Pietro Terna[#], Irene Bonafine[♪]
[#]Department of Economics and Public Finance, University of Torino, Italy
terna@econ.unito.it
[♪]Department of Economics, University of Torino, Italy
irene.bonafine@unito.it

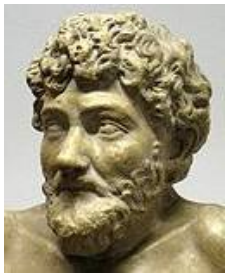
Agent-based modeling as a flexible technique to join the RTGS system and the Money Market: methodological and implementation issues

BoF-PSS - 8th Simulator Seminar and Workshop on 26 - 27 August 2010 – Bank of Finland

A technical premise

SLAPP, or Swarm-Like Agent Protocol in Python, is a simplified implementation of the original Swarm protocol, choosing Python as a simultaneously simple and complete object-oriented framework.

SLAPP is evolving to AESOP



AESOP (Agents and Emergencies for Simulating Organizations in Python), written upon SLAPP as a simplified way to describe and generate interaction within artificial agents:

- *bland* agents (simple, unspecific, basic, insipid, …) doing basic actions;
- *tasty* agents (specialized, with given skills, acting in a discretionary way, …), playing specify roles into the simulation scenario.

The current package for ”Agent based simulation model of RTGS system and money market”, or Payments and Money market model (in short: P&3M), is a special implementation of SLAPP.

In perspective, it will evolve to a (special) AESOP application.

Why a new tool and why SLAPP (Swarm-Like Agent Based Protocol in Python) as a preferred tool?

- For didactical reasons, applying a such rigorous and simple object oriented language as Python
- To build models upon transparent code: Python does not have hidden parts or feature coming from magic, it has no obscure libraries
-
- **To use the openness of Python**
- **To apply easily the SWARM protocol (www.swarm.org)**

The openness of Python (www.python.org)

- … going from Python to R
  (R is at http://cran.r-project.org/ ;
  rpy/rpy2 libraries are at http://rpy.sourceforge.net/ or, a novelty,
  PypeR at http://rinpy.sourceforge.net)

- … going from OpenOffice (Calc, Writer, …) to Python and viceversa (via the Python-UNO bridge, incorporated in OOo)

- … doing symbolic calculations in Python (via http://code.google.com/p/sympy/)

- … doing declarative programming with PyLog, a Prolog implementation in Python (http://christophe.delord.free.fr/pylog/index.html)

- … using Social Network Analysis from Python; examples:
  - Igraph http://cneurocvs.rmki.kfki.hu/igraph/
  - libsna http://www.libsna.org/
  - PySNA http://www.cs.bilgi.edu.tr/~mgencer/software.html#pySNA

The SWARM protocol

SLAPP is **a demonstration that we can easily implement the Swarm protocol** [Minar, N., R. Burkhart, C. Langton, and M. Askenazi (1996), *The Swarm simulation system: A toolkit for building multi-agent simulations*. Working Paper 96-06-042, Santa Fe Institute, Santa Fe (*)] **in Python**
(*) http://www.swarm.org/images/b/bb/MinarEtAl96.pdf

Key points (quoting from that paper):

- *Swarm defines a structure for simulations, a framework within which models are built.*
- *The core commitment is to a discrete-event simulation of multiple agents using an object-oriented representation.*
- *To these basic choices Swarm adds the concept of the **"swarm,"** a **collection of agents** with a **schedule of activity**.*

The container:

the Swarm protocol

Swarm = a library of functions and a **protocol**

modelSwarm

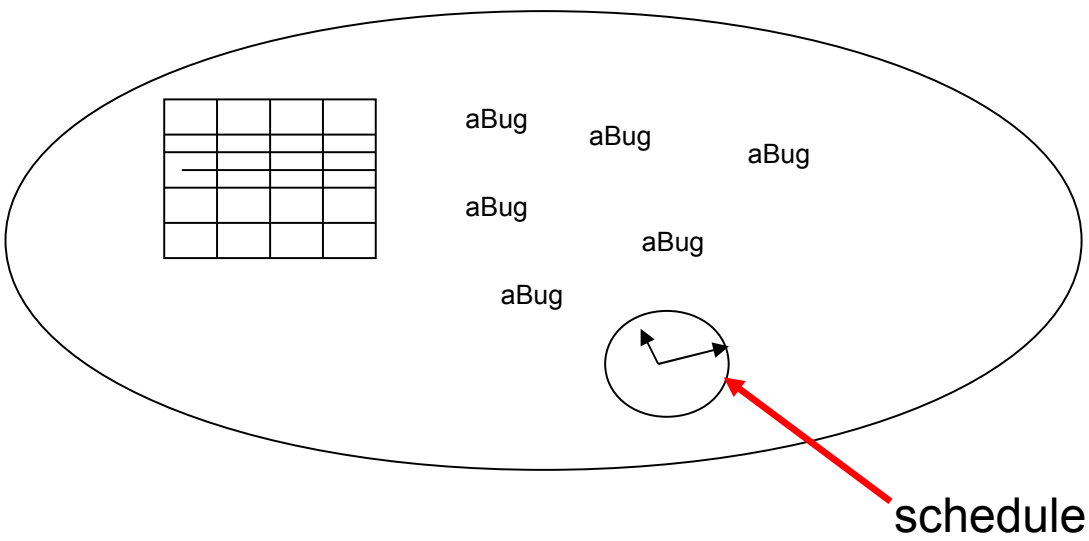
create agents

create actions

run modelSwarm

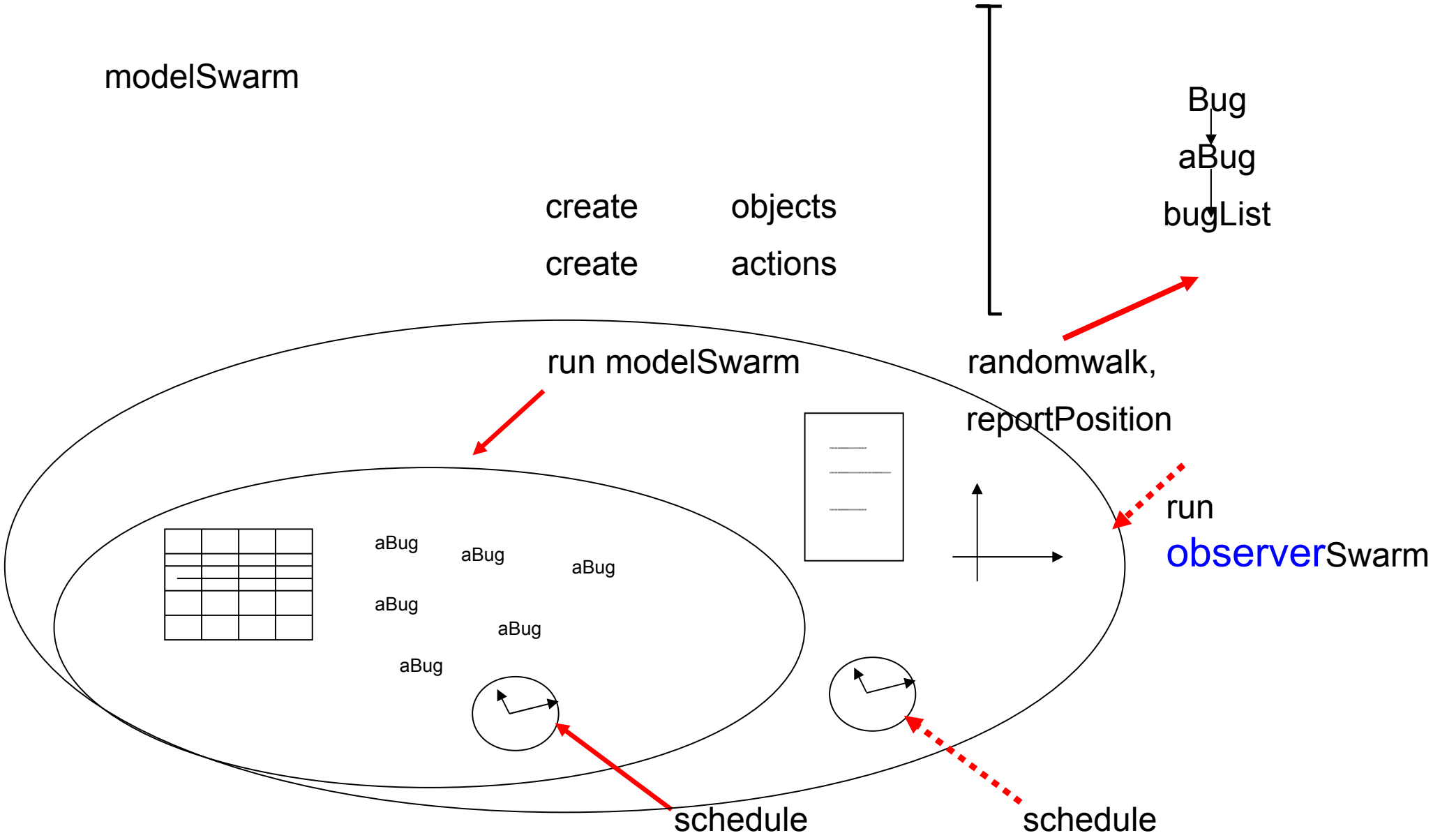
Bug
aBug
bugList

randomwalk,
reportPosition

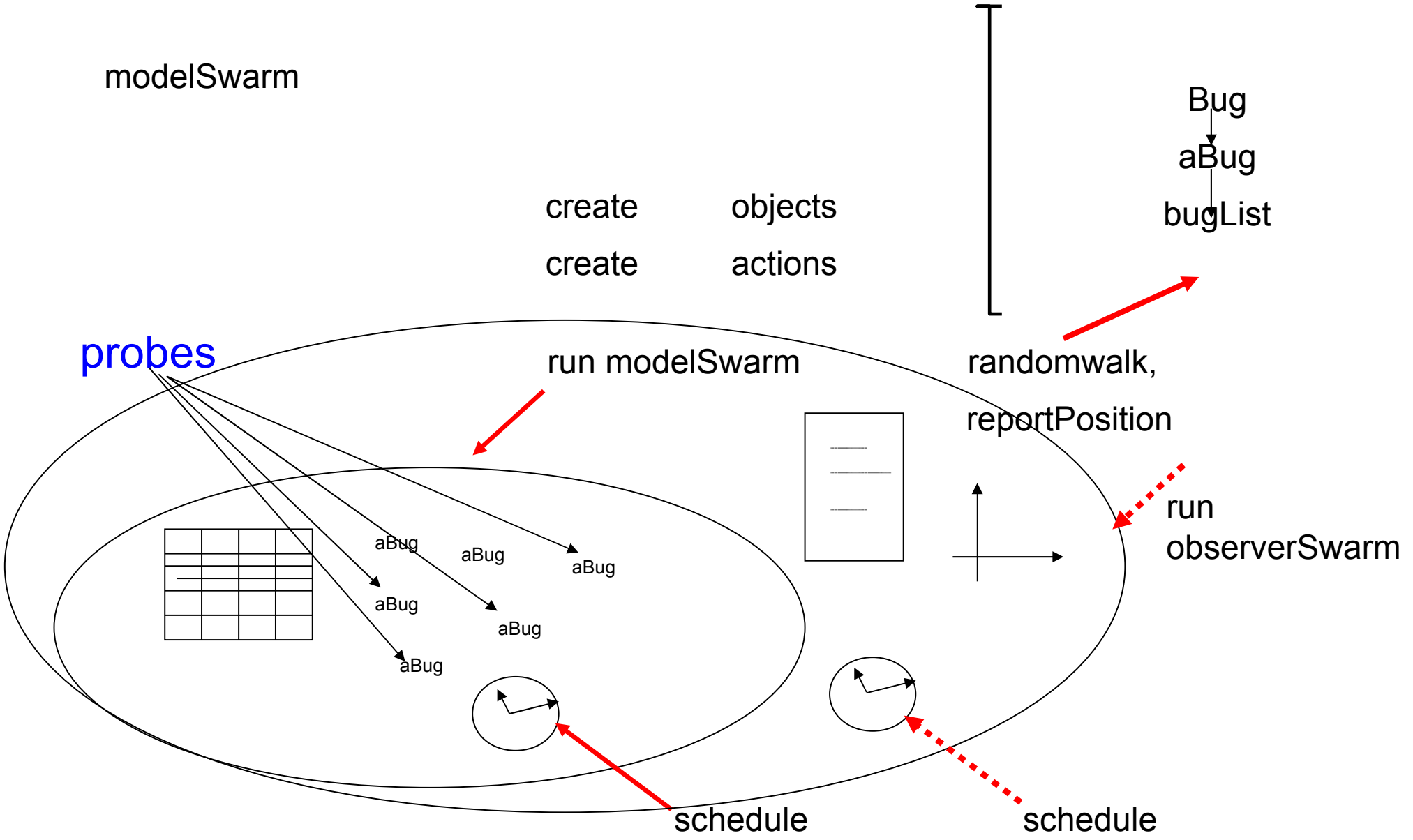


schedule

Swarm = a library of functions and a **protocol**

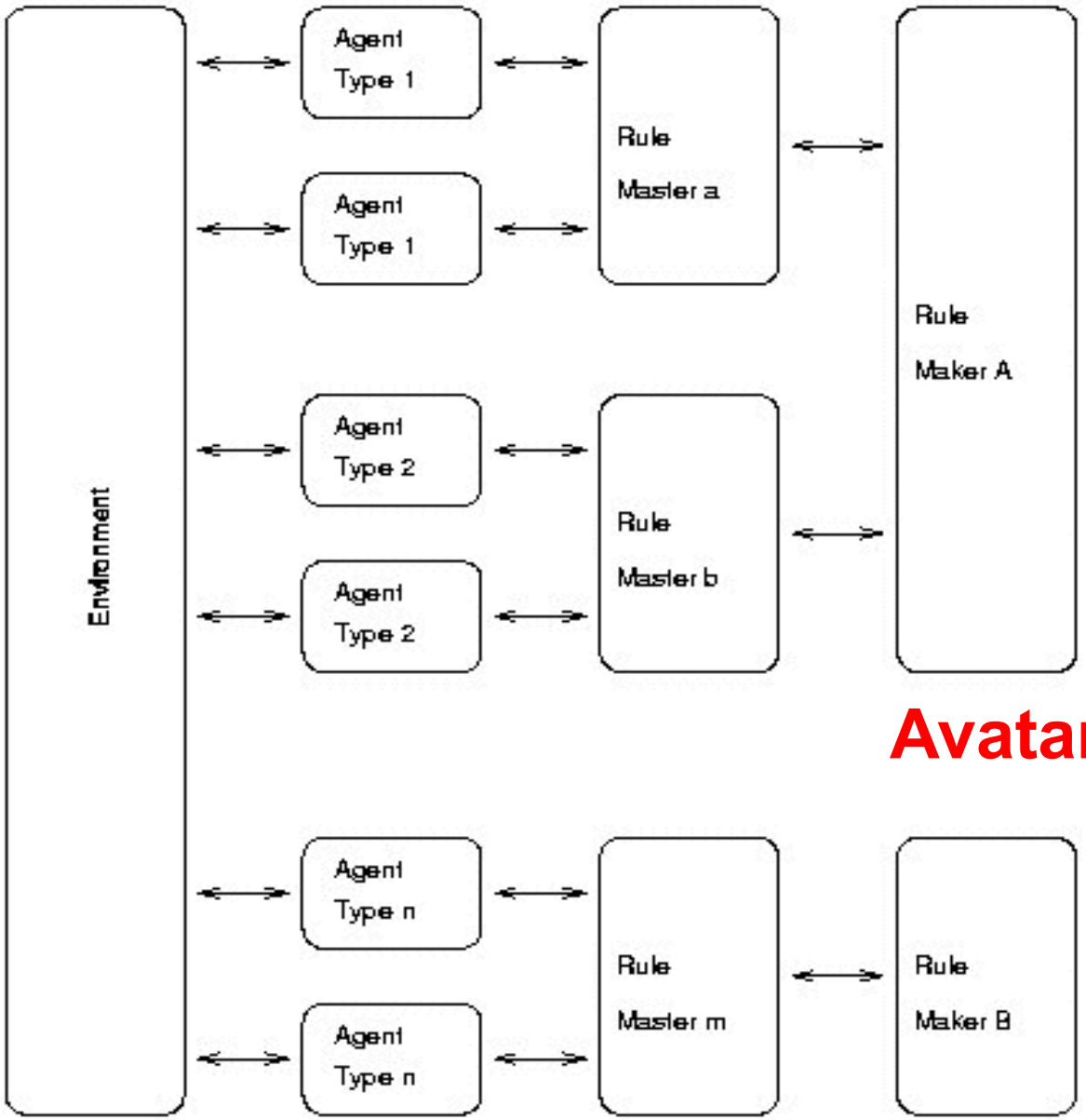


Swarm = a library of functions and a **protocol**



The content:

the ERA scheme (Environment, Agents and Rules)



Fixed rules

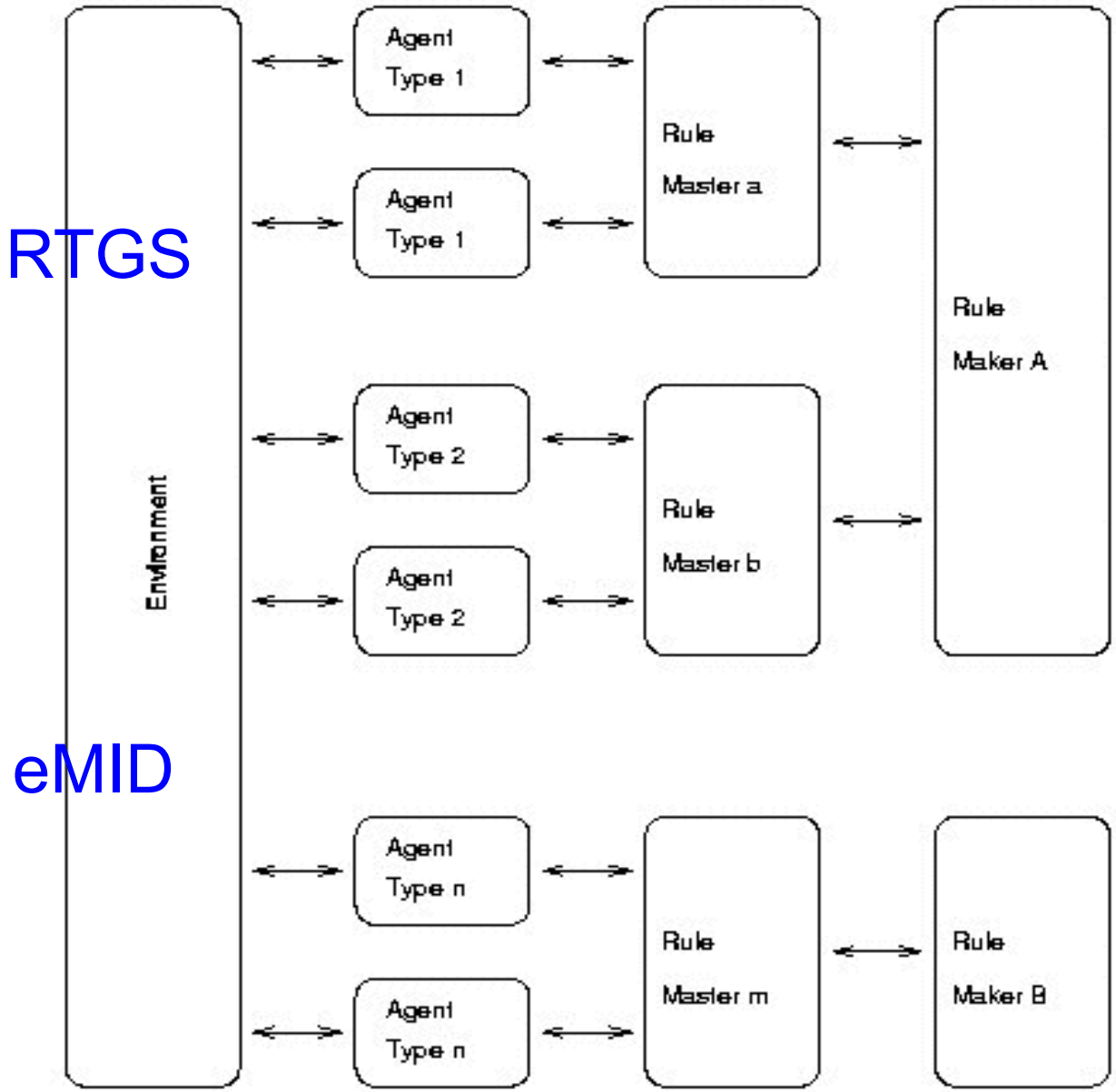
NN

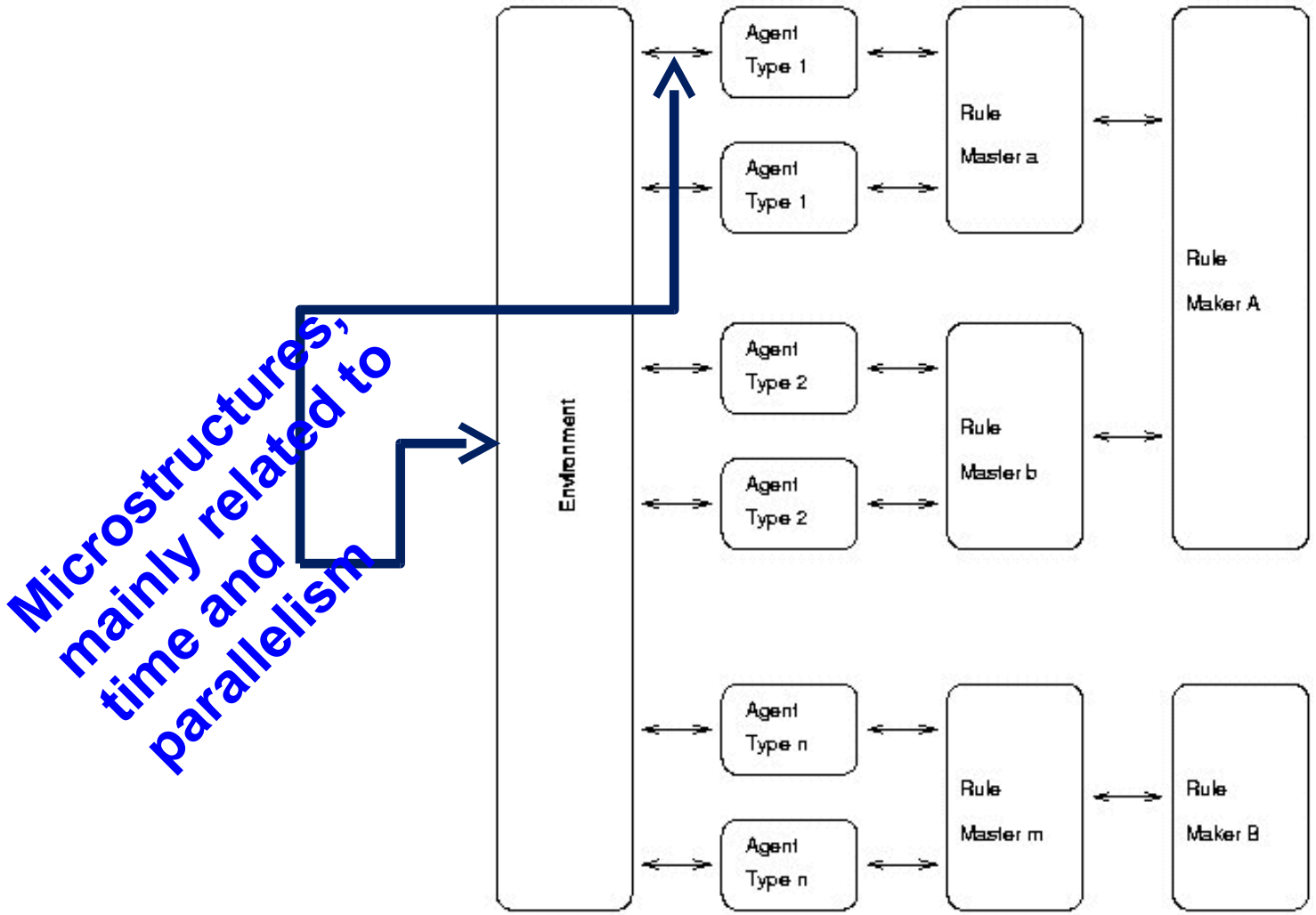
CS

GA

Reinforcement learning

Avatar


RTGS
eMID
Environment
Agent
Type 1
Agent
Type 1
Agent
Type 2
Agent
Type 2
Agent
Type n
Agent
Type n
Rule
Master a
Rule
Master b
Rule
Master m
Rule
Maker A
Rule
Maker B



http://web.econ.unito.it/terna/ct-era/ct-era.html

Eating the pudding:

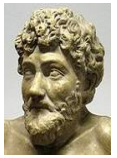
let's play with P&3M

Details about P&3M (i) technicalities and (ii) use are reported in a conceptual map on line at

http://eco83.econ.unito.it/terna/P&3M/P&3M.html

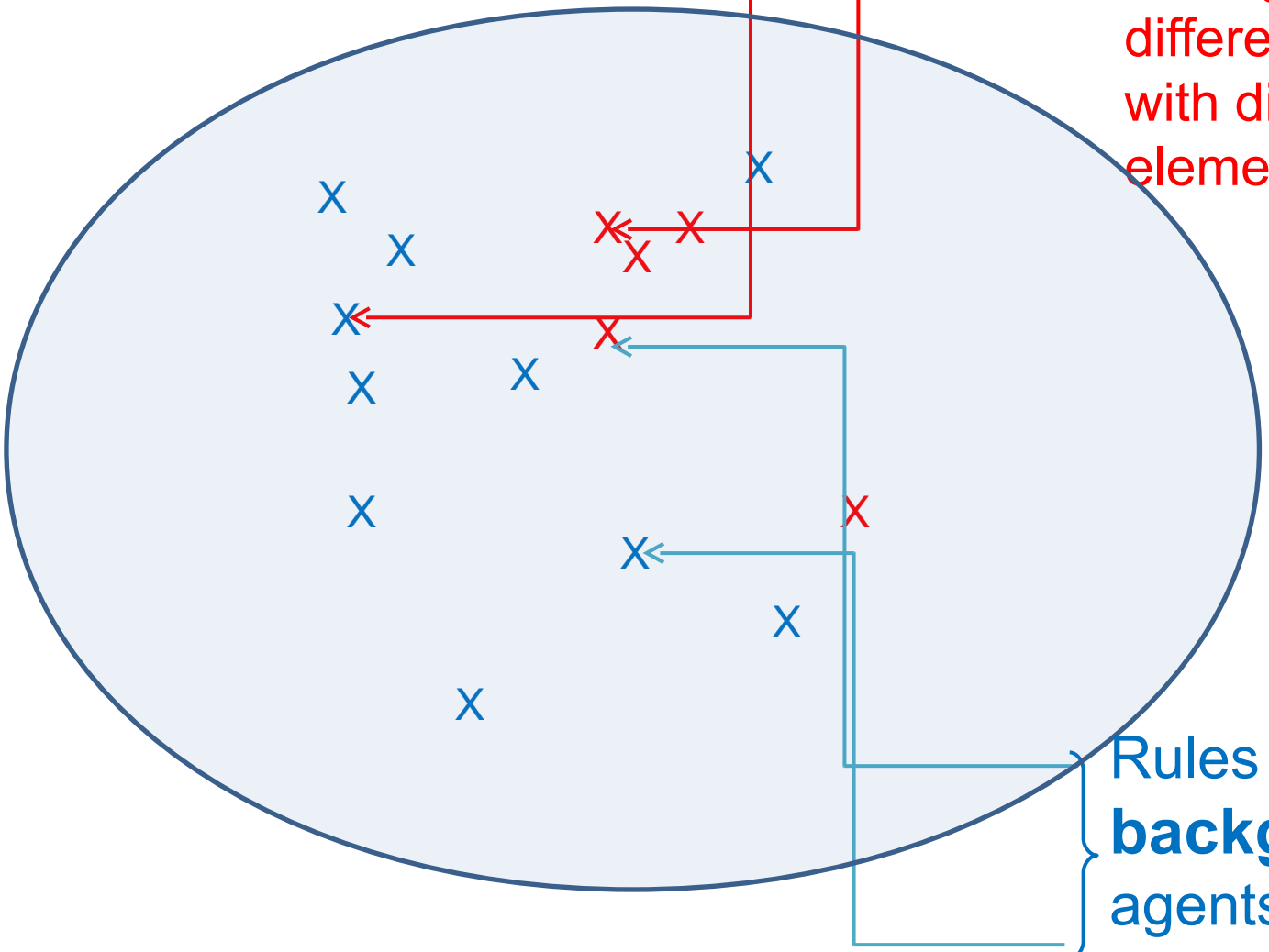
The future improvements:

Aesop



1. Agents and schedule

Bland* and **tasty#** agents



Rules operating "in the **foreground**", explicitly managed via scripts (on different sets of agents, with different numbers of elements)

Rules operating "in the **background**" for all the agents, or only for the blue ones or for a few specific sets of tasty agents

**Bland = simple, unspecific, basic, insipid, …*
#Tasty = specialized, with given skills, discretionary, …

Schedule driving agents (only bland agents here, no tasty agents)

Agent -> all agents; Agent0 -> bland agents; in this case the two sets are coincident

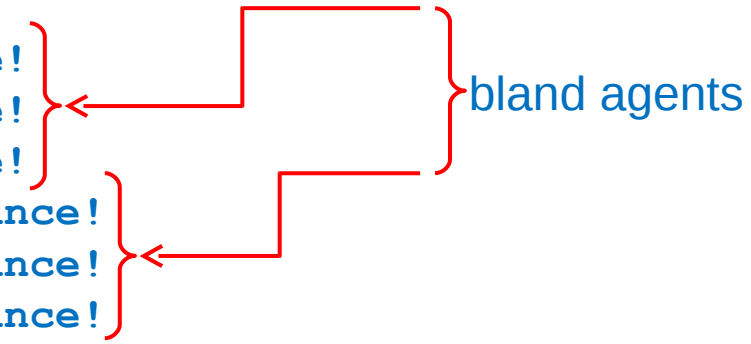
1	#		1		standard (background) actions, like move, are applied to all agents
2	Agent0	eat			0 means internal agents
3	Agent0	dance			
4	#		2		
5	#		4		
6	Agent		0.5	dance	Agent without speficitation means all
7	Agent3	eat			3 means agents of type 3
8	WorldState		0.5	setGeneralMovingProb	
9	#		5		comments here or in successive columns
10	Agent	eat			
11	Agent	dance			
12	#		30		
13	Agent1	dance			1 means agents of type 1
14	#		31		
15	Agent1		0.5	dance	
16					

Empty sets, in this case

Acting on bland (blue) agents and on tasty (red) ones

```
How many 'bland' agents? 3
X Size of the world? 10
Y Size of the world? 10
How many cycles? (0 = exit) 5
World state number  0  has been created.
Agent number  0  has been created at  7 ,  1
Agent number  1  has been created at  3 ,  2
Agent number  2  has been created at  7 ,  0

Time = 1
agent #  1 moving
agent #  2 moving
agent #  0 moving
I'm agent  1:  nothing to eat here!
I'm agent  2:  nothing to eat here!
I'm agent  0:  nothing to eat here!
I'm agent  0:  it's not time to dance!
I'm agent  1:  it's not time to dance!
I'm agent  2:  it's not time to dance!
Time =  1 ask all agents to report position
Agent number  0  moved to X =  0.9726902013  Y =  7.0273097987
Agent number  1  moved to X =  6.9726902013  Y =  2.0
Agent number  2  moved to X =  7.0  Y =  6.0273097987
```



Schedule driving agents (bland and tasty ones)

Agent -> all agents; Agent0 -> background agents

1	#		1		standard (background) actions, like move, are applied to all agents
2	Agent0	eat			0 means internal agents
3	Agent0	dance			
4	#		2		
5	#		4		
6	Agent		0.5	dance	Agent without speficitation means all
7	Agent3	eat			3 means agents of type 3
8	WorldState		0.5	setGeneralMovingProb	
9	#		5		comments here or in successive columns
10	Agent	eat			
11	Agent	dance			
12	#		30		
13	Agent1	dance			1 means agents of type 1
14	#		31		
15	Agent1		0.5	dance	
16					

Non empty sets, in this case

Effects on bland (blue) agents and tasty (red) ones

agType3.txt

1111

```
How many 'bland' agents? 3
X Size of the world? 3
Y Size of the world? 3
How many cycles? (0 = exit) 32
World state number  0  has been created.
Agent number  0  has been created at  0 ,  2
Agent number  1  has been created at  1 ,  0
Agent number  2  has been created at  0 ,  2

creating agType1 # 111
Agent number  111  has been created at  1 ,  1
creating agType1 # 222
Agent number  222  has been created at  2 ,  0

creating agType3 # 1111
Agent number  1111  has been created at  2 ,  2

Time = 1
agent #  2 moving
agent # 222 moving
agent #  0 moving
agent # 111 moving
agent #  1 moving
agent # 1111 moving
```

tasty agents

bland and tasty agents

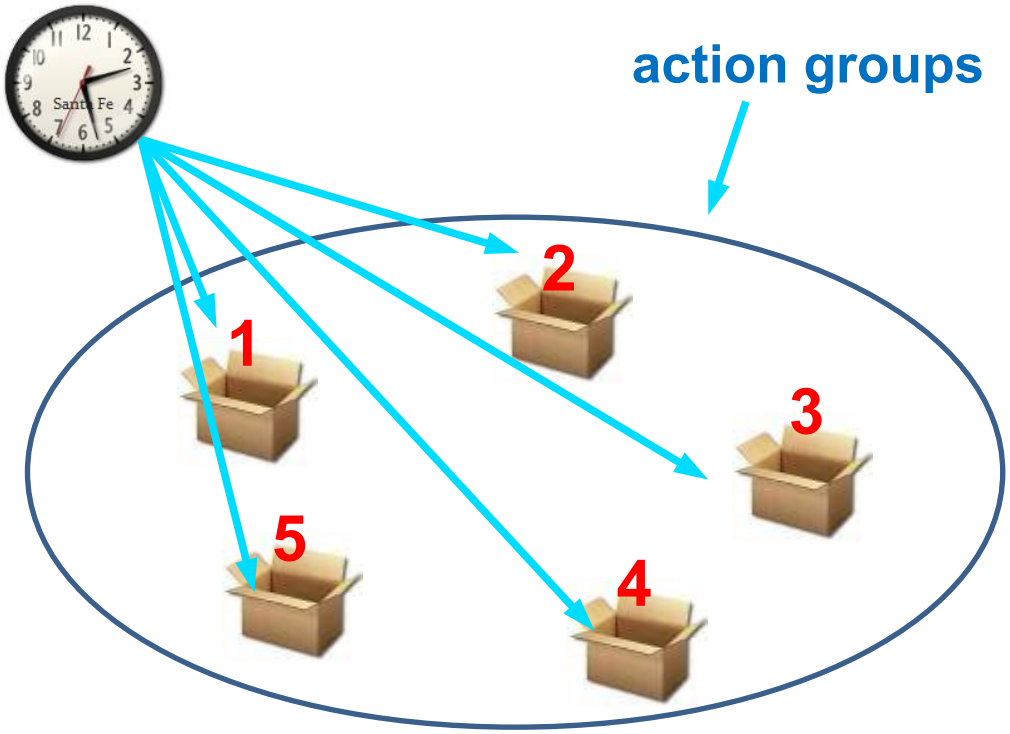
```
Time = 5
agent # 222 moving
agent #  1 moving
I'm agent 1111:  nothing to eat here!
I'm agent  2:  nothing to eat here!
I'm agent 111:  nothing to eat here!
I'm agent  0:  nothing to eat here!
I'm agent 222:  nothing to eat here!
I'm agent  1:  nothing to eat here!
I'm agent  0:  it's not time to dance!
I'm agent 222:  it's not time to dance!
I'm agent 1111:  it's not time to dance!
I'm agent  2:  it's not time to dance!
I'm agent 111:  it's not time to dance!
I'm agent  1:  it's not time to dance!
```

```
Time =31
agent #  1 moving
agent # 111 moving
agent #  0 moving
agent #  2 moving
I'm agent 222:  it's not time to dance!
Time =  31 ask all agents to report position
```

#	31	
Agent1	0.5	dance

Aesop

2. Schedule improvement



What in each box?

Tasks to be executed (with p=1 or with p<1)

Tasks are included into the code in a static way, or can be added/activated dynamically by other tasks, also via agents' actions

Tasks can be read – via a 'read' task schedule element – from an external source (file, web interaction, …)

A special type of task to be read from an external source is that of the recipes

tasks read from an external archive

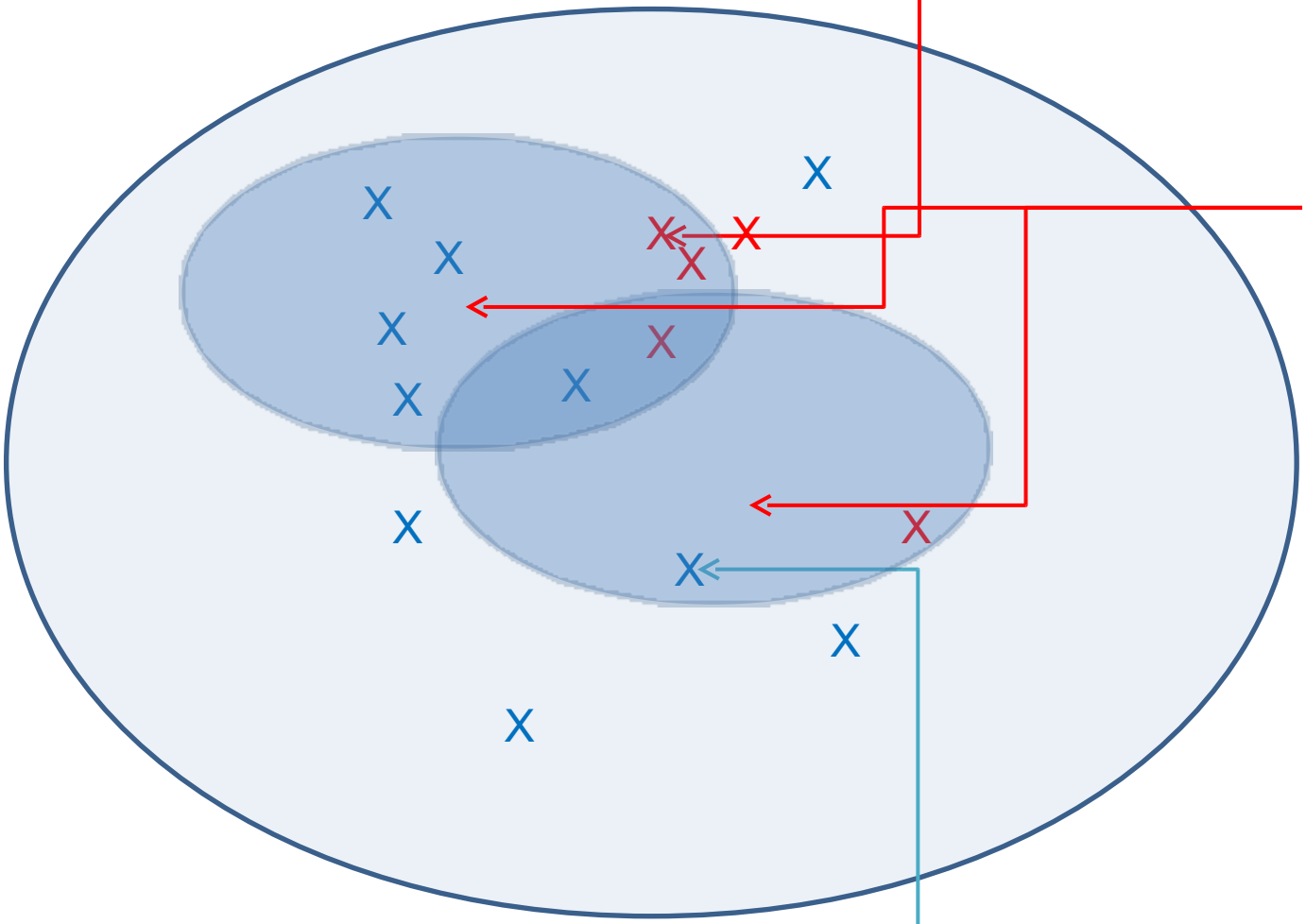
a_n – a specific agent (instance of class A)
a_X – a randomly chosen agent (instance of class A)
a_%all – a quota of all the agents (instances) of the class A
a_all – all the agents (instances) of the class A

methods specific of each agent or inherited from the basic type ‘agent’

Aesop

3. Artificial neural networks into the agents

bland and **tasty** agents can **contain** ANNs



ANN

Networks of ANNs, built upon agent interaction

ANN

$$y = g(x) = f(B\ f(A\ x))$$

(m) (n)

actions information

or

$$y_1 = g_1(x) = f(B_1\ f(A_1\ x))$$

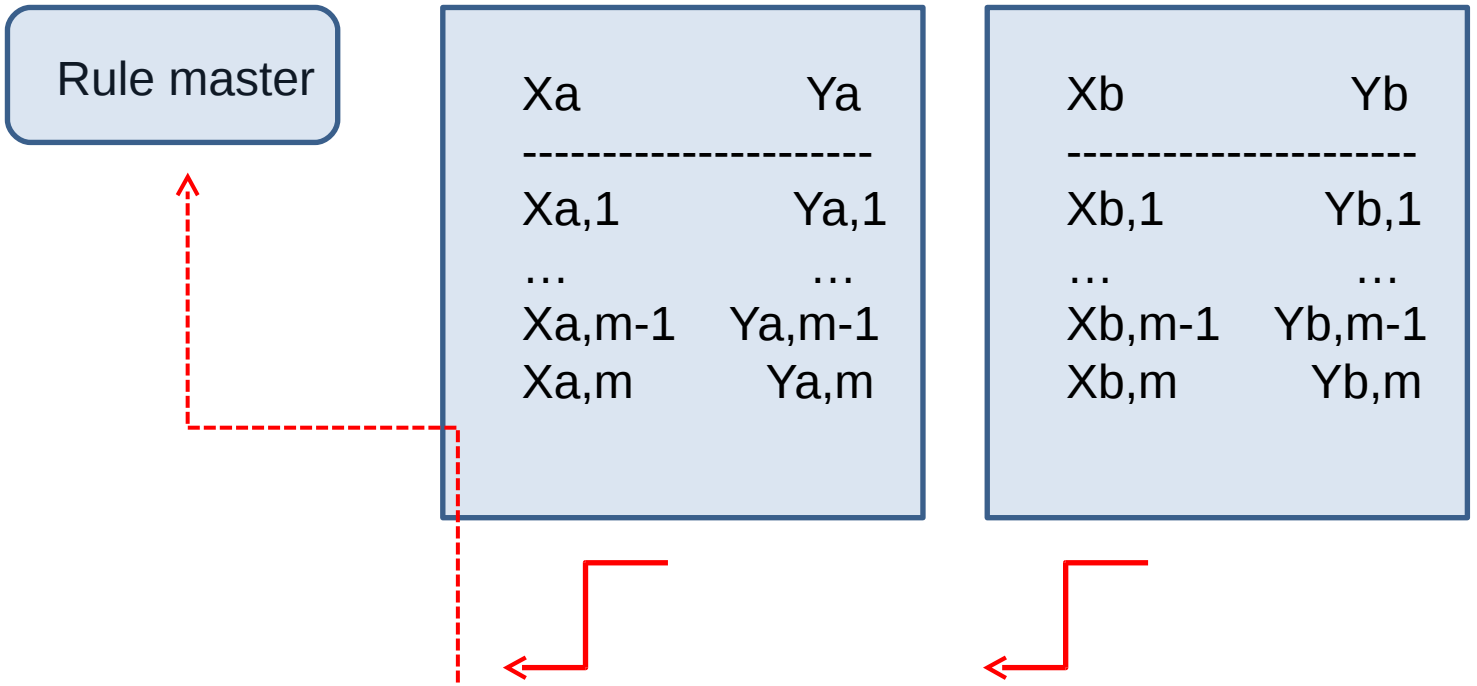
(1) (n)

…

$$y_m = g_m(x) = f(B_m\ f(A_m\ x))$$

(1) (n)

a - Static ex-ante learning (on examples)



Different agents, with different set of examples, estimating and using different matrixes A and B of parameters

b - Continuous learning (trials and errors)

$$z = g([x,y]) = f(B\, f(A\, [x,y]))$$

(p)

(n+m)

Rule master

accounting for laws/norms

actions (allowed, forbidden)

information

effects

Coming from simulation

accounting for social norms

the agents will choose Z maximizing:

(i) individual U, with norms

(ii) societal wellbeing

Emergence of new norms **[modifying U=f(z) , as new norms do]** and laws **[modifying the set y, as new laws/rules do]**

Also U=f(z) can be a NN

In a starting phase, or at given time step, all or a few agents act randomly

Thanks

Pietro Terna[♯], Irene Bonafine[♪]
[♯]Department of Economics and Public Finance, University of Torino, Italy
terna@econ.unito.it
[♪]Department of Economics, University of Torino, Italy
irene.bonafine@unito.it