



Pietro Terna

Department of Economics and Public Finance, University of Torino, Italy

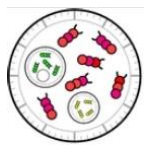
terna@econ.unito.it

web.econ.unito.it/terna

SLAPP and AESOP to run scripts of ABMs



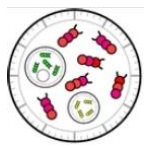
Abstract and outline



2009 SwarmFest - SLAPP

SLAPP, [Swarm-Like Agent Protocol in Python](#), as a simplified implementation of the original Swarm protocol, choosing Python as a simultaneously simple and complete object-oriented framework.

The SLAPP project has also the goal of offering to scholars interested in agent-based models a set of programming examples that could be easily understood in all their details and adapted to other applications.

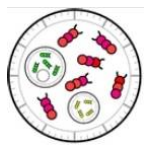


2010 SwarmFest - AESOP



AESOP, ([Agents and Emergencies for Simulating Organizations in Python](#)), written upon SLAPP and intended to be a simplified way to describe and generate interaction within artificial agents.

- bland agents (simple, unspecific, basic, insipid, ...) used to populate our simulated world with agents doing basic actions;
- tasty agents (specialized, with given skills, acting in a discretionary way, ...), used to specify important roles into the simulation scenario.



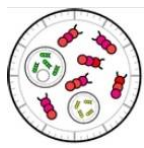
2011 SwarmFest - AESOP

Adding to AESOP/SLAPP a further step:

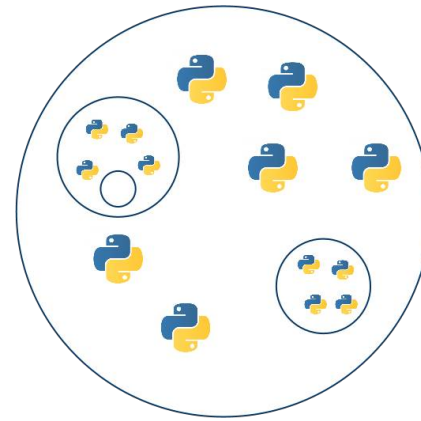
The capability of running sophisticated scripts, written into spreadsheets, describing (flexible) simulation situations, that the users can easily implement and modify ...

... also with internal IF structures, **opening the perspective of repeated trials and errors executions**, to take advantage from reinforcement learning technique.

With the ABM we produce trials and errors cases, extracting knowledge from them via artificial neural networks.



2011 SwarmFest - AESOP



A huge **surprise**: thanks to Steve Rogers, we have now a parallel SLAPP implementation, running in **OMQ**,

<http://zeromq.org/>

(see <http://code.google.com/p/slapp-tools/>).

So ...

OSLAPP is born

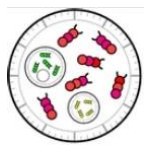


A quick look to **SLAPP**
Swarm-Like Agent Based Protocol in Python



Why a new tool?

- For didactical reasons, applying a such rigorous and simple object oriented language as Python is
- To build models upon transparent code: Python does not have hidden parts or features coming from *magic*, it has no obscure libraries
- **To use the openness of Python**
- **To have the possibility of using the key feature of the Swarm protocol in an easy way**



About Python and ABMs have also a look to
the 2011 Alan Isaac paper at

<http://jasss.soc.surrey.ac.uk/14/2/5.html>



[Alan G. Isaac](#) (2011)

The ABM Template Models: A Reformulation with Reference Implementations

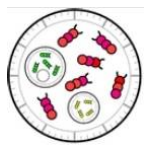
Journal of Artificial Societies and Social Simulation **14** (2) 5
<<http://jasss.soc.surrey.ac.uk/14/2/5.html>>



The openness of Python (www.python.org)



- ... going from Python to R
(R is at <http://www.r-project.org/> ;
rpy library is at <http://rpy.sourceforge.net/>)
- ... going from OpenOffice (Calc, Writer, ...) to Python and viceversa (via
the Python-UNO bridge, incorporated in OOo and LibreOffice)
- ... doing symbolic calculations in Python (via
<http://code.google.com/p/sympy/>)
- ... doing declarative programming with PyLog, a Prolog implementation
in Python (<http://www.cdsoft.fr/pylog/index.html>)
- ... using Social Network Analysis from Python; examples:
 - Igraph <http://cneurocv.s.rmki.kfki.hu/igraph/>
 - NetworkX <http://networkx.lanl.gov/>



The SWARM protocol

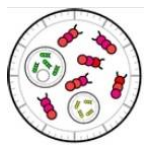


What is SLAPP: basically a **demonstration that we can easily implement the Swarm protocol** [Minar, N., R. Burkhart, C. Langton, and M. Askenazi (1996), *The Swarm simulation system: A toolkit for building multi-agent simulations*. Working Paper 96-06-042, Santa Fe Institute, Santa Fe (*)] **in Python**

(*) <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 SWARM protocol



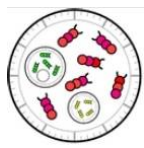
An absolutely clear and rigorous application of the SWARM protocol is contained in the original SimpleBug tutorial (1996?) with **ObjectiveC** code files, and text files, by Chris Langton & Swarm development team (Santa Fe Institute); it is on line at <http://ftp.swarm.org/pub/swarm/apps/objc/sdg/swarmapps-objc-2.2-3.tar.gz> (into the folder “tutorial”, with the texts reported into the README files both in the first tutorial folder and in the internal subfolders)


The same has also been adapted to **Java** by Charles J. Staelin (*jSIMPLEBUG, a Swarm tutorial for Java*, 2000), at <http://www.cse.nd.edu/courses/cse498j/www/Resources/jsimplebug11.pdf> (text only) or <http://eco83.econ.unito.it/swarm/materiale/jtutorial/JavaTutorial.zip> (text and code)

At <http://eco83.econ.unito.it/terna/slapp> you can find the same structure of files, but now implementing the SWARM protocol using **Python**



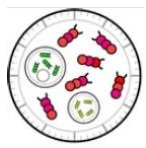
The SLAPP package at
<http://eco83.econ.unito.it/terna/slapp>



- ▶ 1 plainProgrammingBug
- ▶ 2 basicObjectProgrammingBug
- ▶ 3 basicObjectProgrammingManyBugs
- ▶ 4 basicObjectProgram...s_bugExternal+_shuffle
- ▶ 5 objectSwarmModelBugs
- ▶ 6 objectSwarmObserverAgents_AESOP_turtleLib 
- ▶ 7 toBeDeveloped objectSwarmObserverTkBugs
- ▶ 7b toBeDeveloped Tk test
- ▶ 8 toBeDeveloped simpleExpertBug
- ▶ readme.txt
- ▶ SLAPP 0 tutorial.txt
- ▶ Swarm_original 0 tutorial.txt



- 📄 a_note_on_AESOP.txt
- 📄 ActionGroup.py
- ▶ 📁 basic
- 📄 convert_xls_txt.py
- 📄 ModelSwarm.py
- 📄 ObserverSwarm.py
- 📄 Pen.py
- ▶ 📁 school
- 📄 SLAPP 6 objectSwarmObserverAgents.txt
- 📄 start 6 objectSwarmObserverAgentsAESOP.py
- 📄 Swarm_original 7 simpleObserverBug.txt
- 📄 Swarm_original 8 simpleObserverBug2.txt
- 📄 Tools.py
- 📄 WARNING.txt



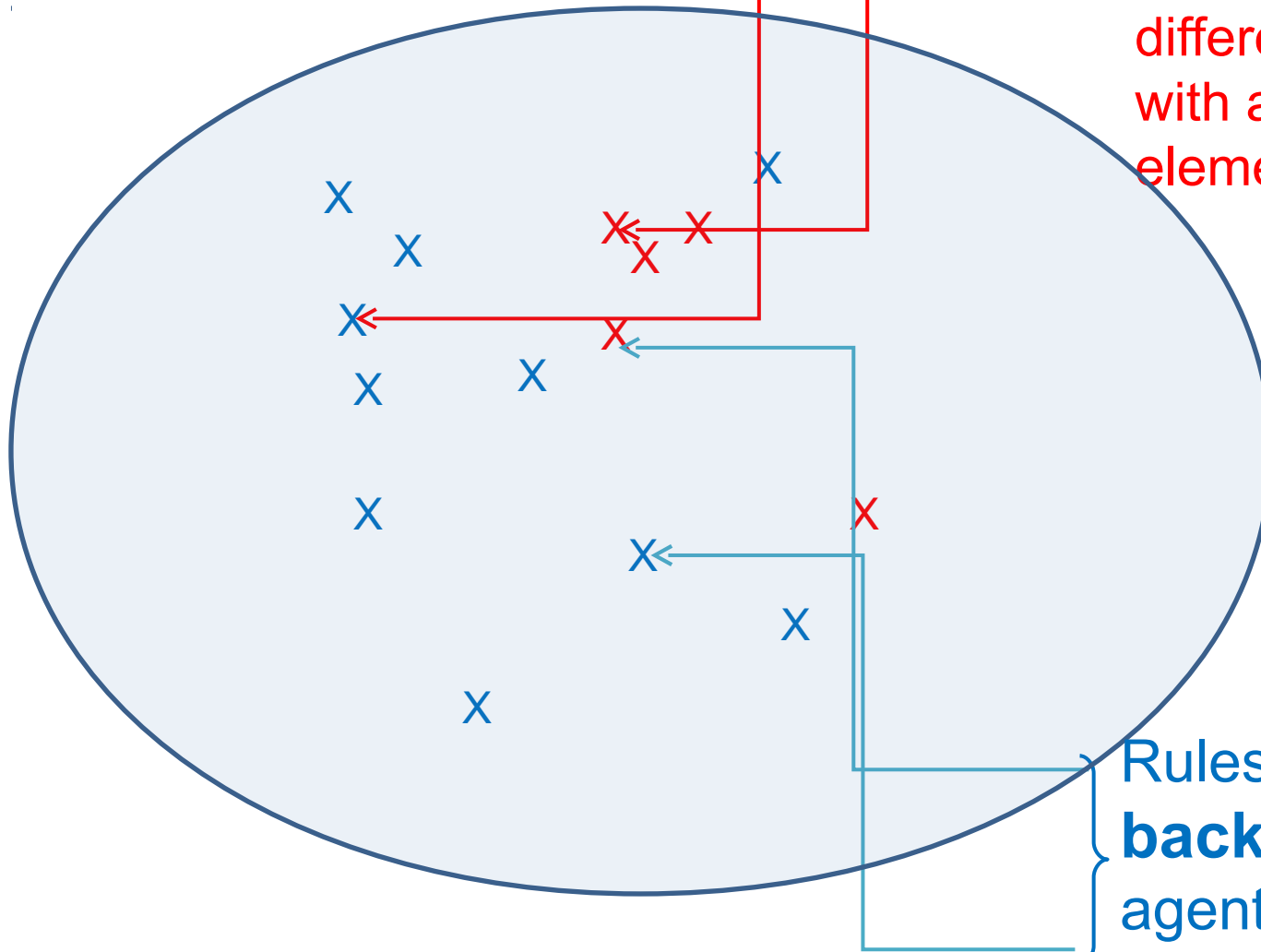
Agents and schedule



Bland* and tasty# agents

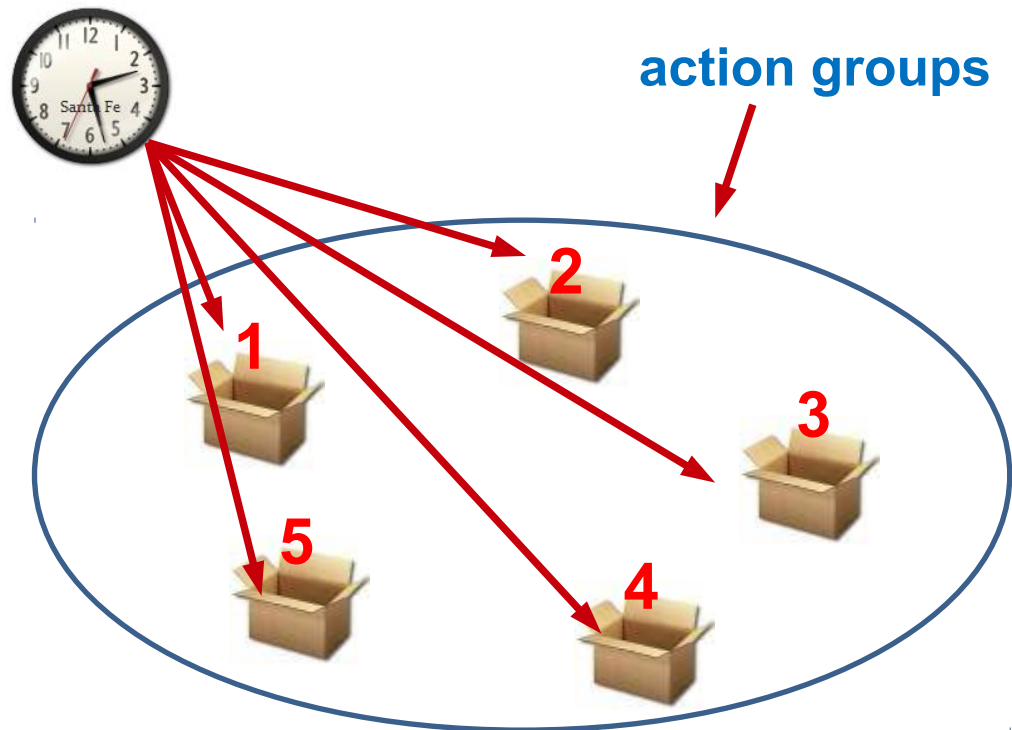
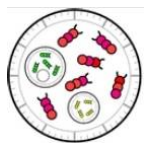


Rules operating “in the **foreground**”, explicitly managed via a script (with different sets of agents, with a different number of elements)



Rules operating “in the **background**” for all the agents, or only for the blue ones or for a specific set of tasty agents

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



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

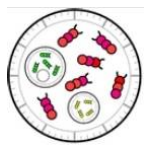
a_all – all the agents (instances) of the class A



[agent method]



methods specific of each agent or inherited from the basic types 'Agent' or 'Turtle'



recipes, as macro, read from an external archive

[[agent method] [agent method] [agent method] [agent method] [agent method] ...]

→ to be executed in a sequential way by a given thread of agents (statically or dynamically)

[[agent method] **[[agent method] [agent method] [agent method]]** [agent method] ...]

with segments to be executed in a parallel way

[[agent method] [agent method] **N**[agent method] [agent method] [agent method] ...]

[..... :		: [agent method] :]
[..... :		:]
[..... :		: [agent method] :]

with components belonging to different recipes to be executed as a whole



An example, studying pupils behavior
in primary school classes (with Sandro
Brignone, Aldo Milano and Tiziana Pasta)



Our goal: to discover how to improve attention and active participation in school.





schedule.xls - LibreOffice Calc

Calibri 11

E86 \sum = bisognerebbe evidenziare il fatto che i b. eseguono la consegna con velocità diverse

	A	B	C	D
1				th
2	macro	repeatedAction		VI
3	#		1	
4	macro	transitionPhase		
5	#		2	
6	macro	checkToObtainAttention		
7	#		3	
8	scPupil	askWell		IN
9	scPupil	talkTeacherWell		
10	macro	individualFocus		
11	#		4	
12	macro	assegnementEasy		
13	zCen		0,2 checkFastWork	
14	scPupil	checkWork		
15	#		5	
16	macro	revision		
17	macro	easyQuestion		
18	all		0,095 bePraised	do
19	all		0,7 fidget	au
20	all		0,286 doWork	Di

schedule transitionPhase checkToObtainAttention individualFocus assegnementEasy revision easyQuestion

Foglio 1 / 8 PageStyle_schedule STD Somma=0 115%



schedule.xls - LibreOffice Calc

Calibri 11

E86 \sum = bisognerebbe evidenziare il fatto che i b. eseguono la consegna con velocità diverse

	A	B	C	D
21	f3dx		0,6 checkTeacherTalkClose	
22	rossoPupil		0,16 tidy	
23	verdePupil		0,1 tidy	
24	#		6	
25	macro	revision		
26	macro	easyQuestion		
27	all		0,5 fidget	al
28	verdePupil		-1 doWork	D
29	verdePupil		0,1 tidy	tic
30	#		7	D
31	macro	revision		
32	verdePupil		0,1 attractTeacherAttentionWell	fu
33	verdePupil		0,4 beQuiteBored	fu
34	scPupil	talkTeacherBad		
35	scPupil	beScolded		
36	#		8	
37	macro	revision		
38	macro	easyQuestion		
39	rossoPupil		0,16 attractTeacherAttentionNotWell	
40	saPupil	tease		fu

schedule transitionPhase checkToObtainAttention individualFocus assignementEasy revision easyQuestion

Foglio 1 / 8 PageStyle_schedule STD Somma=0 115%



schedule.xls - LibreOffice Calc

Calibri 11

E15

	A	B	C	D	E	F	G	H
1	<u>gialloPupil</u>	sitDownNotWell						
2	<u>verdePupil</u>		-1 sitDownNotWell					
3	all		0,3 fidget					
4	<u>saPupil</u>		0,8 shake					
5	<u>verdePupil</u>		-1 shake					
6	<u>treV</u>	talkClose						
7	<u>f4dx</u>	talkClose						
8	<u>saPupil</u>		0,5 answerBad					
9	all		0,15 answerWell					
10	<u>verdePupil</u>		0,1 tidy		<u>questi possono anche essere messi fuori dal macrom</u>			
11	<u>verdePupil</u>		0,1 untidyTidy		"			
12	<u>verdePupil</u>		0,1 wellness		"			
13	<u>rossoPupil</u>		0,16 wellness		"			
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								

Foglio 3 / 8 PageStyle_checkToObtainAttention STD Somma=0 100%



schedule.xls - LibreOffice Calc

Calibri 11

A28

	A	B	C	D	E	F	G	H
1	all	doWork			DA TICK 4, 10,12,13			
2	verdePupil		0,056 wellness		per ora: diciamo a Tutti di eseguire conse			
3	verdePupil		0,167 checkClassmateWork		(in questo modo vedo che l'attenzione din			
4	neroPupil		0,125 checkClassmateWork		(do Work si lascia senza influ sull'attenzio			
5	gialloPupil		0,125 checkClassmateWork					
6	rossoPupil		0,042 checkClassmateWork		per ora: checkFastWork e checkWork (pro			
7	verdePupil		0,14 talkClose					
8	rossoPupil		0,083 talkClose					
9	saPupil		0,25 talkClose					
10	verdePupil		0,028 fidget					
11	rossoPupil		0,042 fidget					
12	saPupil		0,25 fidget					
13	neroPupil		0,125 fidget					
14	neroPupil		0,125 tidy					
15	verdePupil		0,056 tidy					
16	verdePupil		0,056 sitDownNotWell					
17	neroPupil		0,125 helpClassmate					
18	verdePupil		0,028 helpClassmate					
19								
20								
21								
22								
23								
24								

schedule transitionPhase checkToObtainAttention individualFocus assignmentEasy revision easyQuestion

Foglio 5 / 8 PageStyle_assignmentEasy STD Somma=0 100%



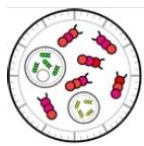
schedule.xls - LibreOffice Calc

Calibri 11

A28

	A	B	C	D	E	F	G	H	I	J
1	all	0,17	answerWell							
2	all	0,27	answerChorus							
3	verdePupil	0,019	answerWrong							
4	<u>gialloPupil</u>	0,08	answerWrong							
5	<u>rossoPupil</u>	0,028	answerWrong							
6	<u>scPupil</u>	0,17	answerWrong							
7	<u>saPupil</u>	0,33	answerBad							
8	<u>scPupil</u>	0,17	answerBad							
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										
20										
21										
22										
23										
24										

Foglio 7 / 8 PageStyle_easyQuestion STD Somma=0 100%



schedule.xls - LibreOffice Calc

Calibri 11

A1

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4	#									
5	all	payAttention								
6	all	sitDownWell								
7	#									
8	all	payAttention								
9	all	sitDownWell								
10	#									
11	all	payAttention								
12	all	sitDownWell								
13	#									
14	all	payAttention								
15	all	sitDownWell								
16	#									
17	all	payAttention								
18	all	sitDownWell								
19	#									
20	all	payAttention								
21	all	sitDownWell								
22	#									
23	all	payAttention								
24	all	sitDownWell								

checkToObtainAttention individualFocus assignementEasy revision easyQuestion repeatedAction

Foglio 8 / 8 PageStyle_repeatedAction STD Somma=0 100%



schedulelf.xls - LibreOffice Calc

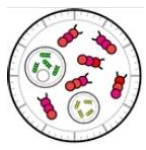
Calibri 11

C9

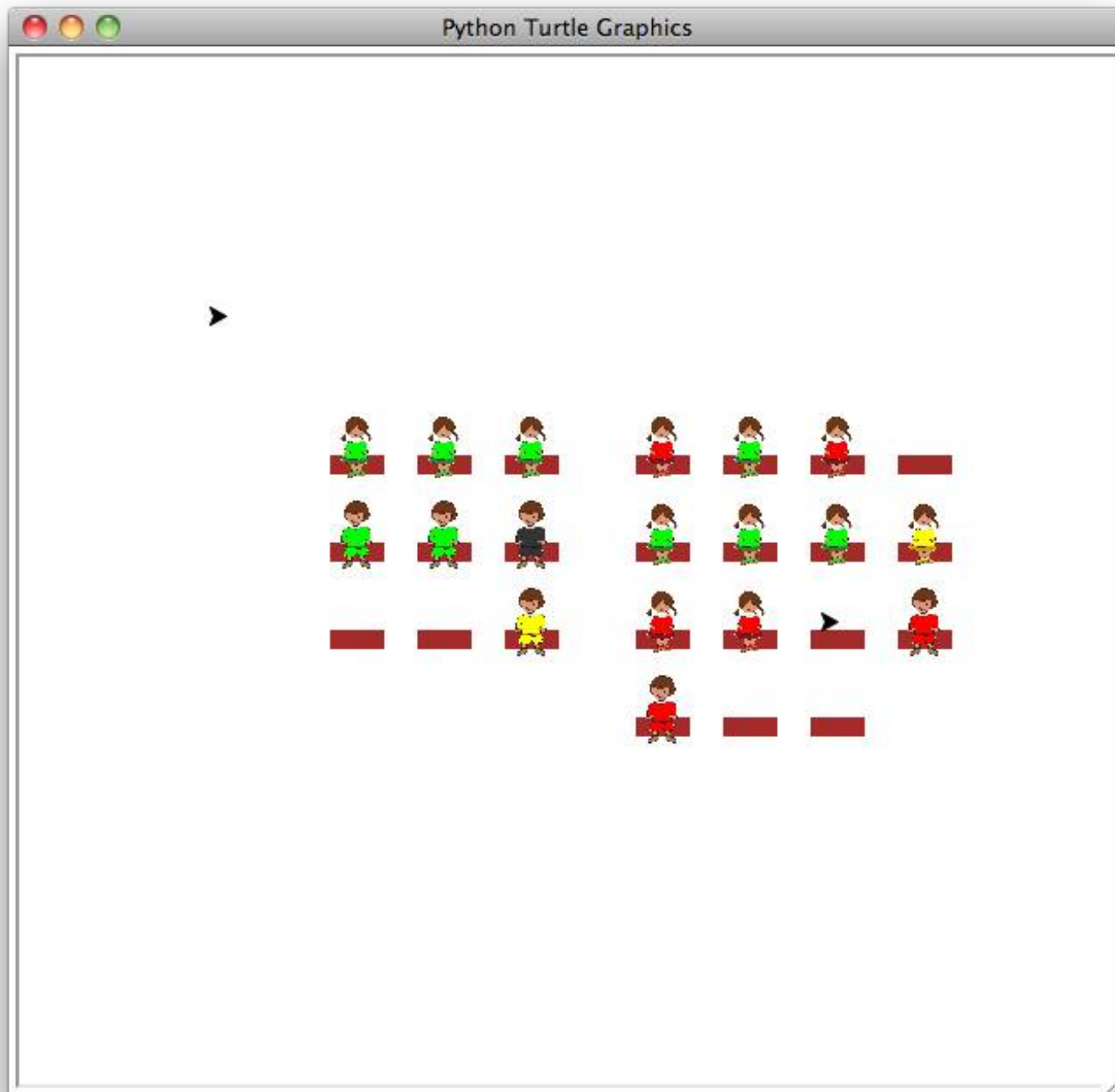
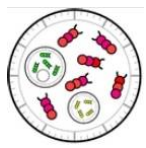
	A	B	C	D	E	F
1	all	payAttention				
2	all	sitDownWell				
3	gialloPupil	sitDownNotWell				
4	verdePupil		-1 sitDownNotWell			
5	all		0,3 fidget			
6	saPupil		0,8 shake			
7	verdePupil		-1 shake			
8	saPupil	shakelf_verdePupil				
9	treV	talkClose				
10	f4dx	talkClose				
11	saPupil		0,5 answerBad			
12	all		0,15 answerWell			
13						
14						
15						
16						
17						
18						
19						

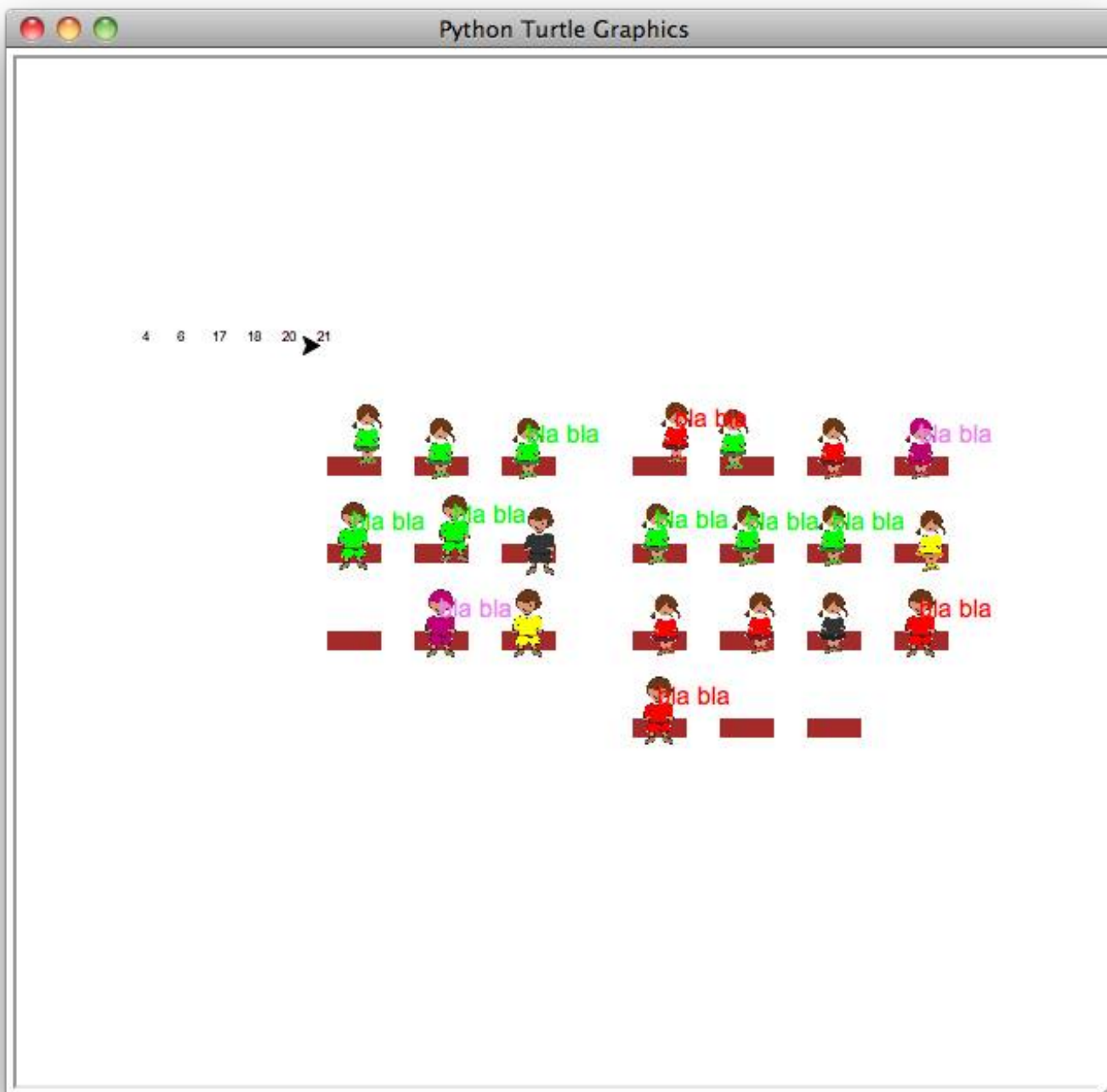
transitionPhase checkToObtainAttention

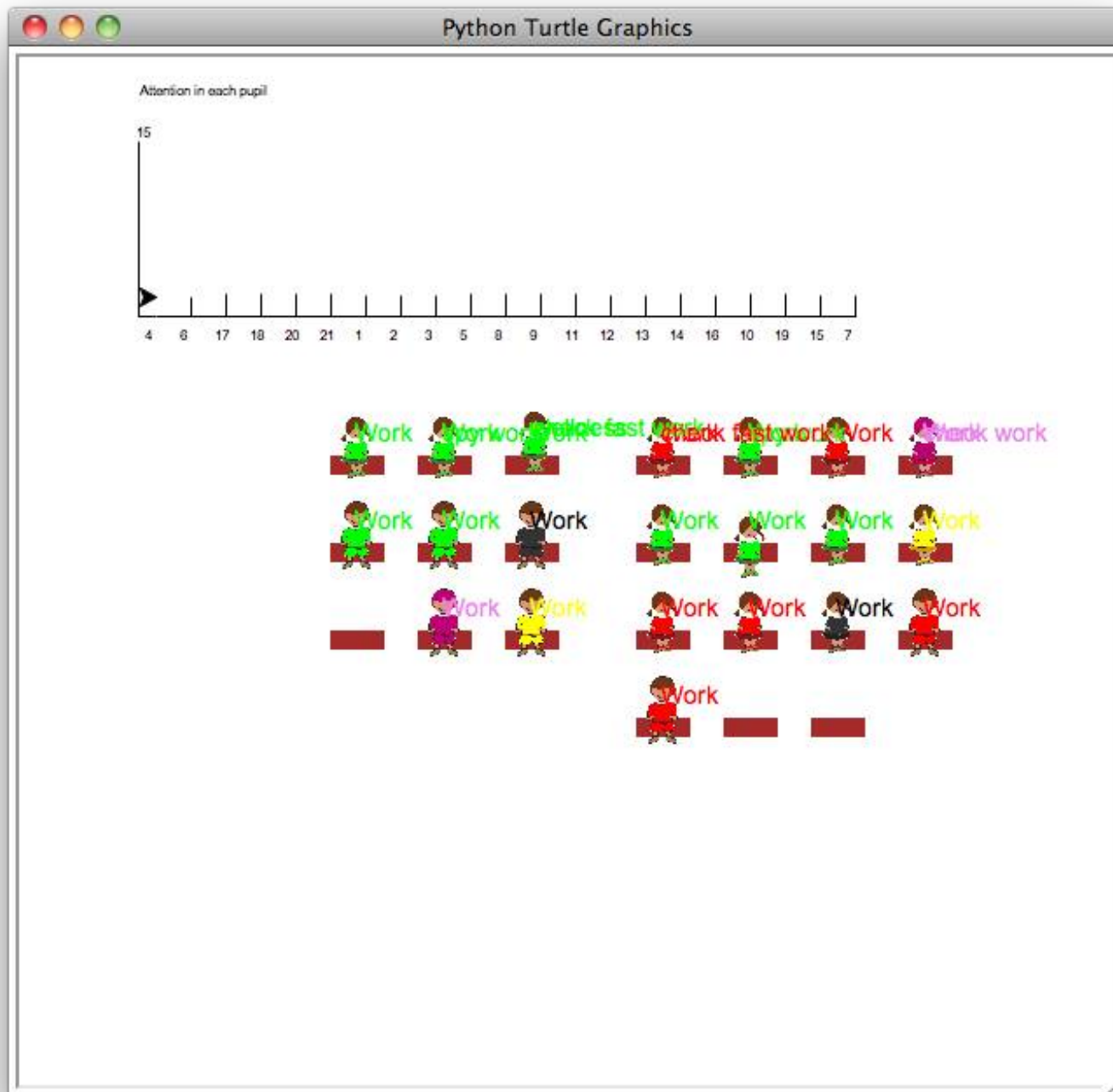
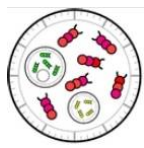
Foglio 3 / 3 PageStyle_checkToObtainAttention STD Somma=0 128%

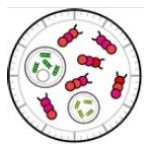


Graphic output



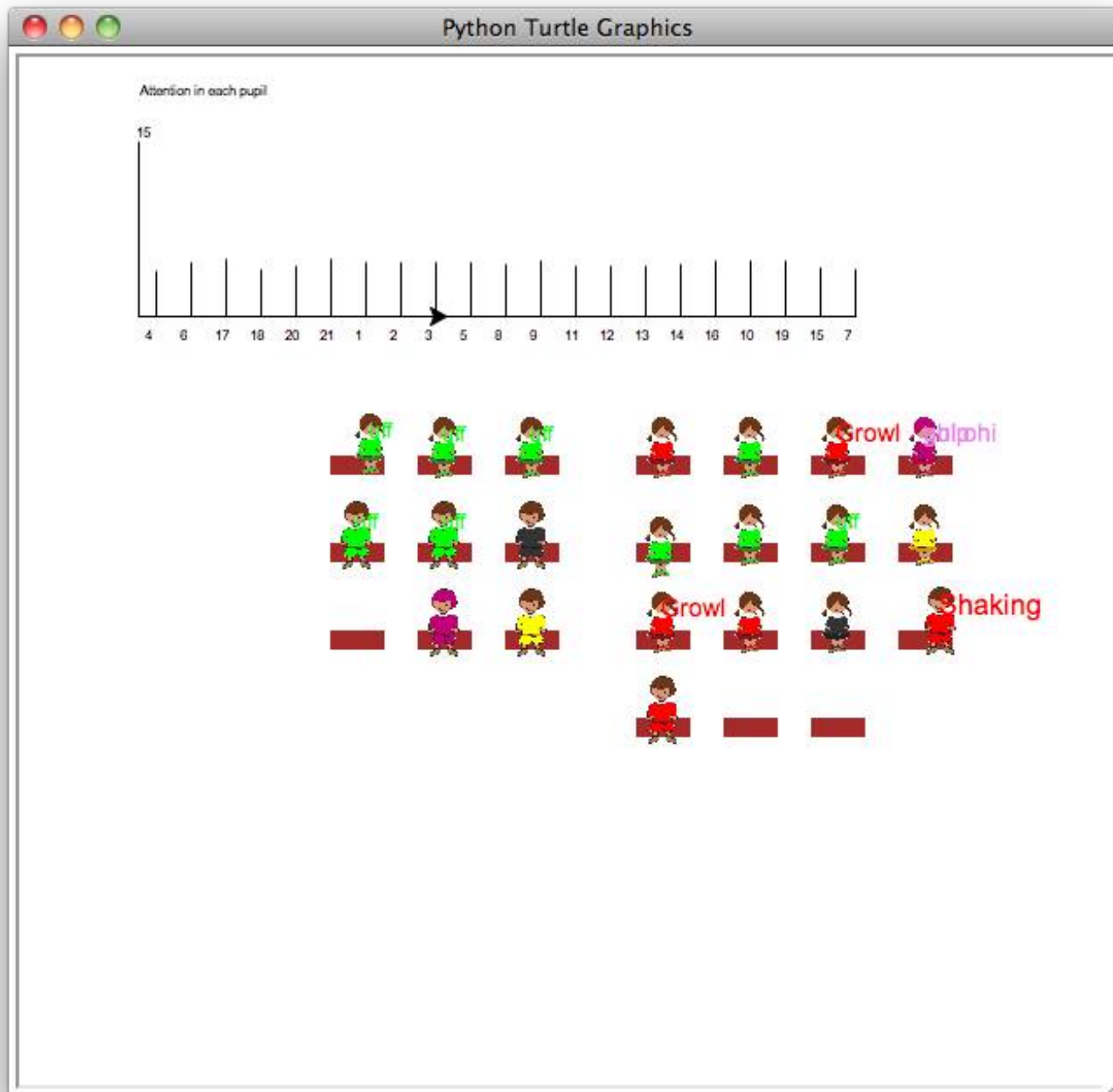
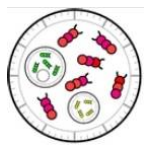


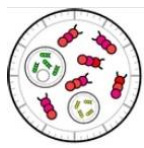




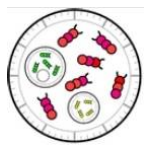
```
*Python Shell*
I'm gialloPupil agent 14: I'm answering chorus
I'm saPupil agent 15: I'm answering chorus
I'm verdePupil agent 8: I'm answering chorus
I'm rossoPupil agent 20: I'm answering chorus
I'm rossoPupil agent 18: I'm answering chorus
I'm verdePupil agent 12: I'm answering chorus
I'm verdePupil agent 13: I'm answering chorus
I'm rossoPupil agent 4: answering wrong
I'm scPupil agent 7: answering bad
I'm scPupil agent 7: I'm being praised
I'm verdePupil agent 8: I'm being praised
I'm rossoPupil agent 6: I'm fidgeting
I'm verdePupil agent 8: I'm fidgeting
I'm verdePupil agent 1: I'm fidgeting
I'm verdePupil agent 5: I'm fidgeting
I'm verdePupil agent 9: I'm fidgeting
I'm neroPupil agent 10: I'm fidgeting
I'm verdePupil agent 11: I'm fidgeting
I'm verdePupil agent 2: I'm fidgeting
I'm saPupil agent 15: I'm fidgeting
I'm scPupil agent 7: I'm fidgeting
I'm verdePupil agent 12: I'm fidgeting
I'm rossoPupil agent 20: I'm fidgeting
I'm rossoPupil agent 18: I'm fidgeting
I'm rossoPupil agent 4: I'm fidgeting
I'm verdePupil agent 3: I'm doing work
I'm verdePupil agent 13: I'm doing work
I'm rossoPupil agent 4: I'm doing work
I'm rossoPupil agent 21: I'm doing work
I'm gialloPupil agent 14: I'm doing work
I'm gialloPupil agent 16: I'm doing work
I'm verdePupil agent 1: I'm doing work
I'm verdePupil agent 8: I'm doing work
I'm scPupil agent 7: I'm doing work
I'm verdePupil agent 9: I'm doing work
I'm gialloPupil agent 14: I'm checking teacher and talking closely
I'm verdePupil agent 13: I'm checking teacher and talking closely
I'm verdePupil agent 12: I'm checking teacher and talking closely
I'm rossoPupil agent 4: I'm tidying
I'm verdePupil agent 13: I'm tidying
attention index for each pupil = [4,3.37564190111] |
```

Ln: 423 Col: 0



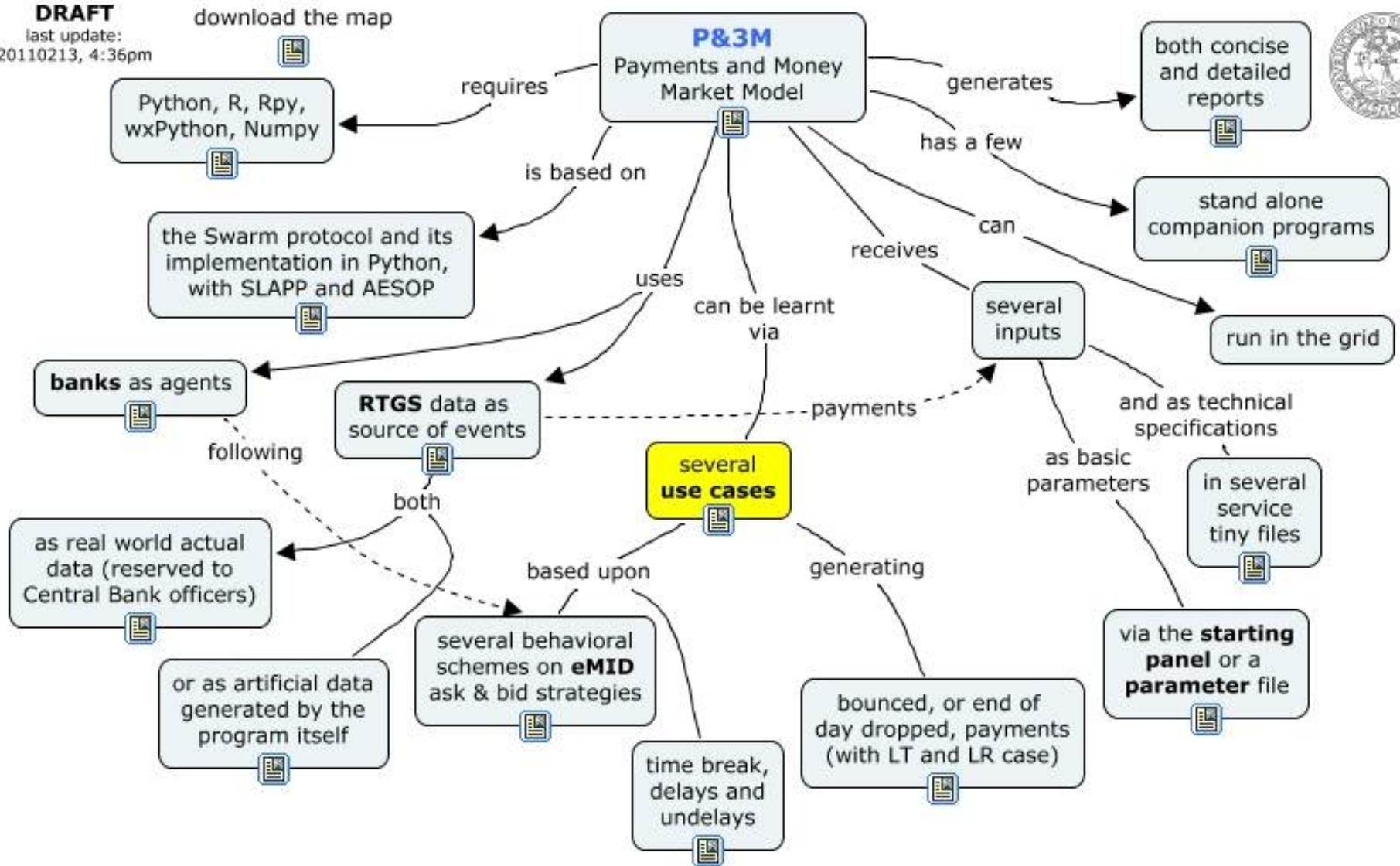


A second example, using large data sets and real time flows of data (Banks, Payments and Systemic Risk), with
Luca Arciero, Claudia Biancotti,
Leandro D'Aurizio, Giuseppe Ilardi,
Claudio Impenna, Cristina Picillo



The interbank payment model

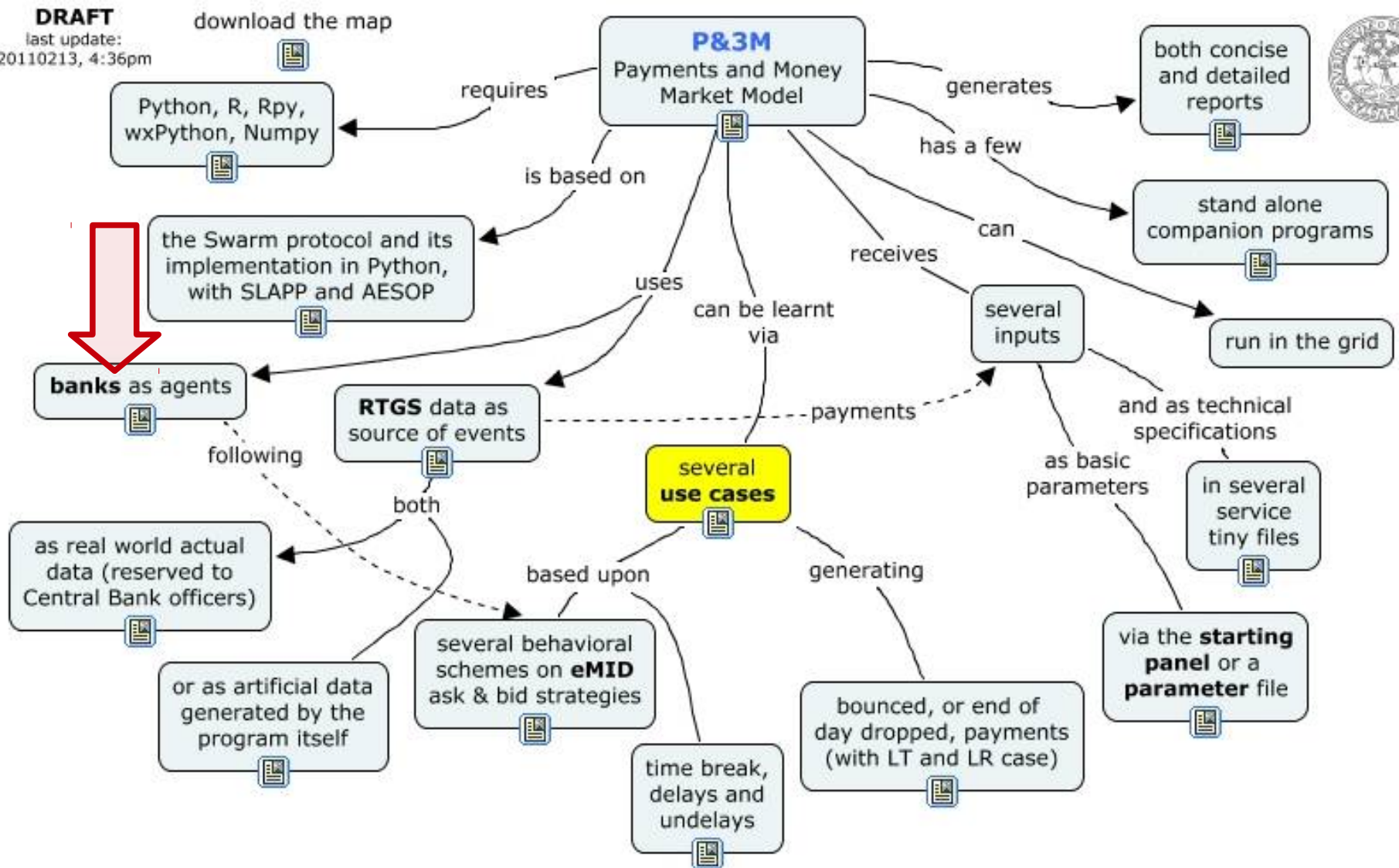
DRAFT
last update:
20110213, 4:36pm





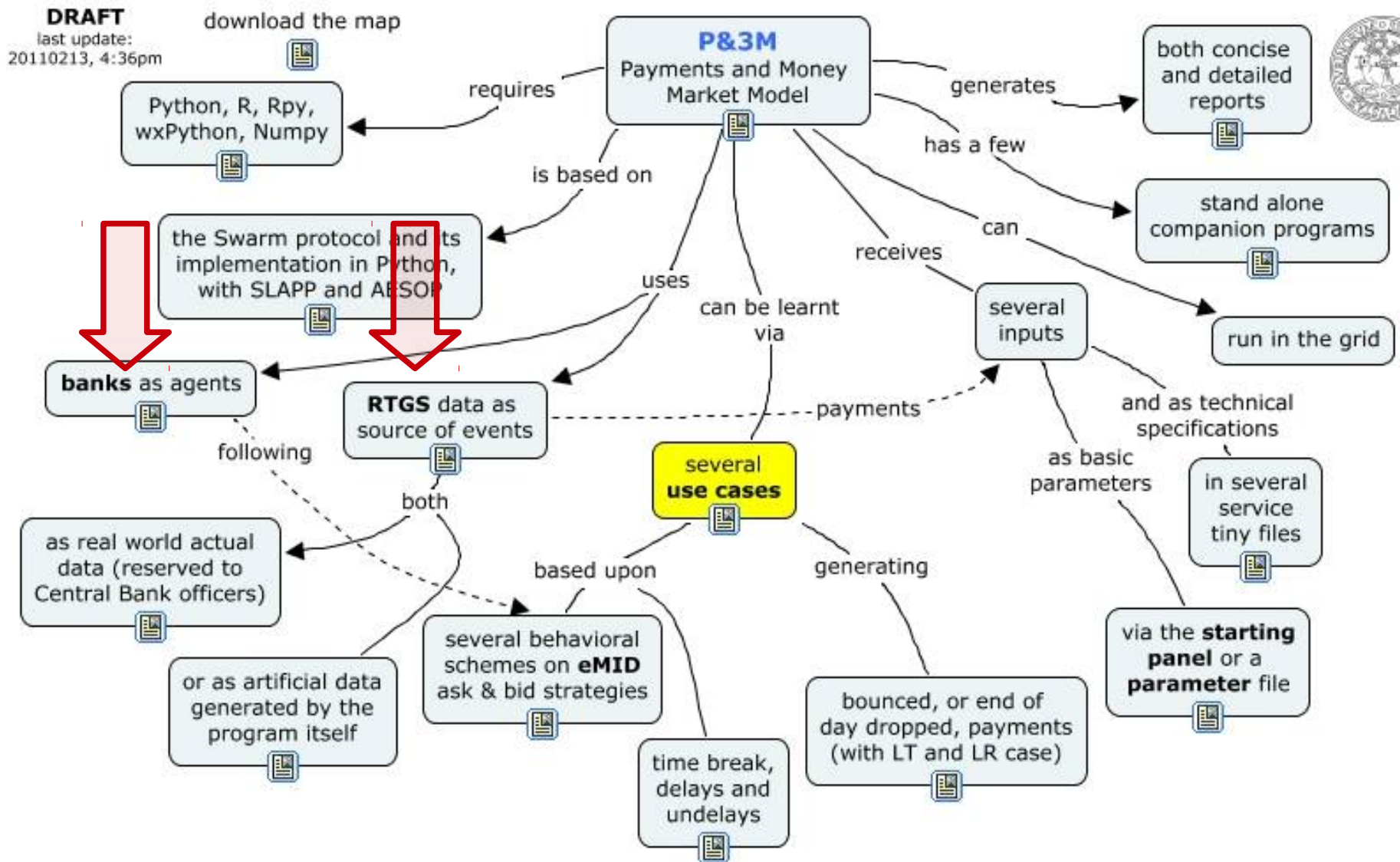
The interbank payment model

DRAFT
last update:
20110213, 4:36pm





The interbank payment model

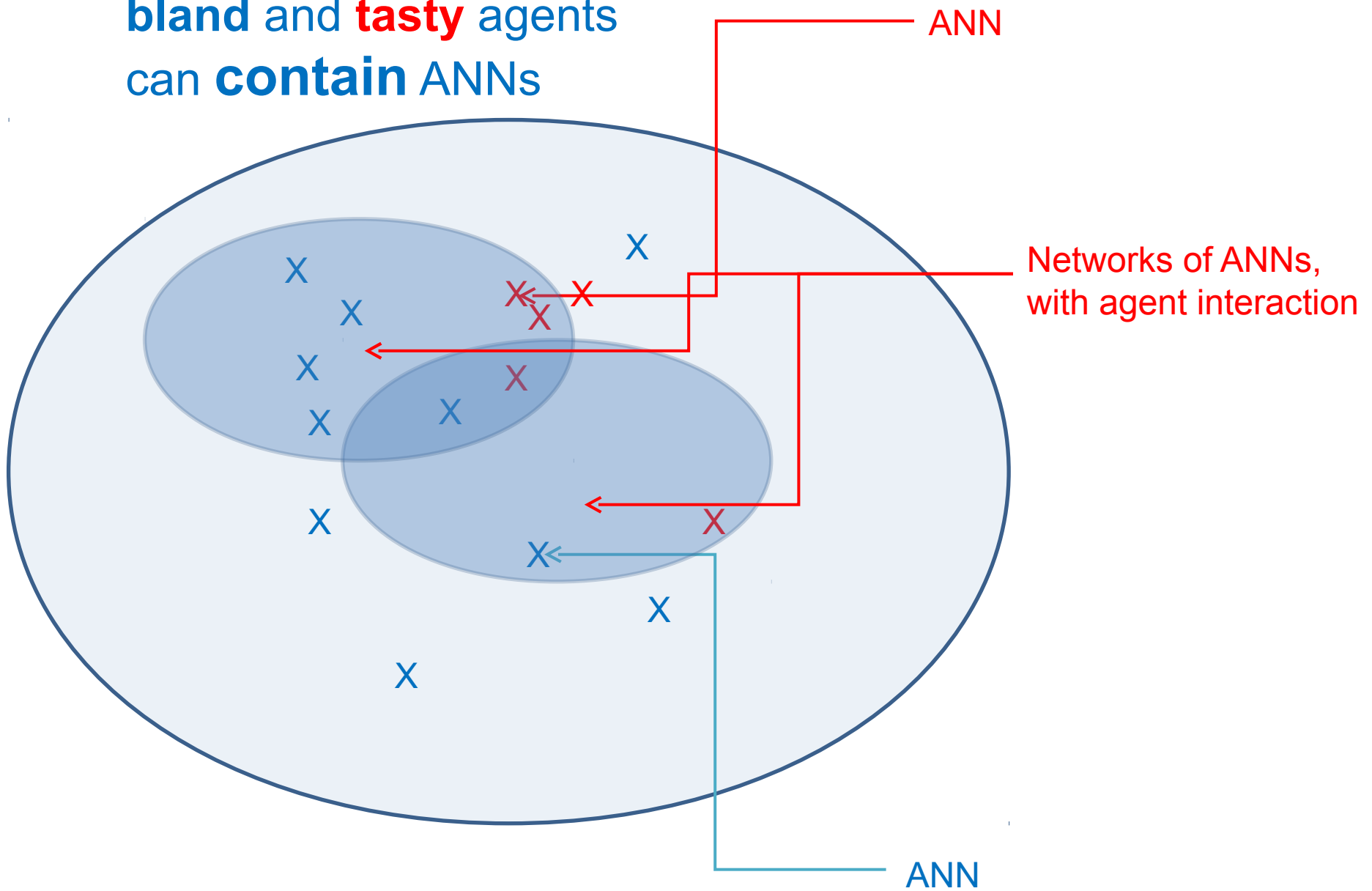




Artificial neural networks into the agents



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





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

(m)

(n)

actions

information

or

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

(1)

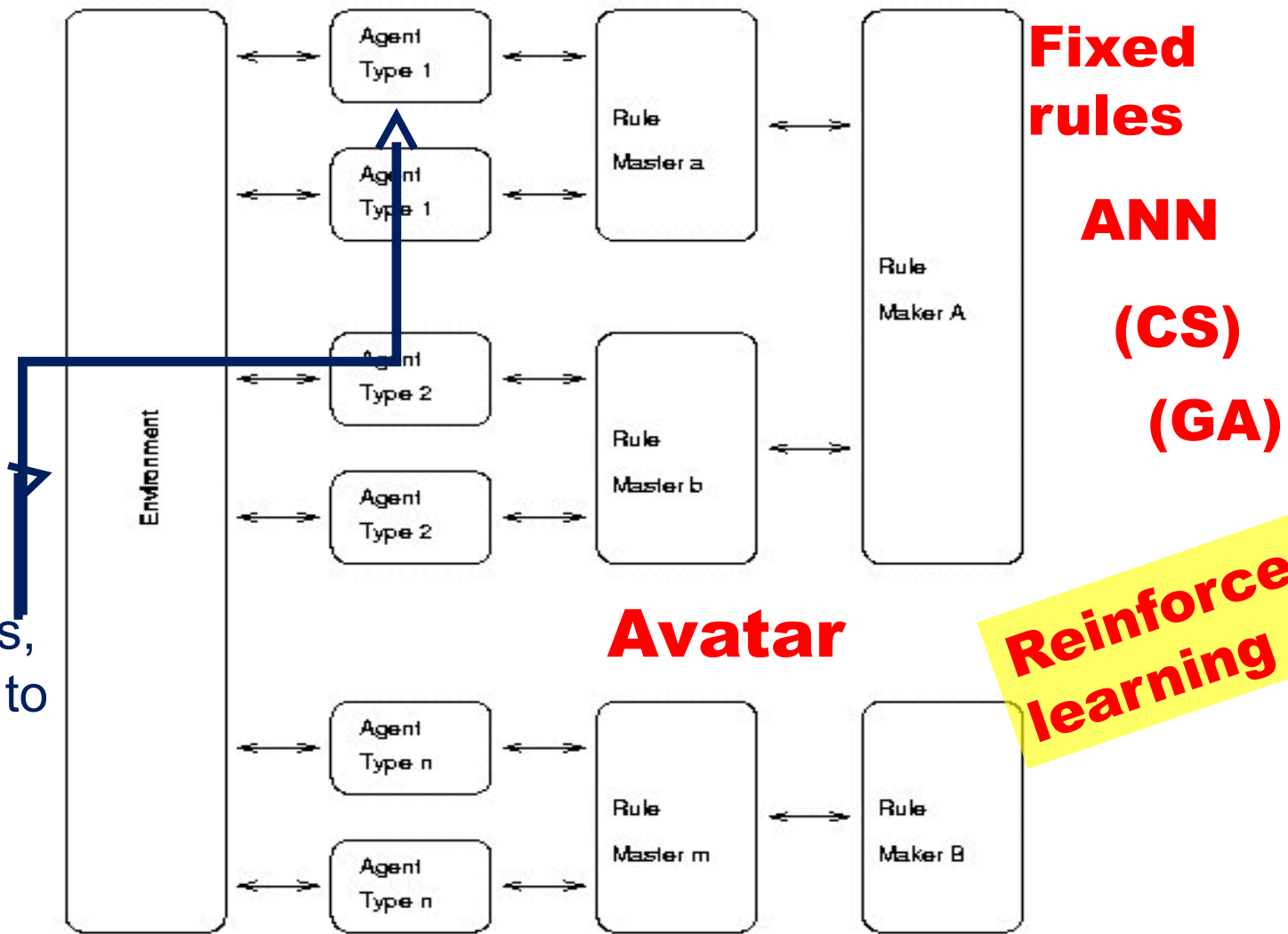
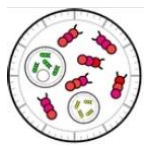
(n)

...

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

(1)

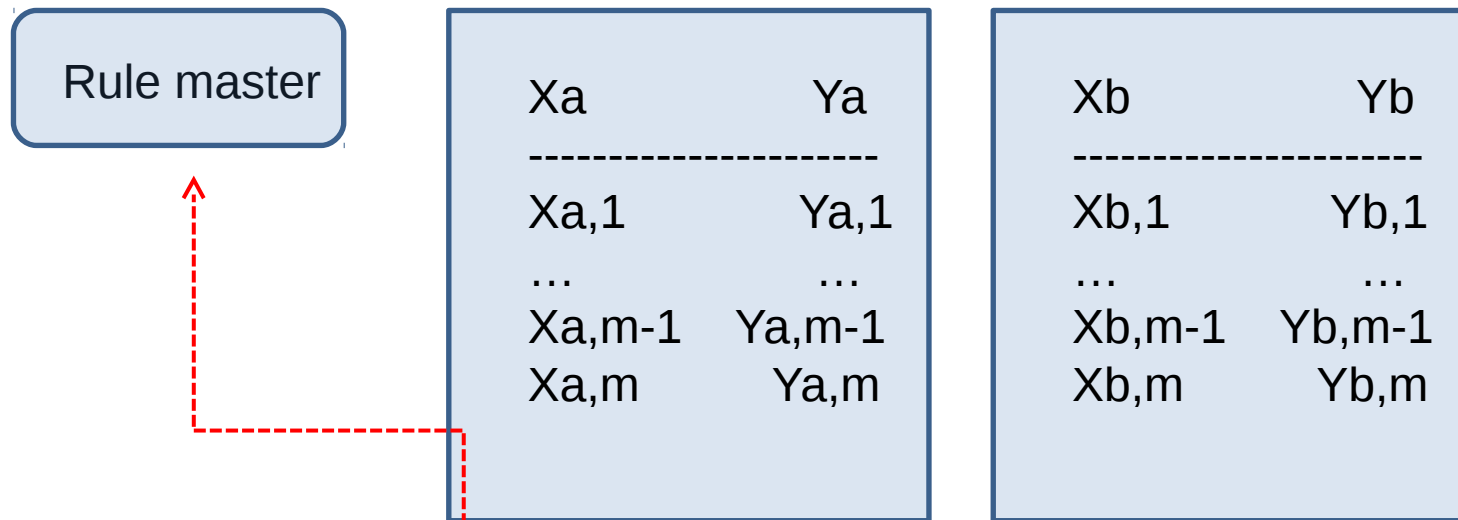
(n)



Microstructures,
mainly related to
time and
parallelism



a - Static ex-ante learning (on ex-ante data)

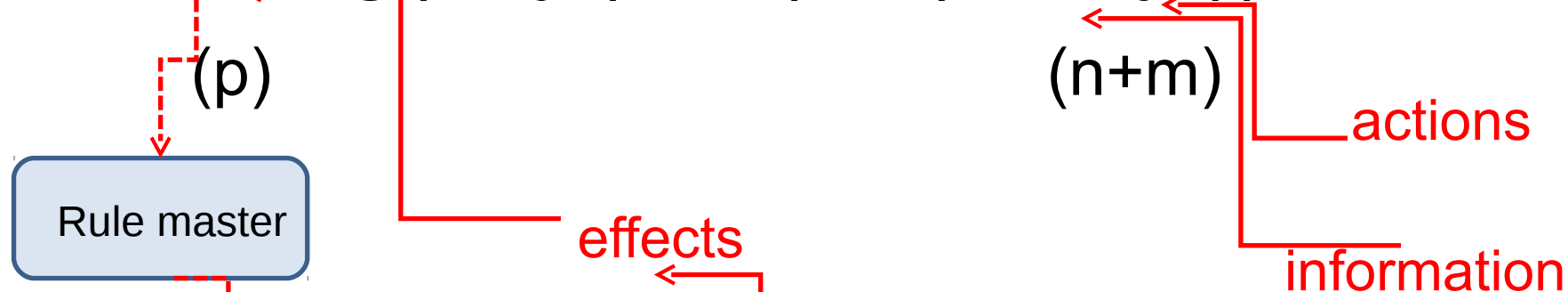


Different agents, with NN built on different set of data (i.e. data from real world or from trials and errors experiments), so with different matrixes A and B of parameters



b - Continuous learning (trials and errors learning **while** acting)

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



Different agent, generating and using different sets, A and B, of parameters (or using the same set of parameters, as collective learning)

Coming from the simulation

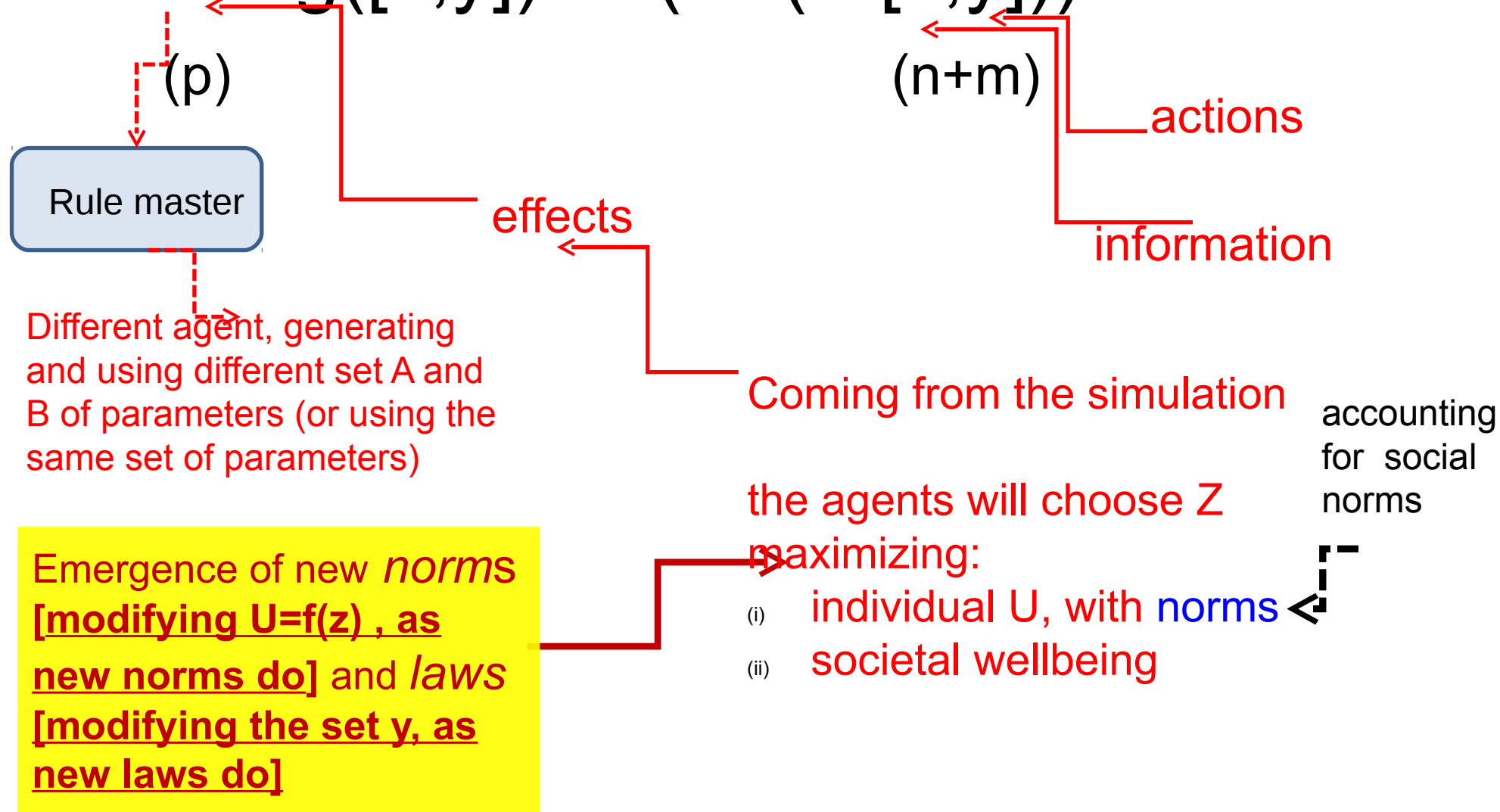
the agents will choose Z maximizing:

- (i) individual U, with norms
- (ii) societal wellbeing



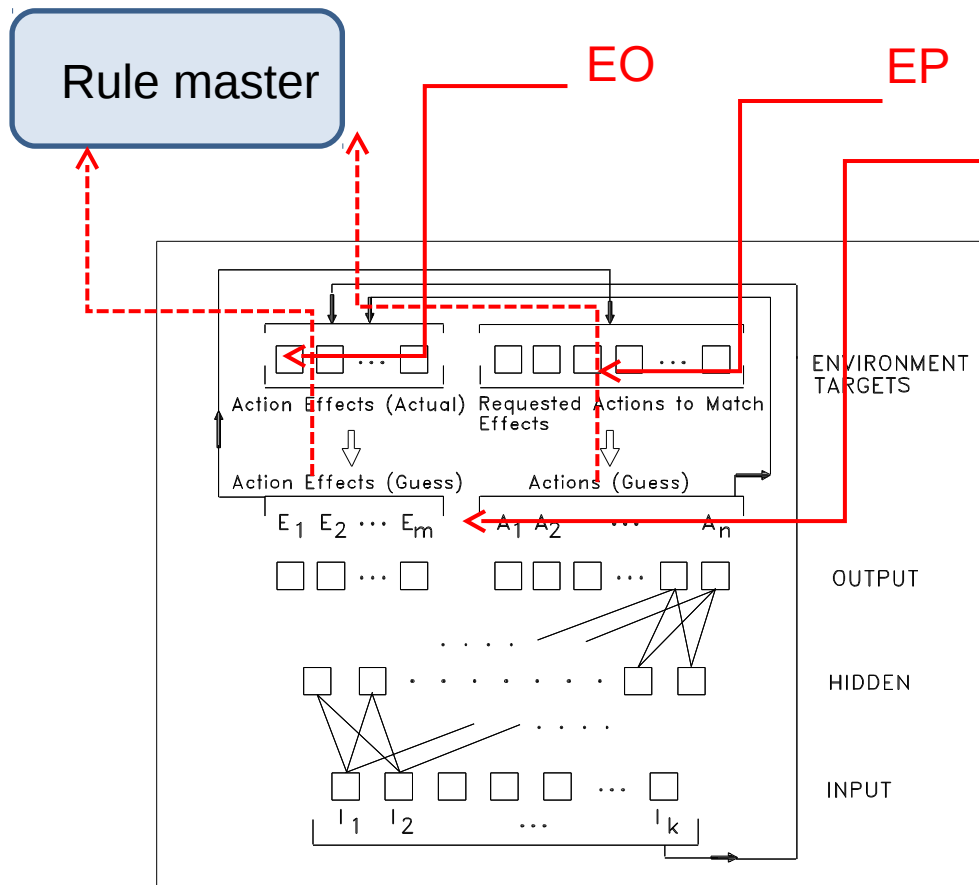
b - Continuous learning (trials and errors learning **while** acting)

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





c - Continuous learning (cross-targets)



Developing internal consistence between **ACTIONS** (right side of the outputs/targets) and the **EFFECTS** (left side of the outputs/targets)

Finally using **External Objectives (EO)** and **External Proposal (EP)**

A short presentation with comments at <http://web.econ.unito.it/terna/ct-era/ct-era.html>



Pietro Terna

Department of Economics and Public Finance, University of Torino, Italy

terna@econ.unito.it

web.econ.unito.it/terna

Thanks
