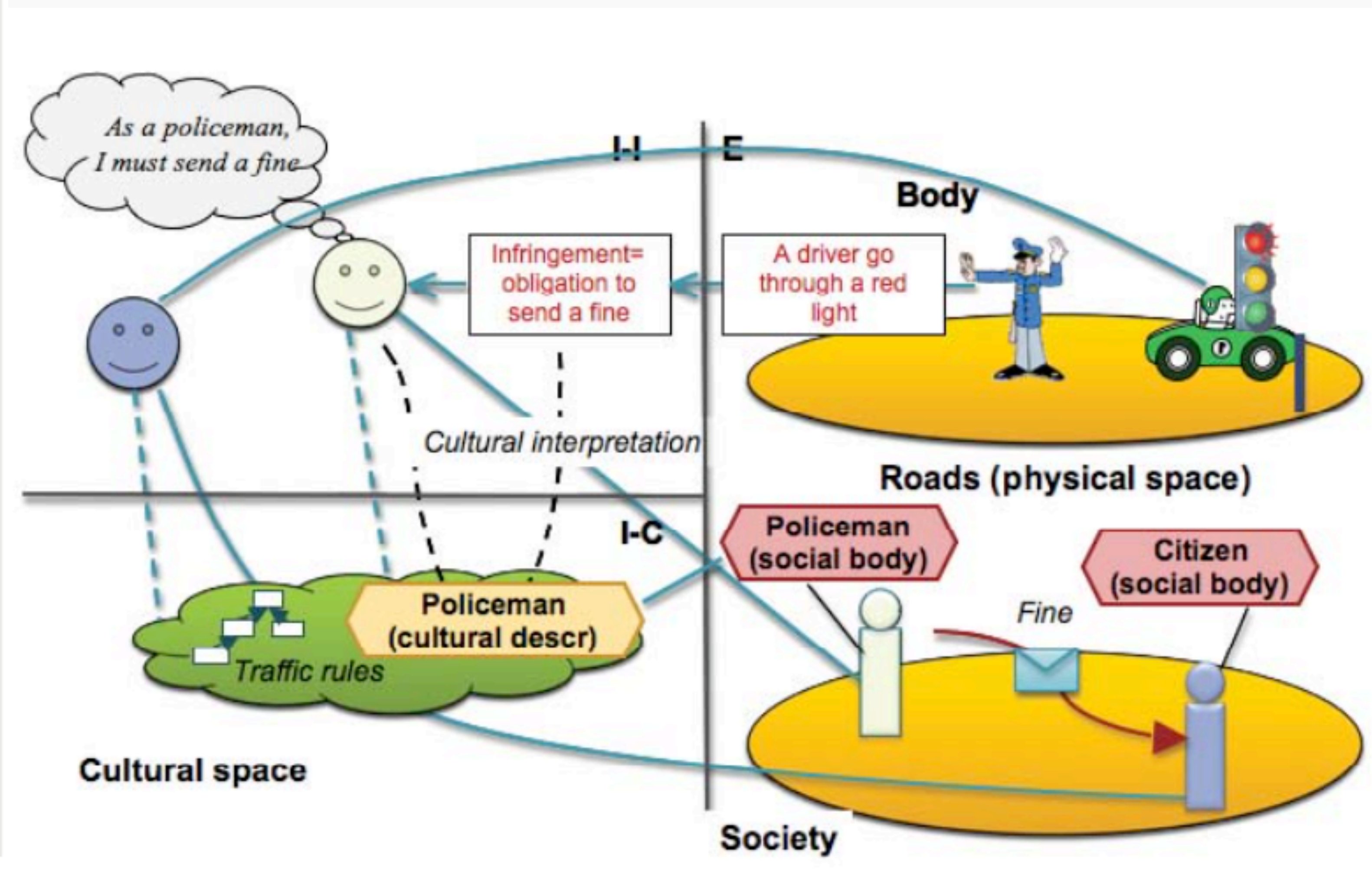


Fig. 1. MASQ meta-model



Dinu, Razvan, Tiberiu Stratulat, and Jacques Ferber. 2012. "A Formal Model of Agent Interaction Based on MASQ." In AMPLE'2012: 2nd International Workshop on Agent-Based Modeling for PoLicy Engineering. Montpellier, France.

Scheduling

Abstract architecture

Let

$E = \{e, e', \ldots\}$ a finite set of discrete instantaneous environment states, and

$Ac = \{a, a', \ldots\}$ the set of possible actions available to agents.

A run, r, of an agent in an environment is a sequence of *interleaved* environment states and actions:

$$r : e_0 \xrightarrow{a_0} e_1 \xrightarrow{a_1} e_2 \xrightarrow{a_2} e_3 \xrightarrow{a_3} \ldots \cdots \xrightarrow{a_{n-1}} e_n.$$

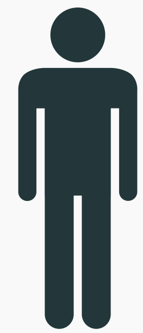
- **In-place** vs. out-place

Wooldridge, Michael J. 2009. An Introduction to MultiAgent Systems, 2nd Edition. John Wiley & Sons Ltd.

Abstract architecture

$$Mem_a : P_a \times S_a \to S_a$$

$$Percept_a : \Sigma \to P_a$$



a

$$Decision_a : P_a \times S_a \to \Sigma$$

$$Behavior_a : \Sigma \to \Sigma$$

$$\sigma \mapsto Decision_a(p_a, Mem_a(p_a, s_a))$$

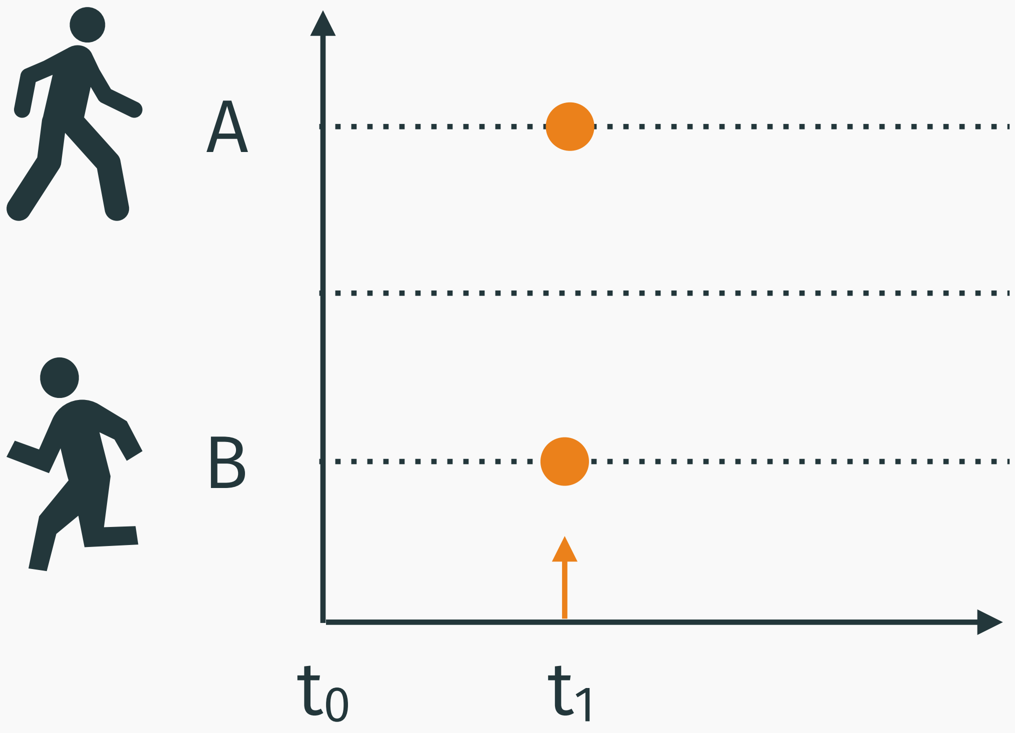
with

$$p_a = Percept_a(\sigma)$$

Genesereth, Michael R, and Nils J Nilsson. 1987. Logical Foundations of Artificial Intelligence. Morgan Kaufman.

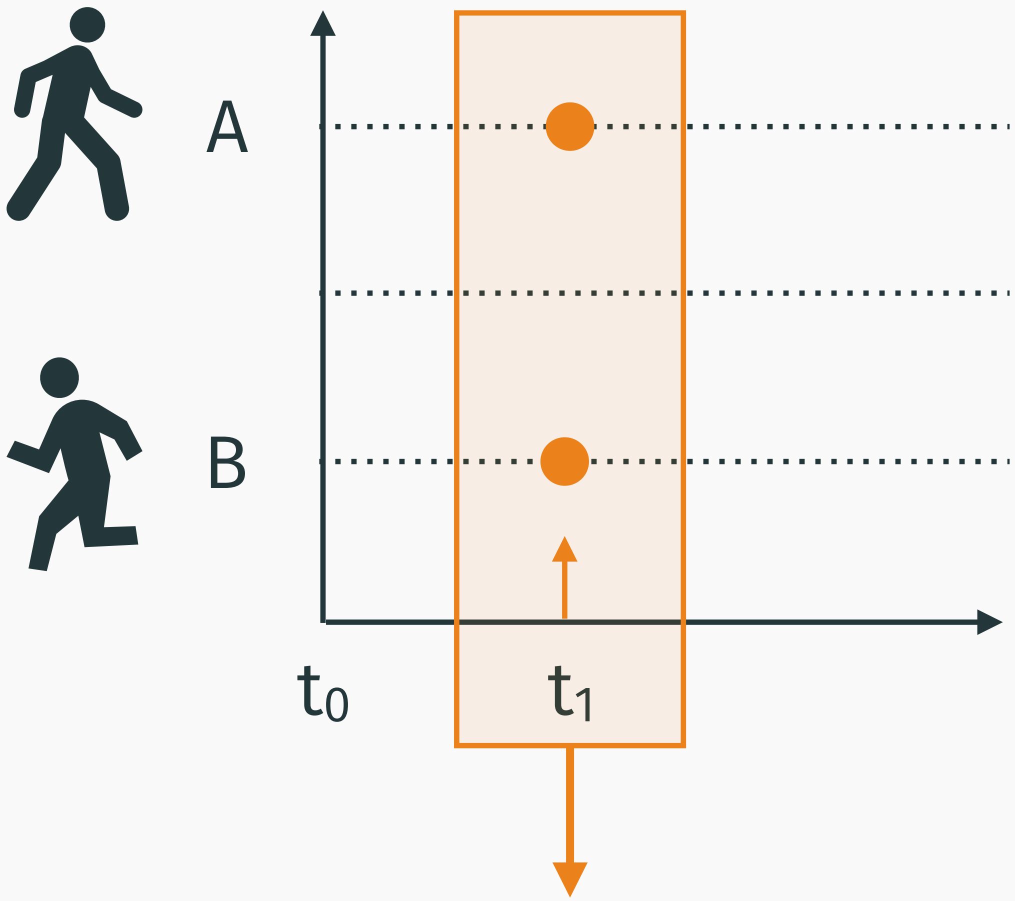
Operational semantics

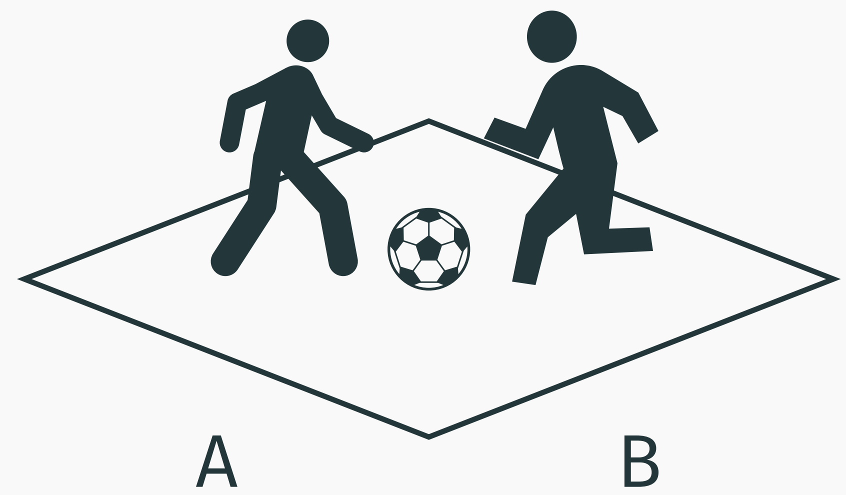
```
def simulate(abm: ABM) {
  time = 0
  env = abm.env
  env.state = env.initial_state
  for (ag in abm.agents) {
    ag.state = ag.initial_state
  }
  while (not termination_condition()) {
    for (ag in abm.agents) {
      percept = ag.percept(env.state)
      ag.state = ag.mem(percept, ag.state)
      env.state = ag.decision(percept, ag.state)
    }
    time += 1
  }
}
```


A
B
t0
t1


A
B

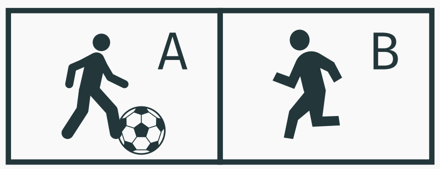
Scheduling

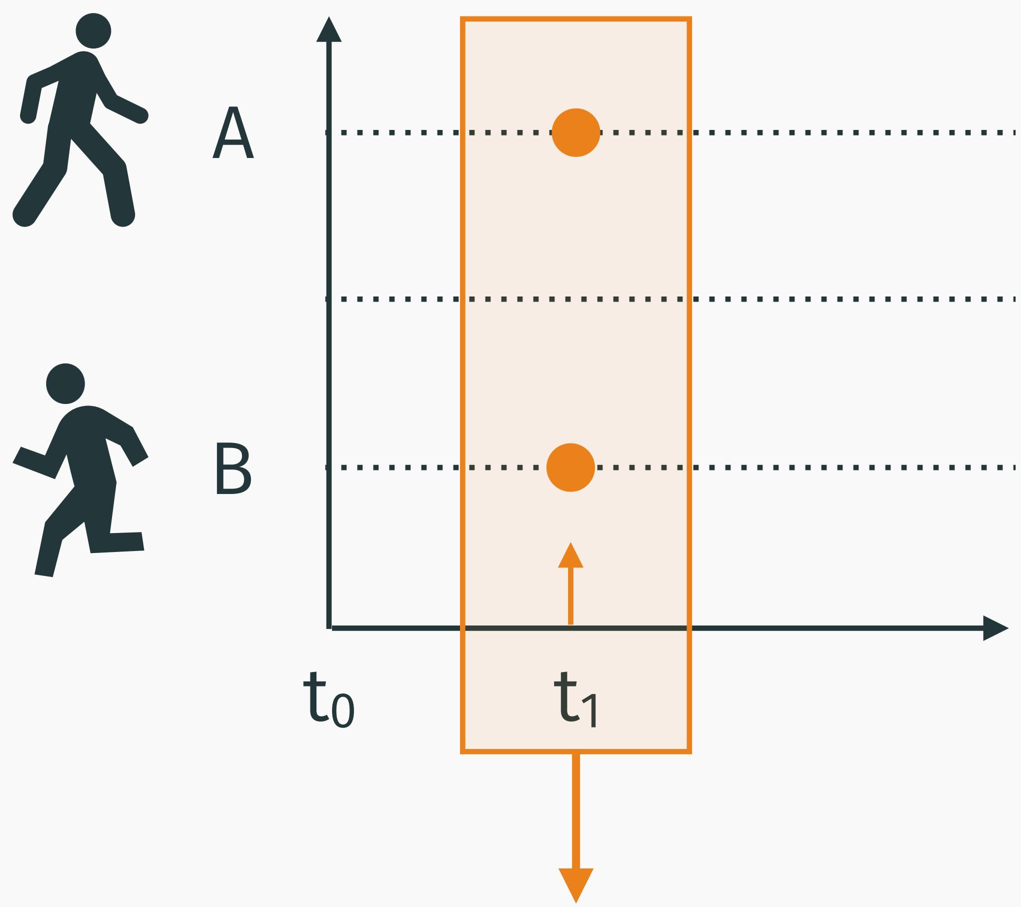




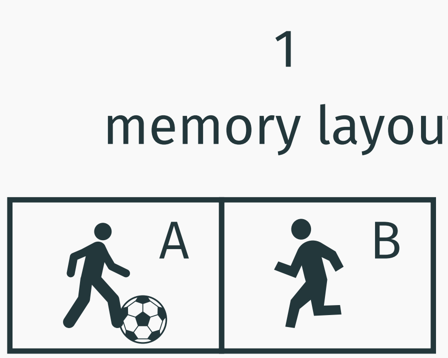
Sequential application

1
memory layout





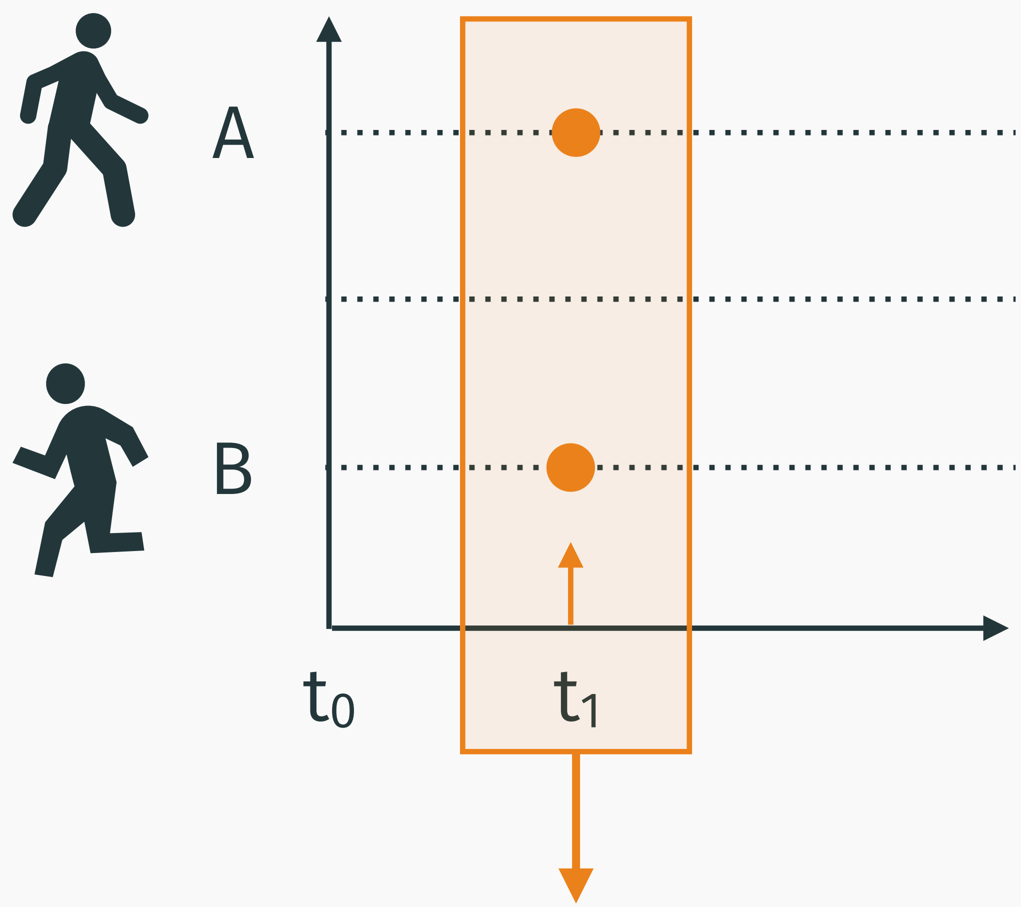
Sequential application



Operational semantics (random)

```
def simulate(abm: ABM) {
  time = 0
  prng_seed = abm.seed
  env = abm.env
  env.state = env.initial_state
  for (ag in abm.agents) {
    ag.state = ag.initial_state
  }
  while (not termination_condition()) {
    for (ag in shuffle(abm.agents)) {
      percept = ag.percept(env.state)
      ag.state = ag.mem(percept, ag.state)
      env.state = ag.decision(percept, ag.state)
    }
    time += 1
  }
}
```

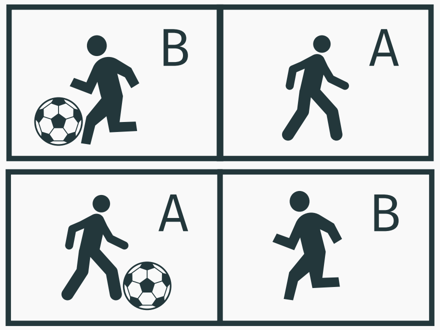
Scheduling



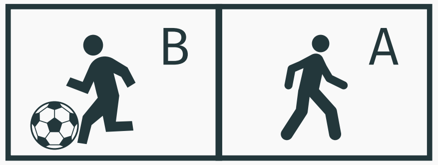
Sequential application

1
memory layout

2
random



3
sort (eg: by dribbling skill)

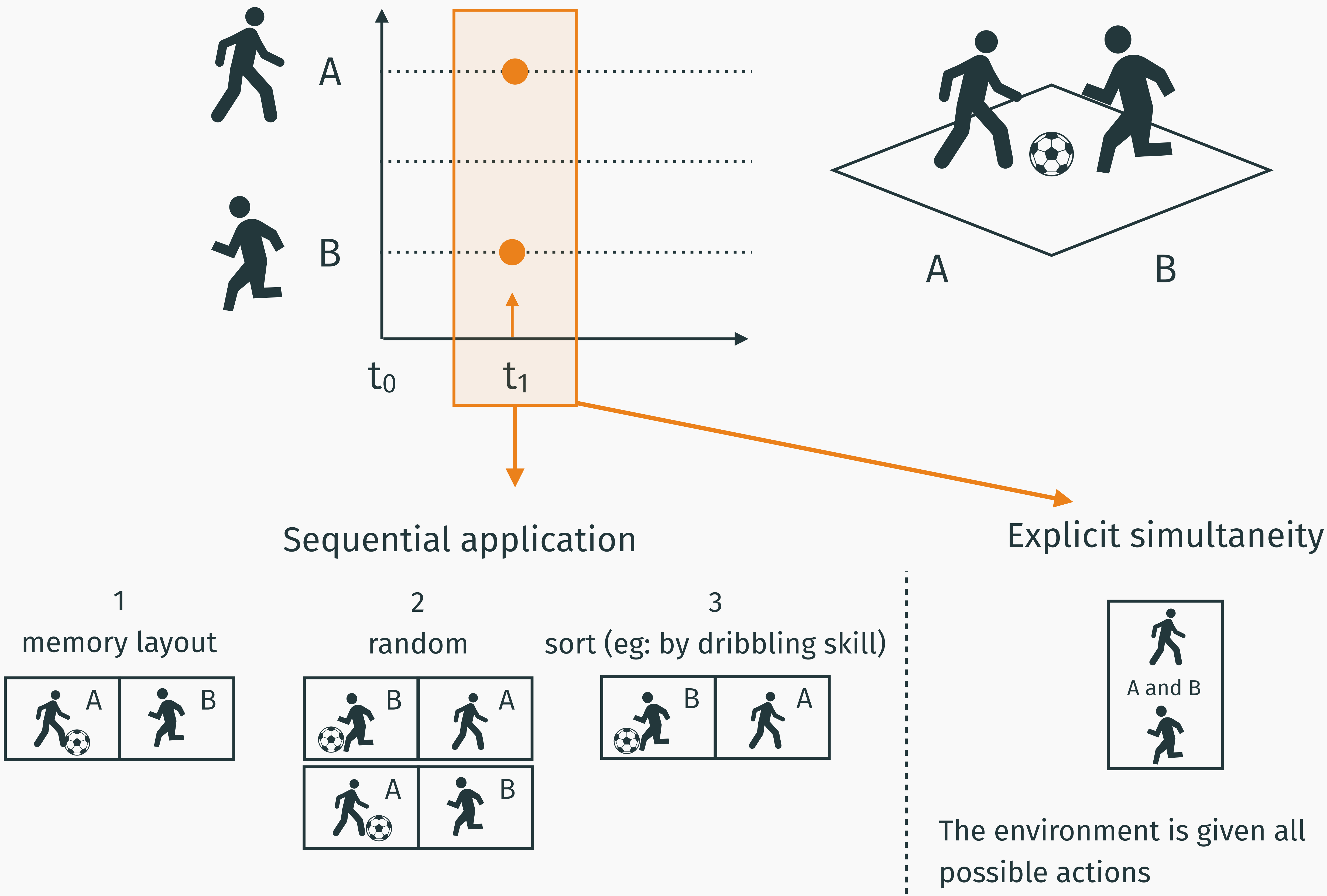


Operational semantics (explicit order)

```
def simulate(abm: ABM) {
  time = 0
  env = abm.env
  env.state = env.initial_state
  for (ag in abm.agents) {
    ag.state = ag.initial_state
  }
  while (not termination_condition()) {
    for (ag in sort(abm.agent_comparator, abm.agents)) {
      percept = ag.percept(env.state)
      ag.state = ag.mem(percept, ag.state)
      env.state = ag.decision(percept, ag.state)
    }
    time += 1
  }
}
```

```
abm.agent_comparator = lambda(ag) { ag.dribbling_skill }
```

Scheduling


A
B
t_0
t_1
A
B
Sequential application
Explicit simultaneity
1
memory layout
A
B
2
random
B
A
A
B
3
sort (eg: by dribbling skill)
B
A
A and B
The environment is given all possible actions

The IRM4S model

System: $\Delta = \langle \Sigma, \Gamma \rangle$, where

- Σ is the set of environment states
- Γ is the set of influences

$$Behaviour_a : \Sigma \times \Gamma \to \Gamma$$

$$Natural_e : \Sigma \times \Gamma \to \Gamma$$

$$\begin{aligned} Evolution : \quad \Delta \quad &\to \Delta \\ (\sigma, \gamma) \quad &\mapsto Reaction(\sigma, Influence(\sigma, \gamma)) \end{aligned}$$

$$\begin{aligned} Influence &: \Sigma \times \Gamma \to \Gamma \\ &(\sigma, \gamma) \mapsto \bigcup_{a \in Ag} Behaviour_a(\sigma, \gamma) \cup Natural_e(\sigma, \gamma) \end{aligned}$$

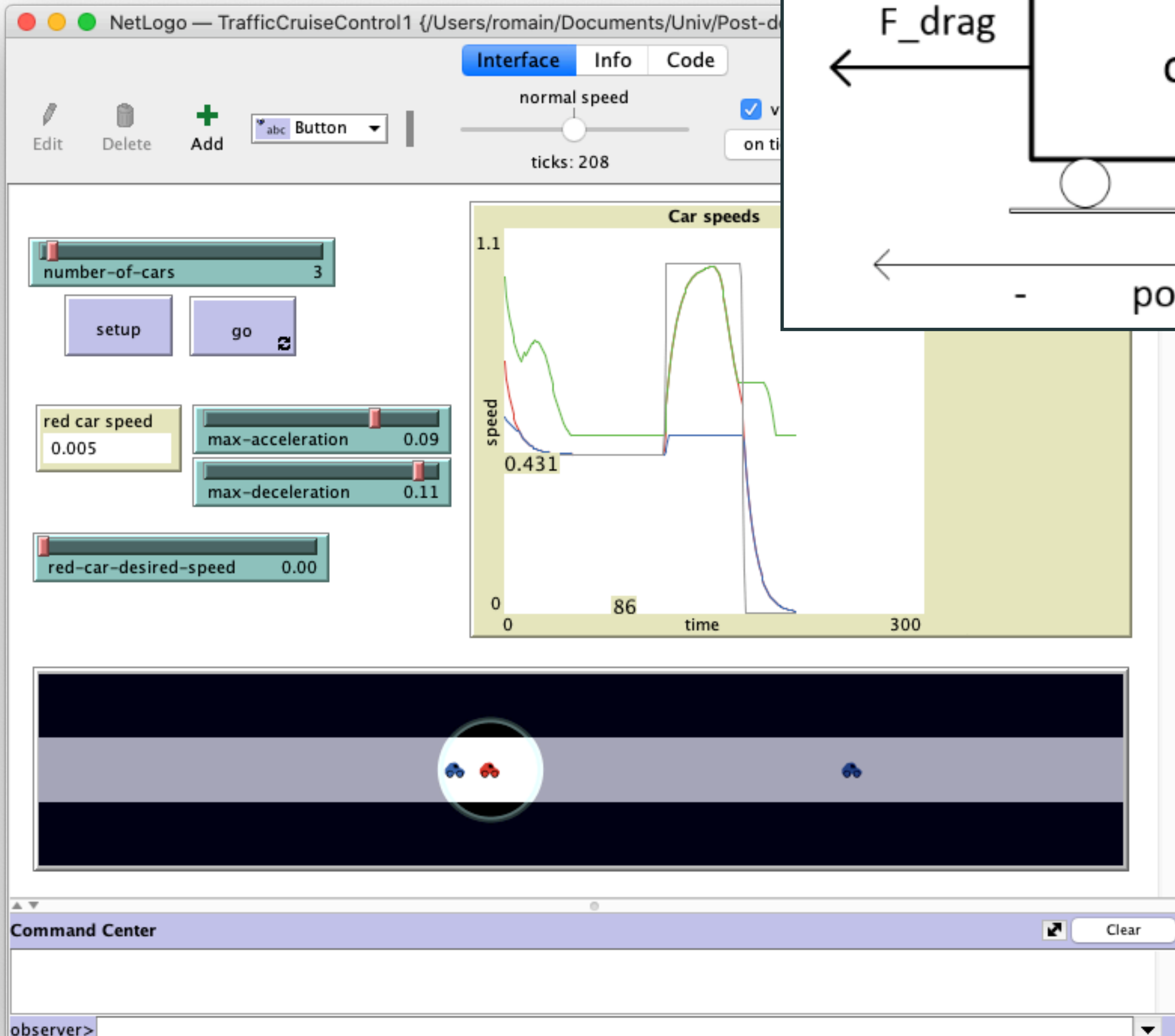
$$Reaction : \Sigma \times \Gamma \to \Sigma \times \Gamma$$

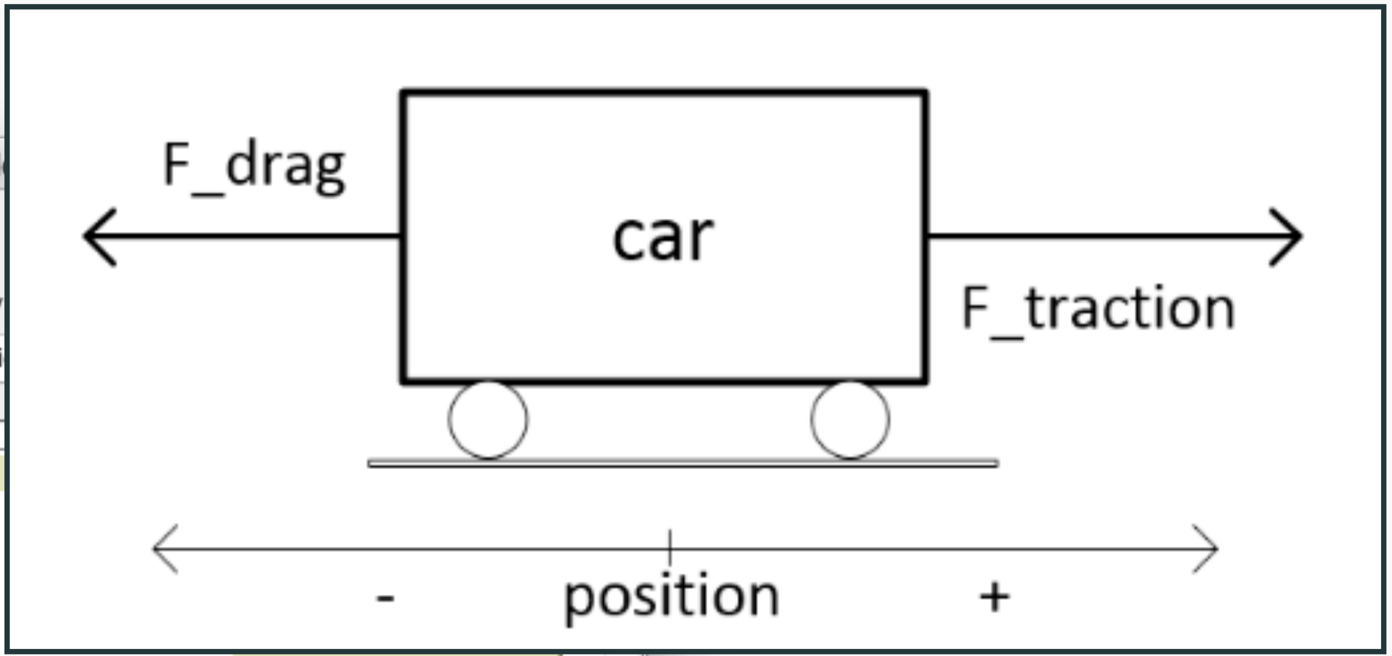
Michel, Fabien. 2007. “The IRM4S Model: The Influence/Reaction Principle for Multiagent Based Simulation.” In Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, 1–3. New York, USA: ACM Press.

Operational semantics

```
def simulate(abm: ABM) {
  time = 0
  env = abm.env
  env.state = env.initial_state
  for (ag in abm.agents) {
    ag.state = ag.initial_state
  }
  while (not termination_condition()) {
    influences = []
    for (ag in abm.agents) {
      percept = ag.percept(env.state)
      ag.state = ag.mem(percept, ag.state)
      influences.add(ag.decision(percept, ag.state))
    }
    influences.add(env.natural(percept, ag.state))
    env.state = reaction(env.state, influences)
    time += 1
  }
}
```

Case study


NetLogo — TrafficCruiseControl1 {/Users/romain/Documents/Univ/Post-d
Interface
Info
Code
Edit
Delete
Add
Button
normal speed
ticks: 208
on ti
number-of-cars 3
setup
go
red car speed
0.005
max-acceleration 0.09
max-deceleration 0.11
red-car-desired-speed 0.00
Car speeds
1.1
speed
0.431
0
86
0
time
300
Command Center
Clear
observer>


F_drag
car
F_traction
-
position
+