Collegio Carlo Alberto
UNIVERSITÀ DEGLI STUDI DI TORINO

SIMULTECH 2015

Agent-based Models for Exploring Social Complexity, with an Application of Network Analysis to Agents

SIMULTECH 2015, Colmar 21-23 July 2015

Pietro Terna
Collegio Carlo Alberto and University of Torino (retired professor)
web.econ.unito.it/terna, pietro.terna@unito.it

Accompanying material

Introductory:

•P. Terna (2013), A Complex Lens for Economics, or: About Ants and their Anthill, in “Spazio filosofico”, 7, pp. 167-177
http://www.spaziofilosofico.it/wp-content/uploads/2013/01/Terna-English.pdf

•P. Terna (2013), Learning agents and decisions: new perspectives, in "Law and Computational Social Science", 1,
http://eco83.econ.unito.it/terna/materiale/terna_def.pdf

Technical/Advanced:

•R. Boero, M. Morini, M. Sonnessa and P. Terna (2015), Agent-based Models of the Economy – From Theories to Applications, Palgrave Macmillan, Houndmills,
http://www.palgrave.com/page/detail/agentbased-models-of-the-economy-/?K=9781137339805

Basics

A note: the slides contain several references; you can find them in a draft paper, on line at **http://goo.gl/ryhyF**

Artifacts of social systems

Leibniz (*xi. De scientia universali seu calculo philosophico*): *… quando orientur controversiae, non magis disputatione opus erit inter duos philosophos, quam inter duos computistas. Sufficiet enim calamos in manus sumere sedereque ad abbacos et sibi mutuo (...) dicere, calculemus*

Calculemus or ... ***Simulemus***

... plus complexity, bounded rationality, chaos ...

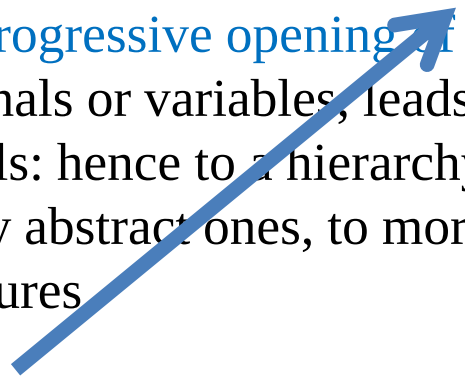
Rosenblueth and Wiener's 1945 paper, "The Role of Models in Science" , as a "manual" from the founders of cybernetics.

(p. 317) A distinction has already been made between material and formal or intellectual models. A material model is the representation of a complex system by a system which is assumed simpler and which is also assumed to have some properties similar to those selected for study in the original complex system. A formal model is a symbolic assertion in logical terms of an idealized relatively simple situation sharing the structural properties of the original factual system.
Material models are useful in the following cases. a) They may assist the scientist in replacing a phenomenon in an unfamiliar field by one in a field in which he is more at home.

(…) b) A material model may enable the carrying out of experiments under more favorable conditions than would be available in the original system.

Rosenblueth and Wiener's 1945 paper, "The Role of Models in Science" , as a "manual" from the founders of cybernetics.

(p. 319) It is obvious, therefore, that the difference between open-box and closed-box problems, although significant, is one of degree rather than of kind. All scientific problems begin as closed-box problems, i.e., only a few of the significant variables are recognized. Scientific progress consists in a progressive opening of those boxes. The successive addition of terminals or variables leads to gradually more elaborate theoretical models: hence to a hierarchy in these models, from relatively simple, highly abstract ones, to more complex, more concrete theoretical structures.

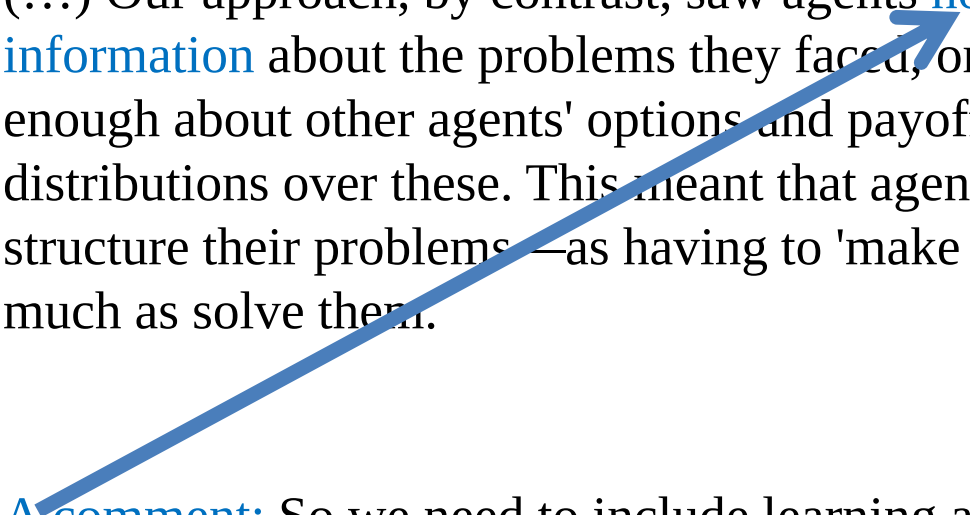


A comment: this is the main role of simulation models in the complexity perspective, **building material models as artifacts running into a computer**, having always in mind to go toward "more elaborate theoretical models".

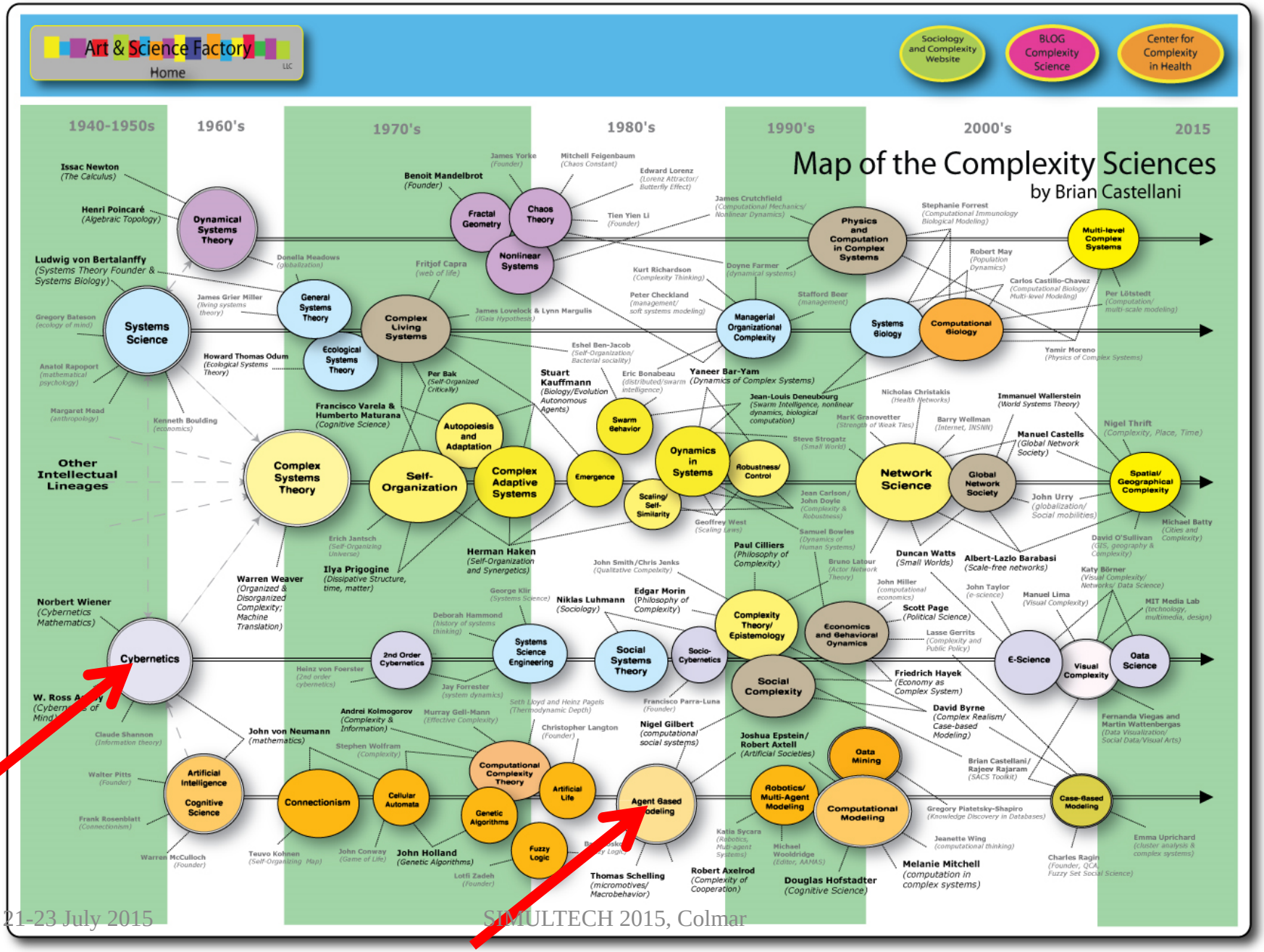
Finally, quoting another paper of the special issue referred above, that of prof.W. Brian Arthur

(…) a second theme that emerged was that of making models based on more realistic cognitive behavior. Neoclassical economic theory treats economic agents as perfectly rational optimizers. This means among other things that agents perfectly understand the choices they have, and perfectly assess the benefits they will receive from these.

(…) Our approach, by contrast, saw agents not as having perfect information about the problems they faced, or as generally knowing enough about other agents' options and payoffs to form probability distributions over these. This meant that agents need to cognitively structure their problems—as having to 'make sense' of their problems, as much as solve them.



A comment: So we need to include learning abilities into our agents.


Art & Science Factory
Home
Sociology and Complexity Website
BLOG Complexity Science
Center for Complexity in Health
1940-1950s
1960's
1970's
1980's
1990's
2000's
2015
Map of the Complexity Sciences
by Brian Castellani
Issac Newton (The Calculus)
Henri Poincaré (Algebraic Topology)
Dynamical Systems Theory
Benoit Mandelbrot (Founder)
Fractal Geometry
Chaos Theory
Nonlinear Systems
James Yorke (Founder)
Mitchell Feigenbaum (Chaos Constant)
Edward Lorenz (Lorenz Attractor/ Butterfly Effect)
Tien Yien Li (Founder)
James Crutchfield (Computational Mechanics/ Nonlinear Dynamics)
Physics and Computation in Complex Systems
Stephanie Forrest (Computational Immunology Biological Modeling)
Multi-level Complex Systems
Robert May (Population Dynamics)
Carlos Castillo-Chavez (Computational Biology/ Multi-level Modeling)
Per Lötstedt (Computation/ multi-scale modeling)
Yamir Moreno (Physics of Complex Systems)
Ludwig von Bertalanffy (Systems Theory Founder & Systems Biology)
Systems Science
General Systems Theory
Donella Meadows (globalization)
Fritjof Capra (web of life)
Complex Living Systems
James Grier Miller (living systems theory)
Ecological Systems Theory
Howard Thomas Odum (Ecological Systems Theory)
James Lovelock & Lynn Margulis (IGaia Hypothesis)
Kurt Richardson (Complexity Thinking)
Peter Checkland (management/ soft systems modeling)
Doyne Farmer (dynamical systems)
Managerial Organizational Complexity
Stafford Beer (management)
Systems Biology
Computational Biology
Gregory Bateson (ecology of mind)
Anatol Rapoport (mathematical psychology)
Margaret Mead (anthropology)
Kenneth Boulding (economics)
Francisco Varela & Humberto Maturana (Cognitive Science)
Per Bak (Self-Organized Criticality)
Autopoiesis and Adaptation
Eshel Ben-Jacob (Self-Organization/ Bacterial sociality)
Stuart Kauffmann (Biology/Evolution Autonomous Agents)
Eric Bonabeau (distributed/swarm intelligence)
Yaneer Bar-Yam (Dynamics of Complex Systems)
Swarm Behavior
Jean-Louis Deneubourg (Swarm Intelligence, nonlinear dynamics, biological computation)
Nicholas Christakis (Health Networks)
Immanuel Wallerstein (World Systems Theory)
MarK Granovetter (Strength of Weak Ties)
Barry Wellman (Internet, INSNW)
Steve Strogatz (Small World)
Manuel Castells (Global Network Society)
Nigel Thrift (Complexity, Place, Time)
Other Intellectual Lineages
Complex Systems Theory
Self-Organization
Complex Adaptive Systems
Emergence
Dynamics in Systems
Robustness/ Control
Scaling/ Self-Similarity
Network Science
Global Network Society
Spatial/ Geographical Complexity
Jean Carlson/ John Doyle (Complexity & Robustness)
Geoffrey West (Scaling Laws)
Samuel Bowles (Dynamics of Human Systems)
John Urry (globalization/ Social mobilities)
Michael Batty (Cities and Complexity)
David O'Sullivan (GIS, geography & Complexity)
Erich Jantsch (Self-Organizing Universe)
Ilya Prigogine (Dissipative Structure, time, matter)
Herman Haken (Self-Organization and Synergetics)
Warren Weaver (Organized & Disorganized Complexity; Machine Translation)
John Smith/Chris Jenks (Qualitative Compexity)
Paul Cilliers (Philosophy of Complexity)
Bruno Latour (Actor Network Theory)
Duncan Watts (Small Worlds)
Albert-Lazlo Barabasi (Scale-free networks)
Katy Börner (Visual Complexity/ Networks/ Data Science)
John Miller (computational economics)
John Taylor (e-science)
Manuel Lima (Visual Complexity)
MIT Media Lab (technology, multimedia, design)
Norbert Wiener (Cybernetics Mathematics)
Cybernetics
George Klir (Systems Science)
Niklas Luhmann (Sociology)
Edgar Morin (Philosophy of Complexity)
Complexity Theory/ Epistemology
Economics and Behavioral Dynamics
Scott Page (Political Science)
Lasse Gerrits (Complexity and Public Policy)
E-Science
Visual Complexity
Data Science
Deborah Hammond (history of systems thinking)
2nd Order Cybernetics
Systems Science Engineering
Social Systems Theory
Socio-Cybernetics
Social Complexity
Friedrich Hayek (Economy as Complex System)
Heinz von Foerster (2nd order cybernetics)
Jay Forrester (system dynamics)
Seth Lloyd and Heinz Pagels (Thermodynamic Depth)
Francisco Parra-Luna (Founder)
David Byrne (Complex Realism/ Case-based Modeling)
Fernanda Viegas and Martin Wattenbergas (Data Visualization/ Social Data/Visual Arts)
Claude Shannon (Information theory)
John von Neumann (mathematics)
Andrei Kolmogorov (Complexity & Information)
Murray Gell-Mann (Effective Complexity)
Stephen Wolfram (Complexity)
Christopher Langton (Founder)
Nigel Gilbert (computational social systems)
Joshua Epstein/ Robert Axtell (Artificial Societies)
Data Mining
Brian Castellani/ Rajeev Rajaram (SACS Toolkit)
Walter Pitts (Founder)
Artificial Intelligence
Cognitive Science
Connectionism
Cellular Automata
Computational Complexity Theory
Artificial Life
Agent Based Modeling
Robotics/ Multi-Agent Modeling
Computational Modeling
Case-based Modeling
Frank Rosenblatt (Connectionism)
Genetic Algorithms
Fuzzy Logic
Gregory Piatetsky-Shapiro (Knowledge Discovery in Databases)
Jeanette Wing (computational thinking)
Emma Uprichard (cluster analysis & complex systems)
Warren McCulloch (Founder)
Teuvo Kohnen (Self-Organizing Map)
John Conway (Game of Life)
John Holland (Genetic Algorithms)
Lotfi Zadeh (Founder)
Thomas Schelling (micromotives/ Macrobehavior)
Robert Axelrod (Complexity of Cooperation)
Katia Sycara (Robotics, Multi-agent Systems)
Michael Wooldridge (Editor, AAMAS)
Douglas Hofstadter (Cognitive Science)
Melanie Mitchell (computation in complex systems)
Charles Ragin (Founder, QCA Fuzzy Set Social Science)

Moving to models

We can now move to models:
in the traditional way

or in the *new* perspective of

the material models of cybernetics founders

the computational artifacts of the agent-based simulation models.

quite close

Following Ostrom (1988), and to some extent, Gilbert and Terna (2000), in social science, excluding material (analogue) models, we traditionally build models as simplified representations of reality, using:

i. Verbal Argumentation and
ii. Mathematical Equations, typically with Statistics and Econometrics

Now we have computational tools:

- Equilibrium Models
- Game Theory
- System Dynamics

} close to ii.

- Serious Gaming
- Agent-Based Simulation

} iii.

Computer simulation (mainly agent-based one) can combine the extreme flexibility of a computer code – where we can create agents who act, make choices, and react to the choices of other agents and to modifications of their environment – and its intrinsic computability.

Economics after the crisis: New model army | The Economist

New model army

Economics after the crisis

Efforts are under way to improve macroeconomic models

Jan 19th 2013 | WASHINGTON, DC | From the print edition

The Economist, Jan 19th 2013

http://www.economist.com/node/21569752

Improving DSGE models is the obvious way to take the lessons of the crisis on board. But others exist too. "Agent-based modelling" tries to depict the transactions that might occur in an actual economy. These models are populated by millions of agents that gradually alter the economy as they interact with each other.

A common objection: reality is intrinsically agent-based. So, why reproduce social structures in an agent-based way, following (iii), when science applies (ii) to describe, explain, and forecast reality, which is, per se, too complicated to be understood? At a first glance, this is a strong criticism.

The reply is that we can, with agent-based models and simulation, produce **artifacts** (the 'material model') of actual systems and "play" with them, i.e., **showing the consequences of perfectly known ex-ante hypotheses and agent behavioral design and interaction**; then we can apply statistics and econometrics to the outcomes of the simulation and compare the results with those obtained by applying the same tests to actual data.

Quoting Hayek (1979), Law Legislation and Liberty, vol.3, Epilogue, p. 156: *Though the conception of conjectural history is somewhat suspect today, when we cannot say precisely how things did happen, to understand how they could have come about may be an important insight*.(*)

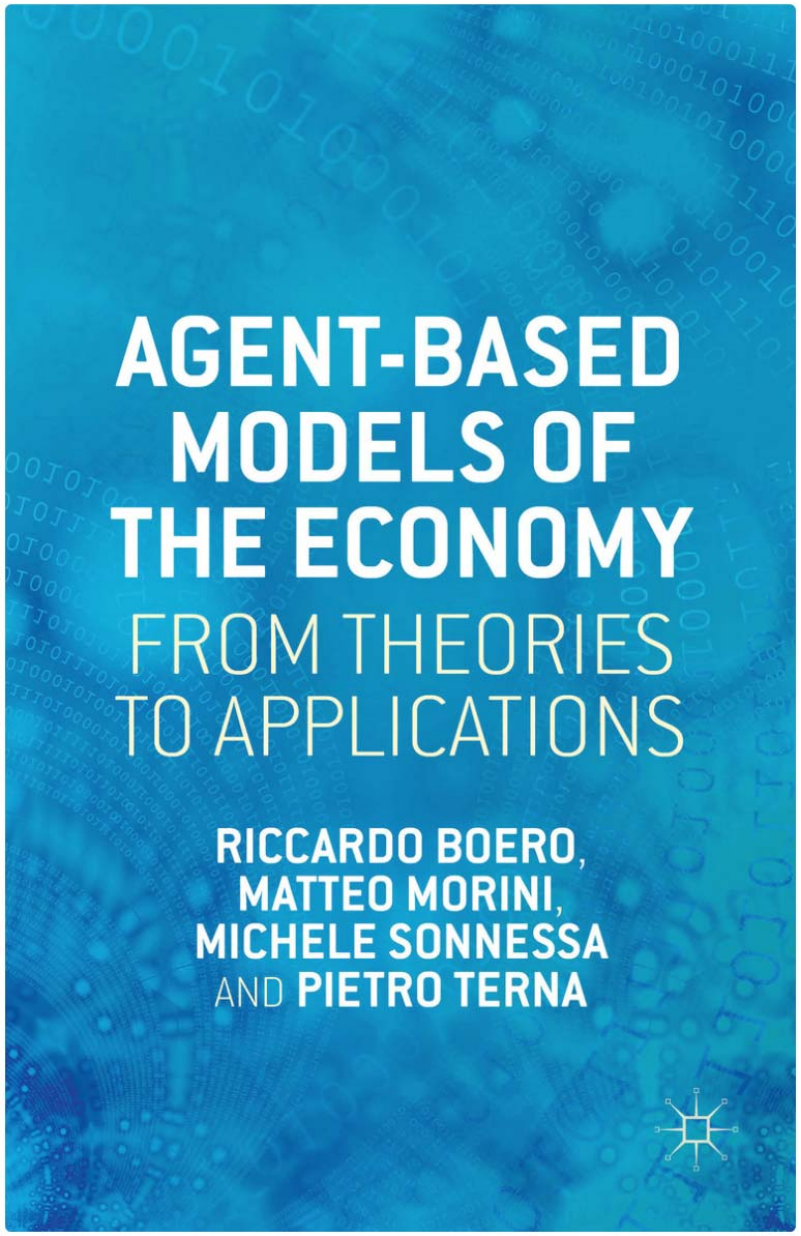
(*) I owe this quotation to a friend of mine, prof.Jack Birner.

Agent-based simulation models have severe weaknesses, primarily arising from:

- The difficulty of fully understanding them without studying the program used to run the simulation;
- The necessity of carefully checking computer code to prevent generation of inaccurate results from mere coding errors;
- The difficulty of systematically exploring the entire set of possible hypotheses in order to infer the best explanation. This is mainly due to the inclusion of behavioral rules for the agents within the hypotheses, which produces a space of possibilities that is difficult if not impossible to explore completely.

A few replies:

- Swarm (http://www.swarm.org) a project that started within the Santa Fe Institute (first release 1994) and that represents a milestone in simulation;
- Swarm has been highly successful, being its protocol intrinsically the basis of several recent tools; for an application of the Swarm protocol in Python, see my SLAPP, Swarm Like Agent Protocol in Python at https://github.com/terna/SLAPP/
- Many other tools have been built upon the Swarm legacy, such as Repast, Ascape, Mason, JAS and also by simpler, but important tools, such as NetLogo and StarLogoTNG.

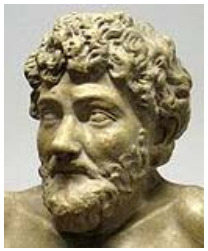


http://www.palgrave.com/page/detail/agentbased-models-of-the-economy-/?K=9781137339805

Technicalities:
Why Swarm, Python SLAPP and why NetLogo?

SLAPP, or Swarm-Like Agent Protocol in Python, is a simplified implementation of the original Swarm protocol (http://www.swarm.org), choosing Python as a simultaneously simple and complete object-oriented framework.

SLAPP contains 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.

SLAPP is also useful:

- 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 leverage the openness of Python
- to apply easily the SWARM protocol

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.santafe.edu/media/workingpapers/96-06-042.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 create 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**.*

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

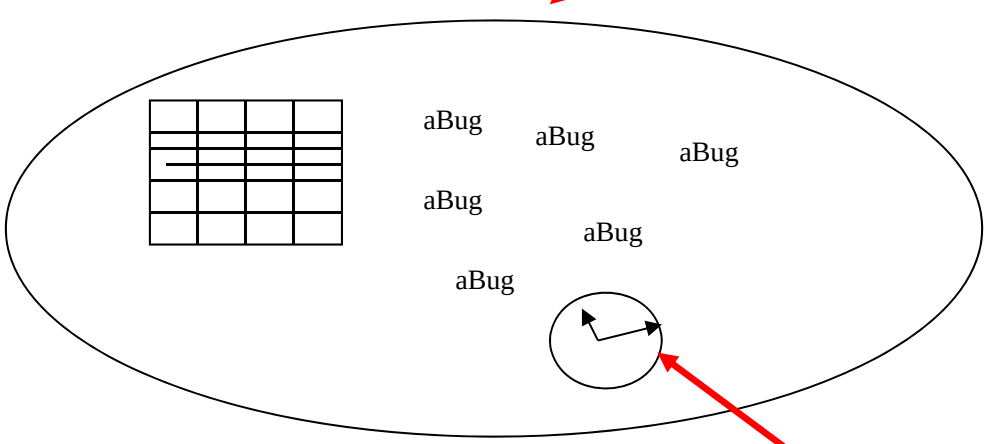
modelSwarm

Bug
aBug
bugList

create agents
create actions

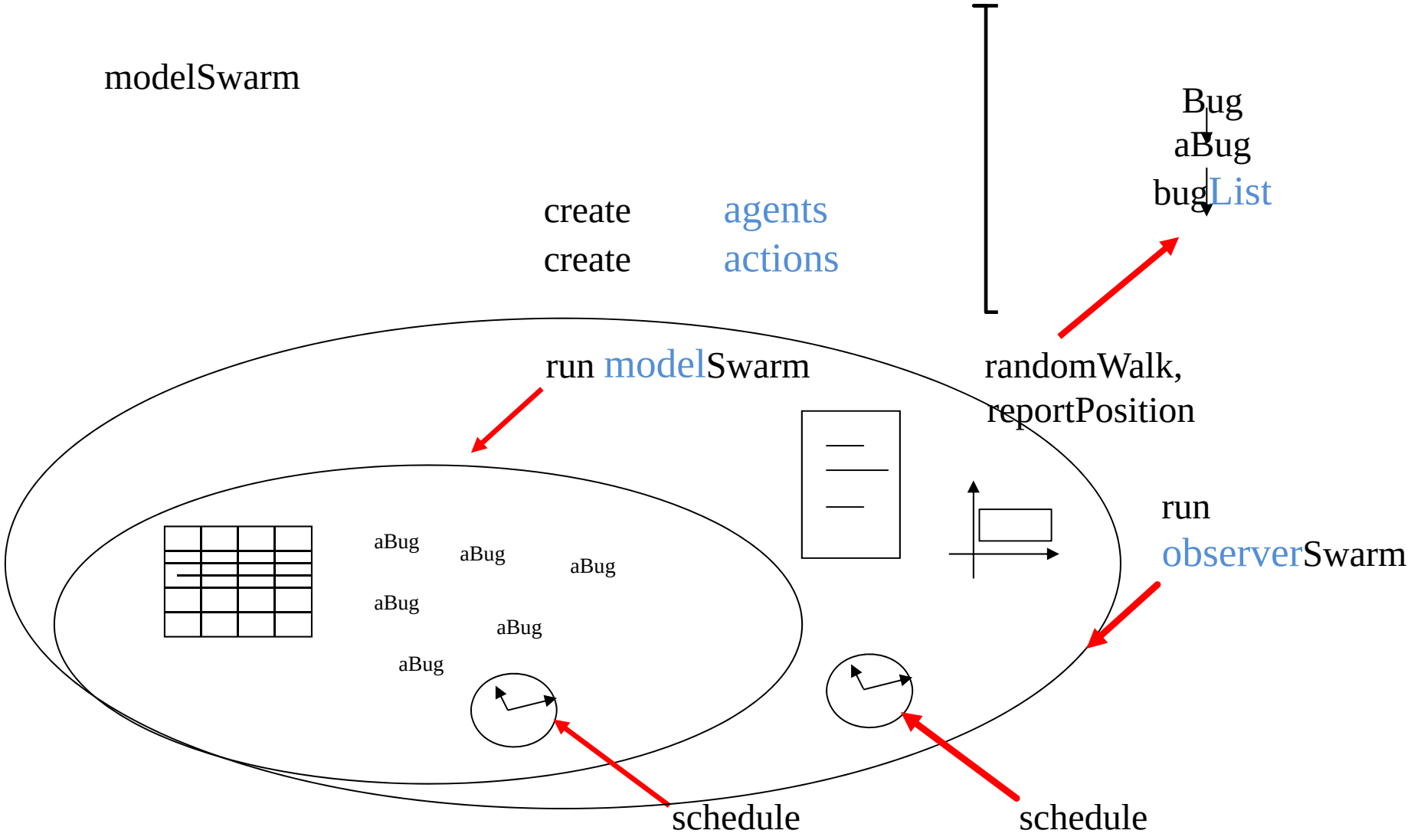
run modelSwarm

randomWalk,
reportPosition

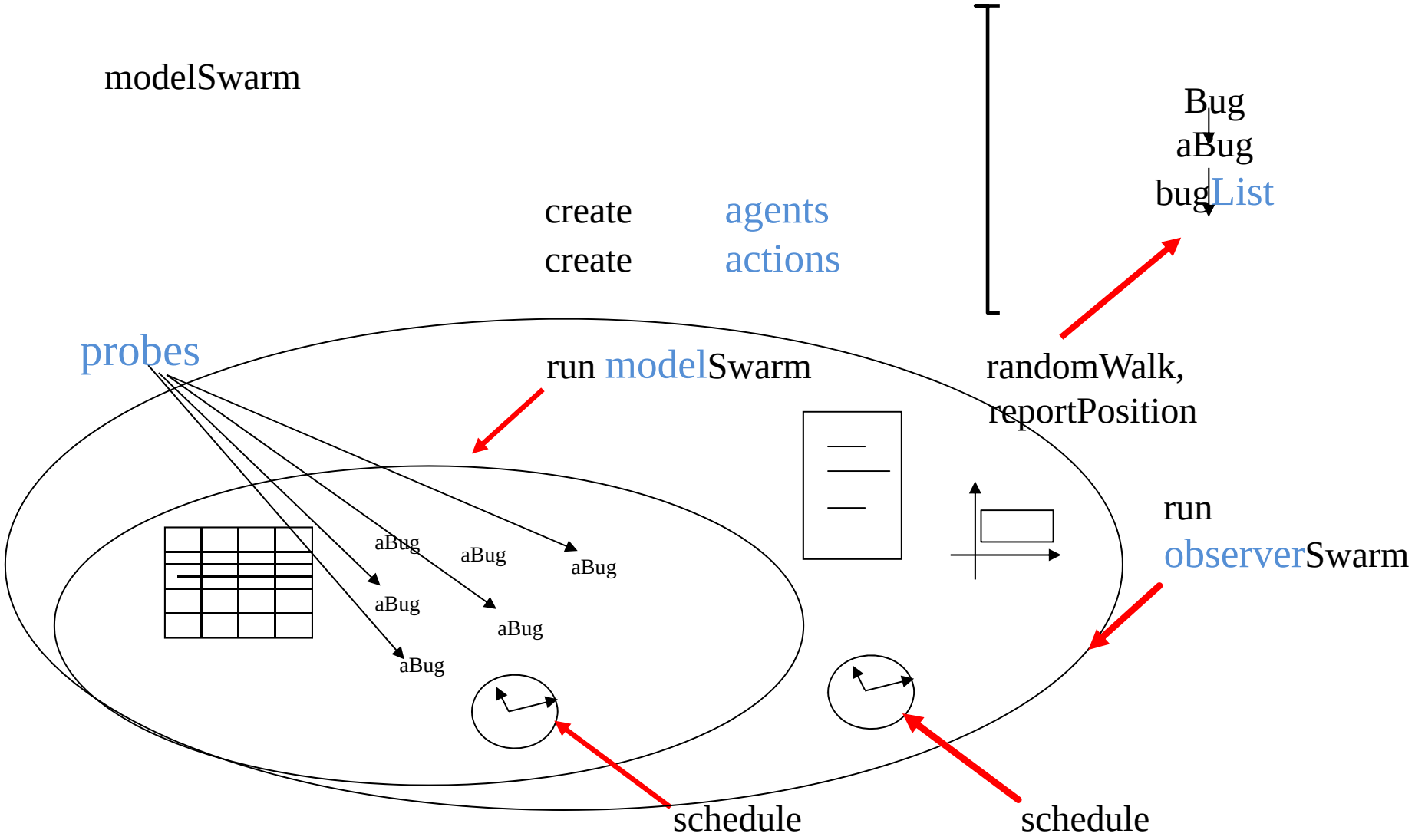


schedule

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



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



Always in technicalities … why NetLogo?

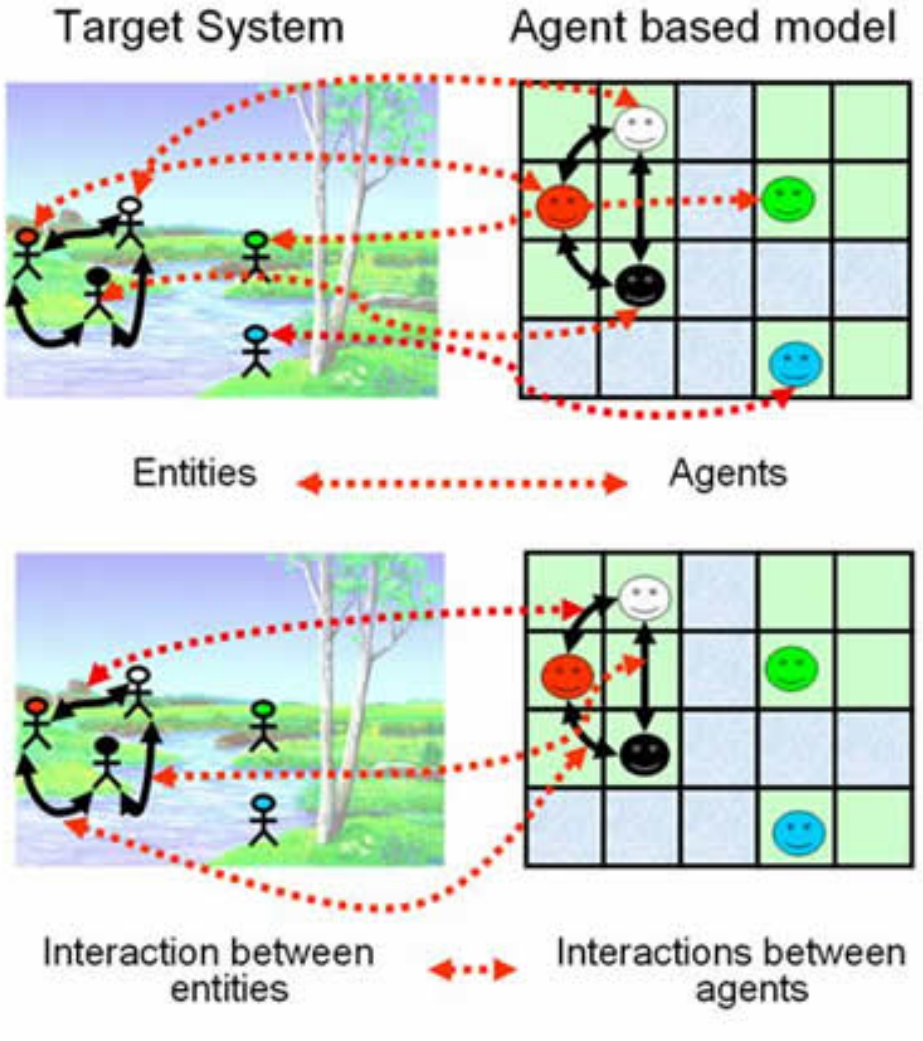
NetLogo is highly diffusing as a rigorous and easy tool, especially useful for prototyping and when we need advanced graphical capabilities

Limits are in coping with the design of complex experiments (and with huge numbers of agents)

Moving to computation

From Bruce Edmonds, Agent-based social simulation

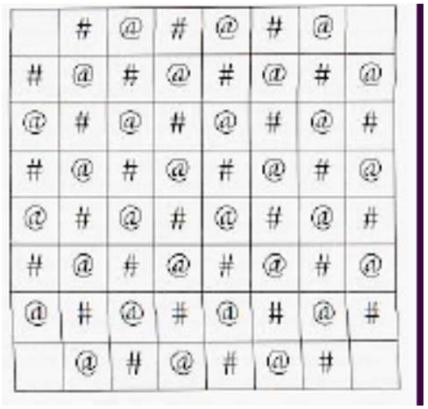
www.methods.manchester.ac.uk/methods/abss/

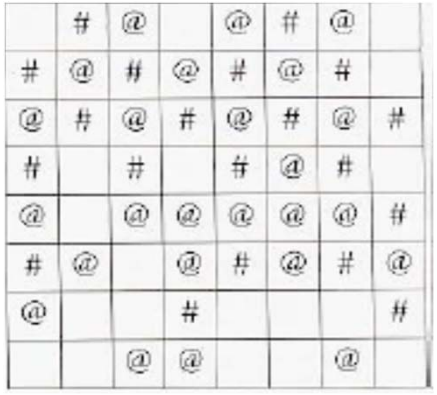


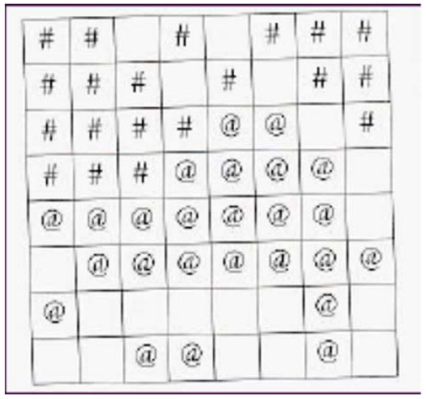
Finally, the importance of calculating: our complex system models live mainly in their computational phase and require calculating facilities more and more powerful.

Schelling model and random mutations

The well known segregation model from prof.Schelling has been initially solved moving dimes and pennies on a board.

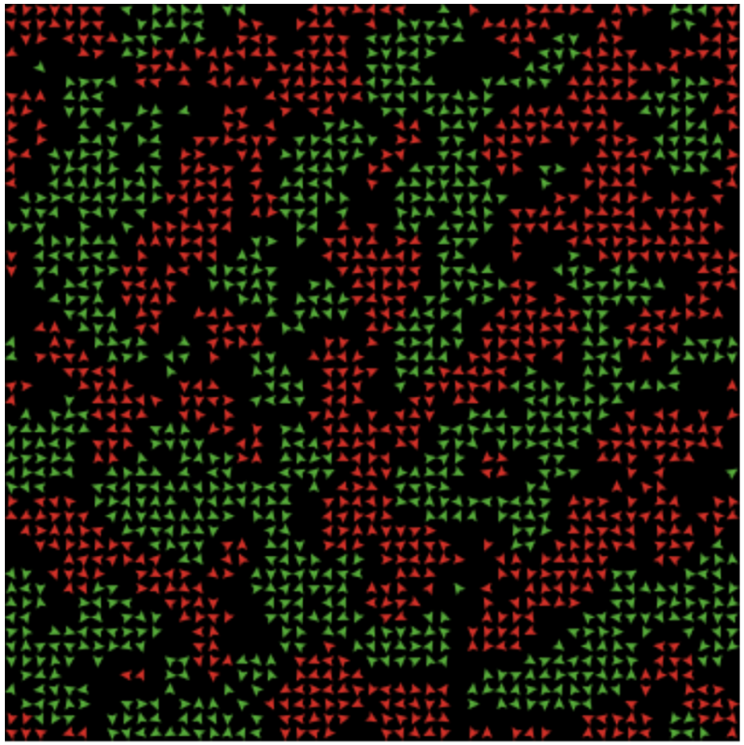


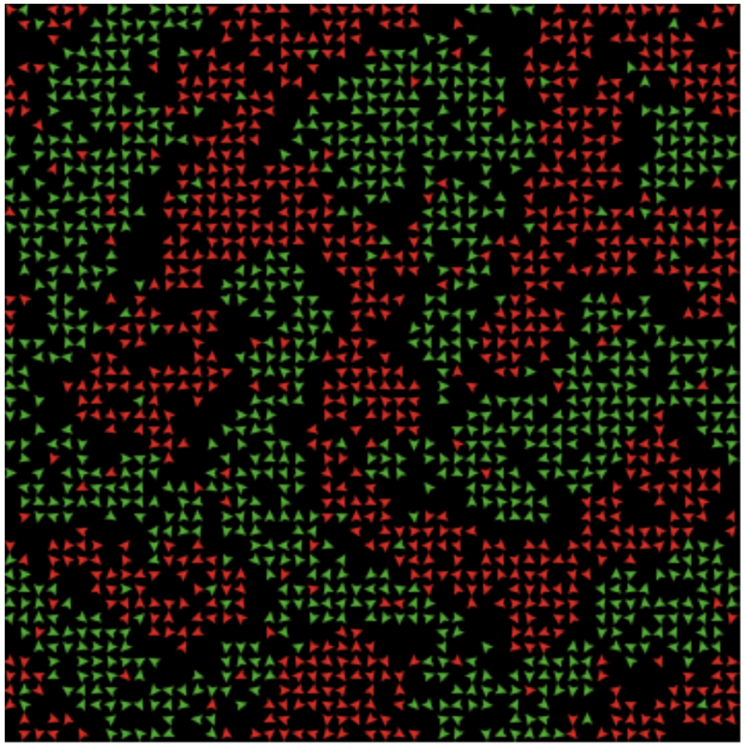




These pictures are from a presentation of Eileen Kraemer, http://www.cs.uga.edu/~eileen/fres1010/Notes/fres1010L4v2.ppt

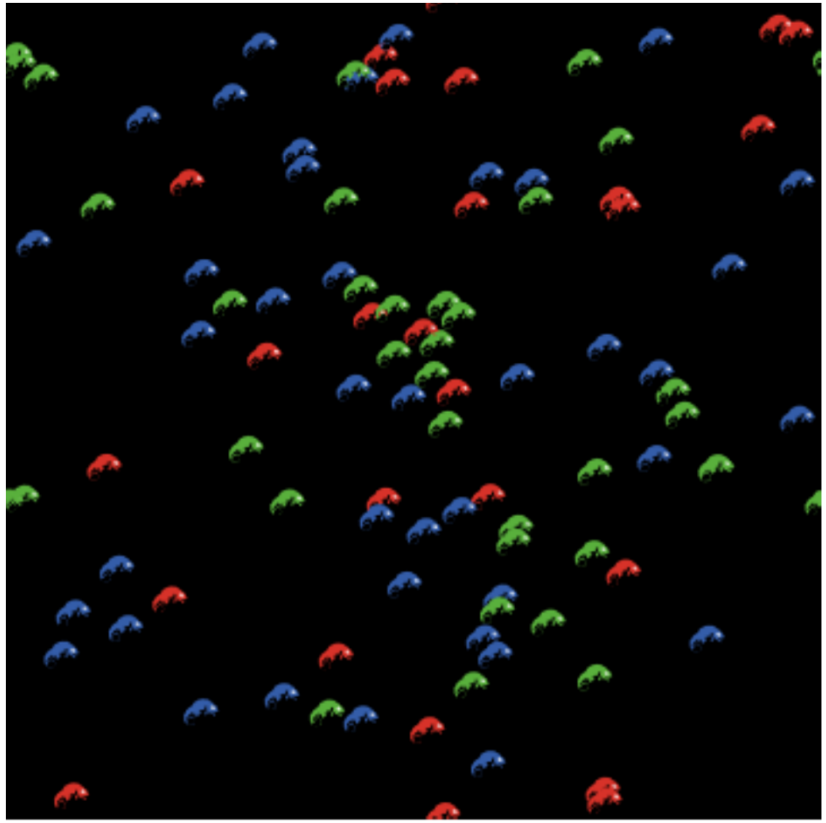
However, if you want to check the survival of the color islands in the presence of random mutations in agents (from an idea of prof.Nigel Gilbert), you need to use a computer and a simulation tool (NetLogo, in this case).

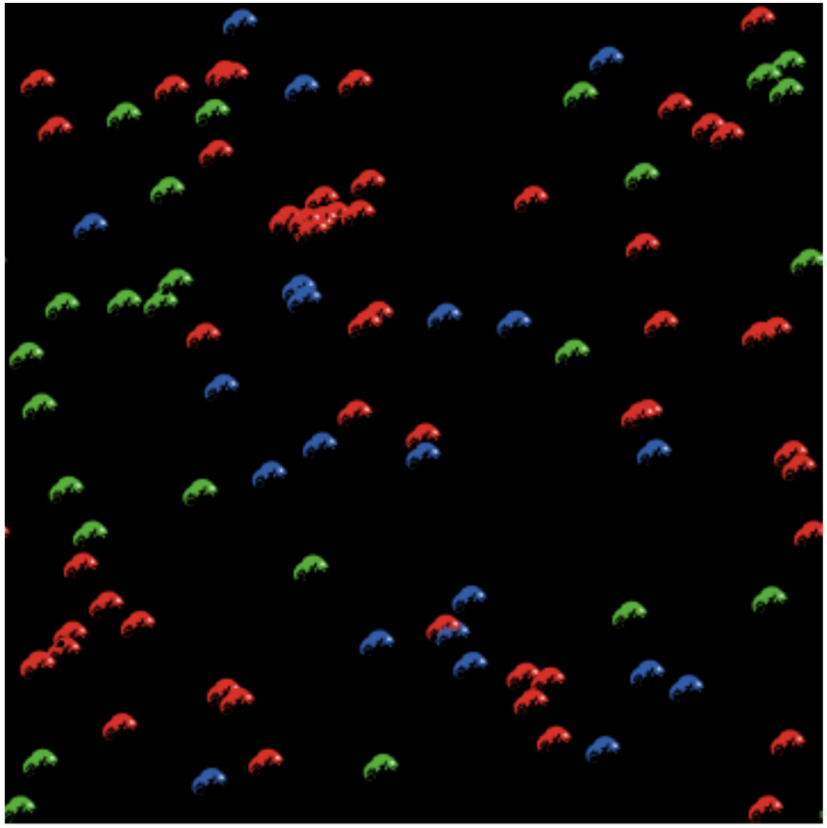


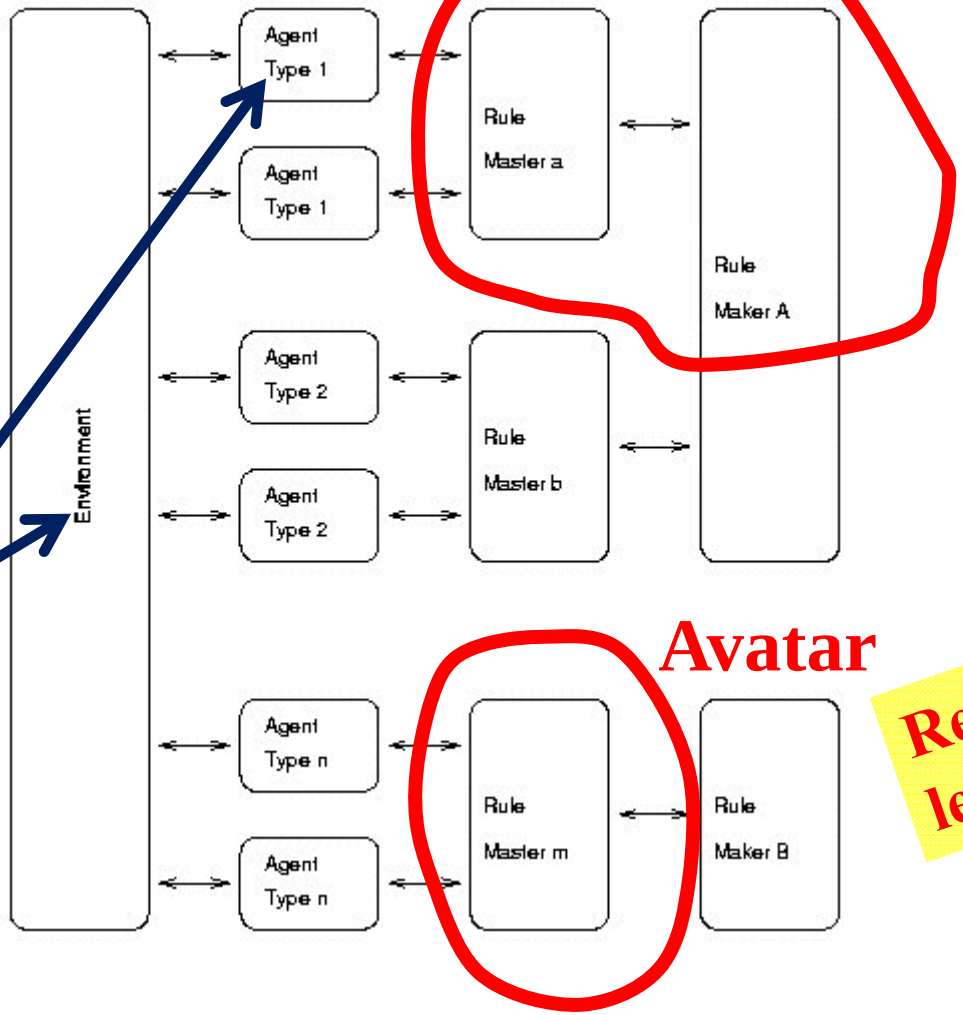


Learning chameleons (http://goo.gl/W9nd8)

In a work of mine you can find, finally, agents requiring a lot of computational capability to learn and behave. They are chameleons changing color when getting in touch with other ones; they can learn strategies, via trials and errors procedures, to avoid that event.







Fixed rules

ANN

(CS)

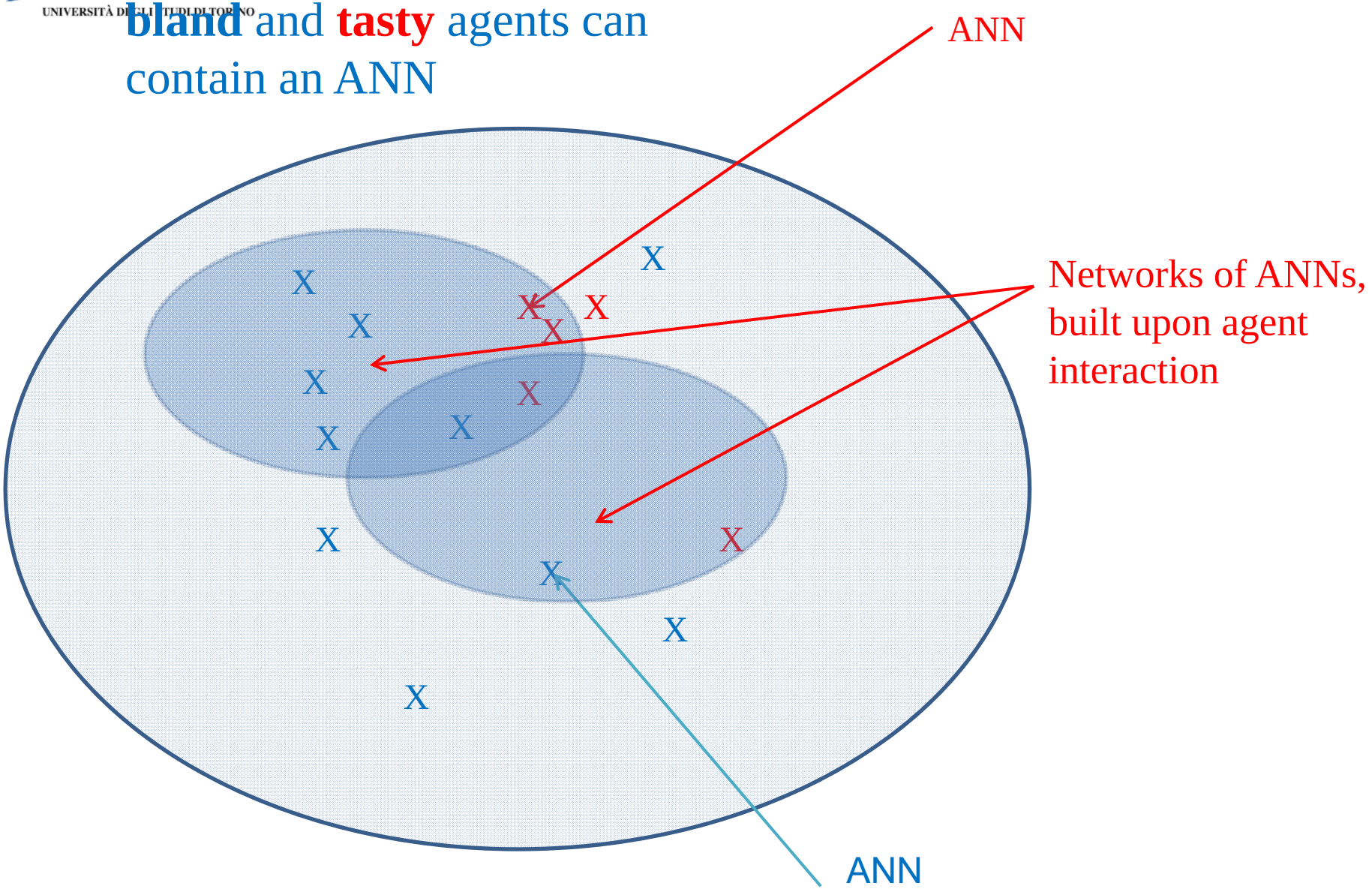
(GA)

Avatar

Reinforcement learning

Microstructures, mainly related to time and parallelism

bland and **tasty** agents can contain an ANN



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

(1) (n+m)

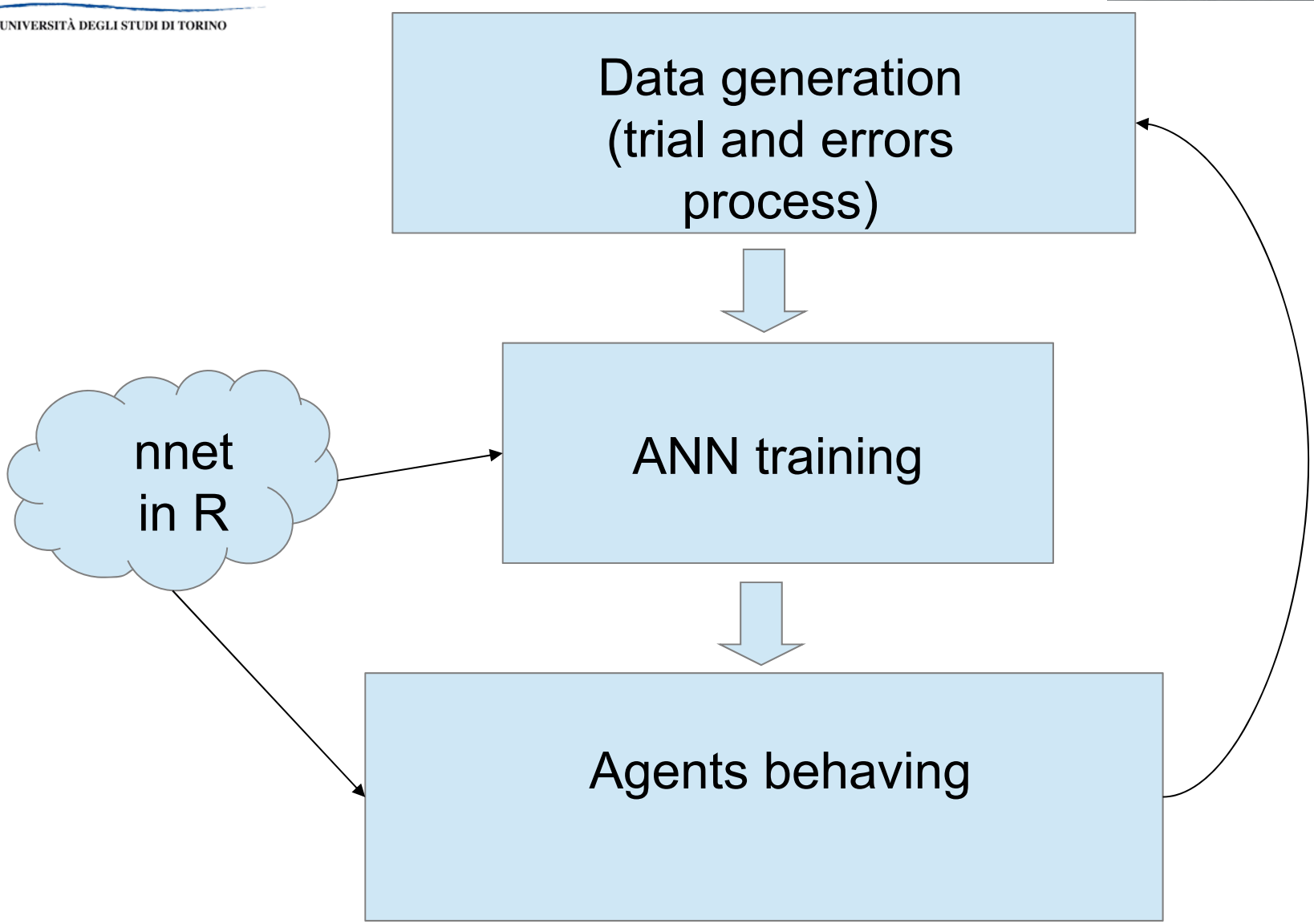
effect information action/s

or, if $z = \{z_1, z_2, \ldots, z_m\}$

$$y = g_m(x,z) = f(B\ f(A\ (x)))$$

(m) (n)

an effect for each possible action


Data generation
(trial and errors
process)
nnet
in R
ANN training
Agents behaving

… and networks?

A step in the project: agents + networks

(Agent-based models meet network analysis: the policy-making perspective)

with Magda Fontana, University of Torino
magda.fontana@unito.it

Considering together the agent-based and network techniques, we have a further important possibility.

Being easier to have network data (i.e. social network data) than detailed behavioral individual information, we can try to understand the links between the dynamic changes of the networks emerging from agent-based models and the behavior of the agents.

As we understand these links, we can apply them to actual networks, to guess about the content of the behavioral black boxes of real-world agents.

How to generate emerging networks to experiment with them

recipeWorld is an agent-based model that simulates the emergence of network out of decentralized autonomous interaction.

The rationale behind it is to offer a few hints to find a framework and a grammar that are flexible and straightforward enough to encompass the widest possible range of purposeful and socially meaningful individual and organizational behavior.

The metaphor is that of a recipe, i.e. *a set of directions with a list of ingredients for making or preparing something, especially food* (as defined in the American Heritage dictionary).

Technically, recipes are sequences of numerical or alphanumerical codes, reported in vectors, and move from an agent to another determining the events and generating the edges of the emerging networks.

Recipes are coded as strings of numbers – their components. Each number (or, if you want, each label), is related to an act, a sub-routine, of the modeled action.

For instance:
[3 1 7 6] means:
execute step 3, then
execute step 1, then
…

Recipes can be of any length and can contain subpart with specific structural characteristics, such as:

[1 4 (3 4 5) 8]
where the instructions in round brackets have to be run in a parallel way; or

[7 4 {10} 9 2]
where the part in curly brackets has to be run putting together a batch of different recipes to be executed in the same time.

Examples in different fields can be suggested:

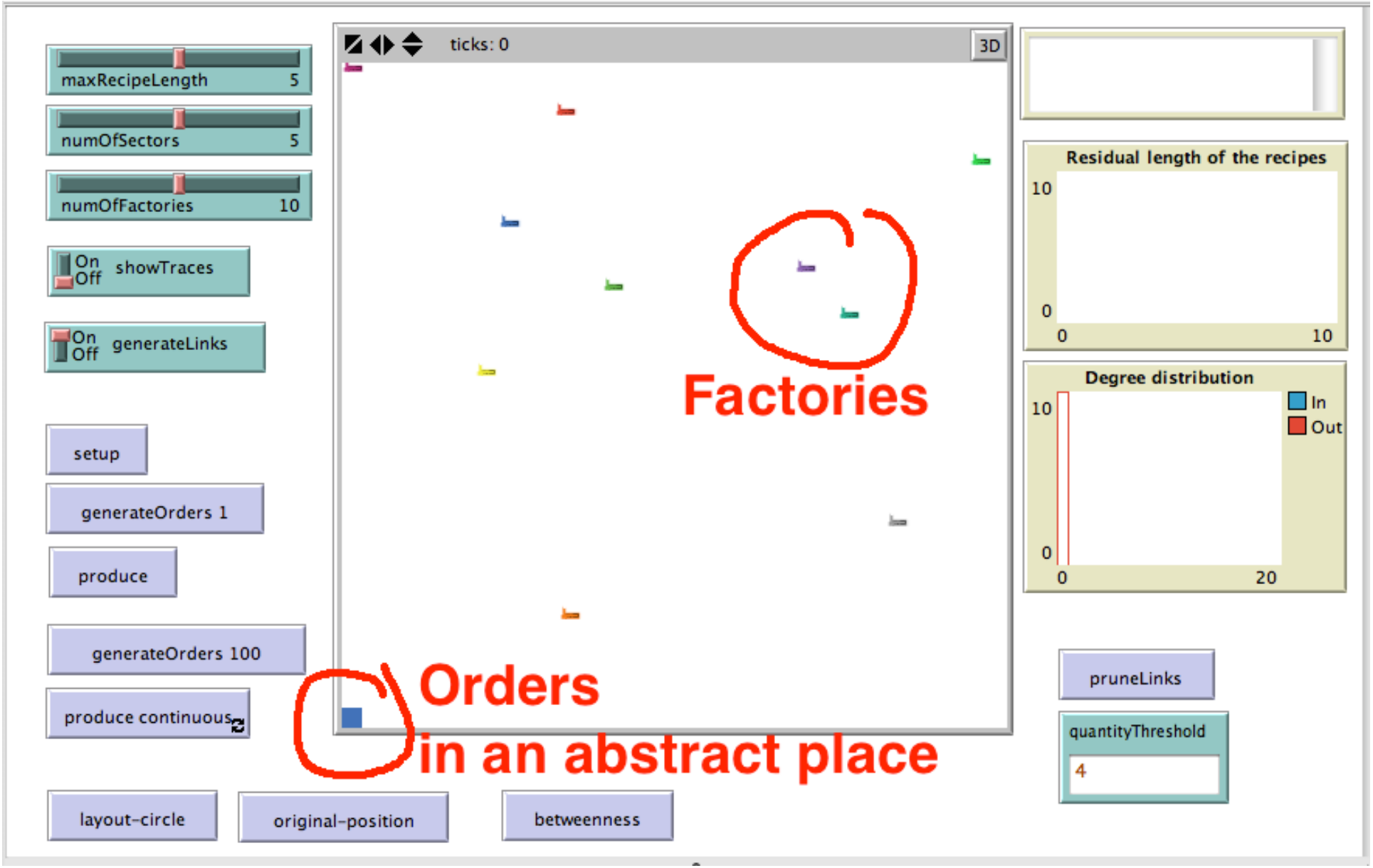
production,

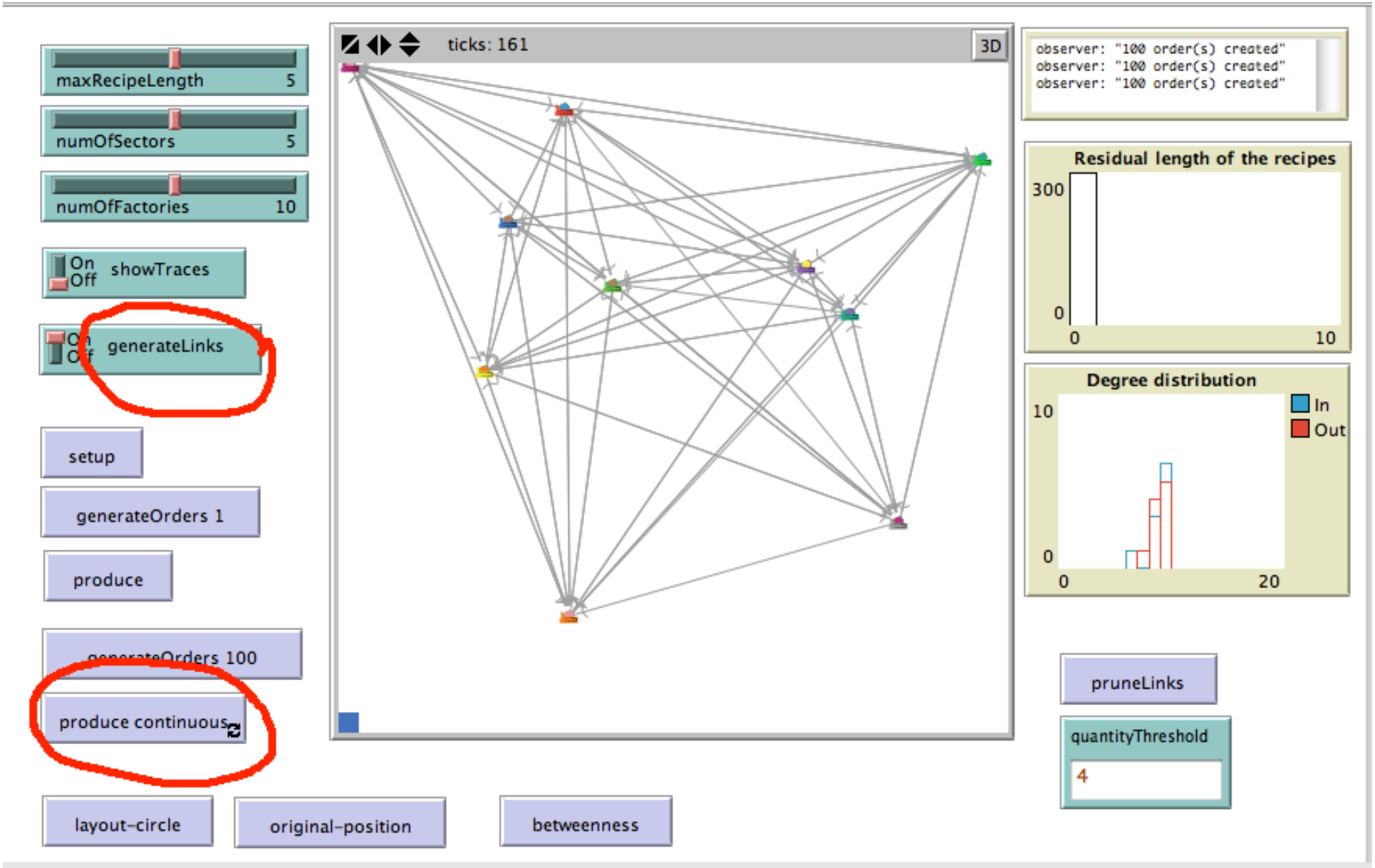
health-care scenarios,

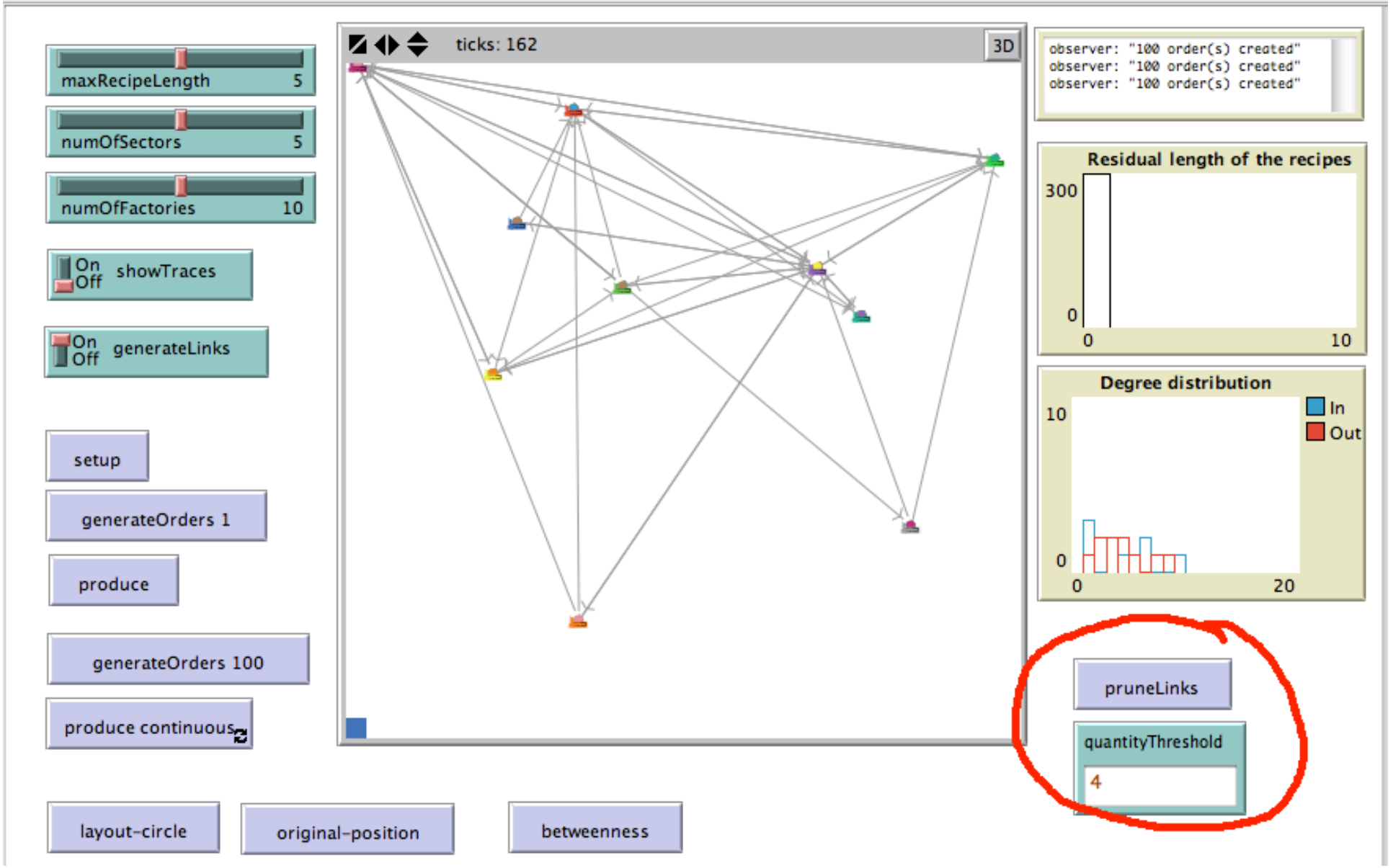
paper co-authorship,

opinion spreading,

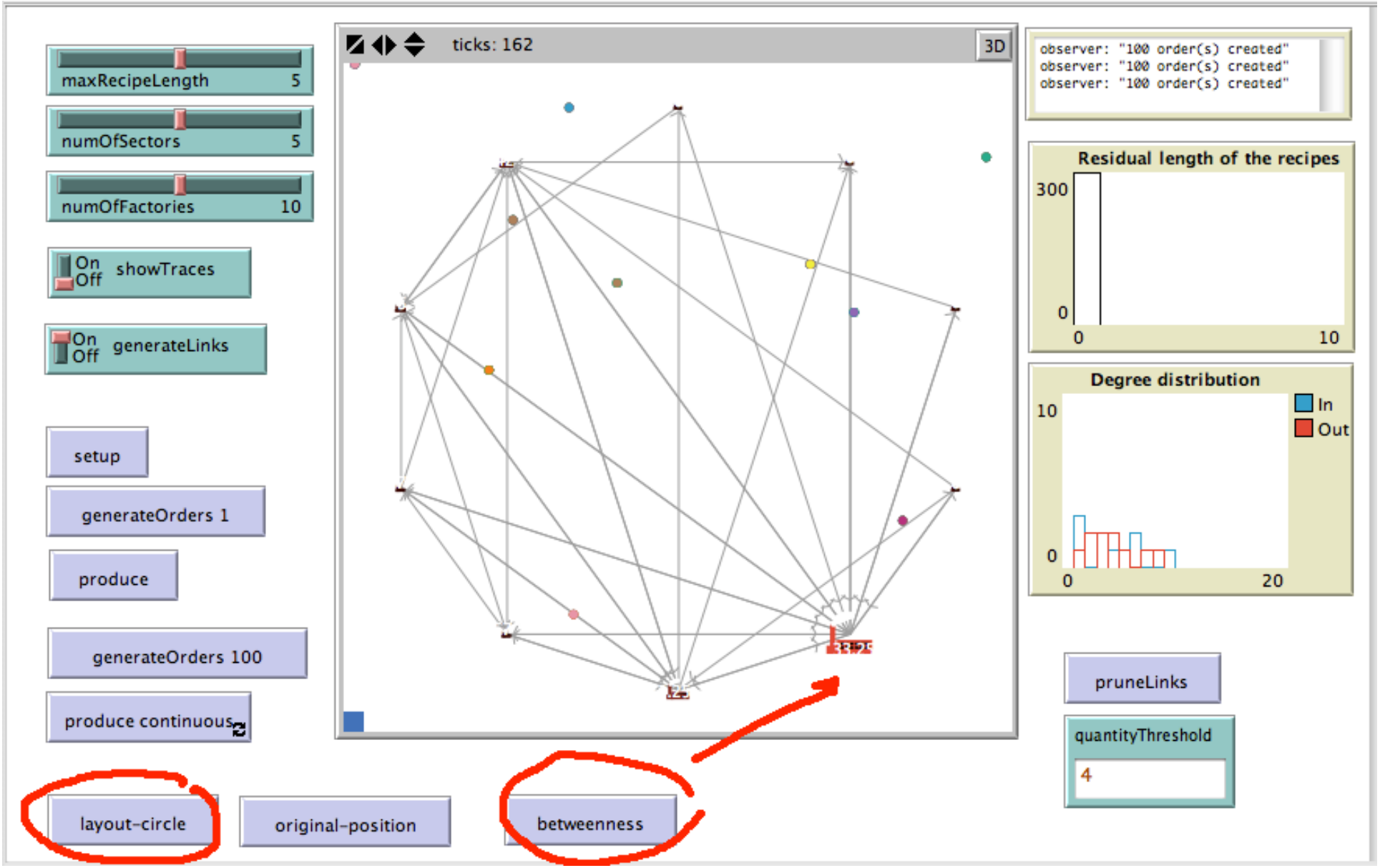
etc.


maxRecipeLength 5
numOfSectors 5
numOfFactories 10
On Off showTraces
On Off generateLinks
setup
generateOrders 1
produce
generateOrders 100
produce continuous
layout-circle
original-position
betweenness
ticks: 0
3D
Factories
Orders
in an abstract place
Residual length of the recipes
10
0
0
10
Degree distribution
10
0
0
20
In
Out
pruneLinks
quantityThreshold
4

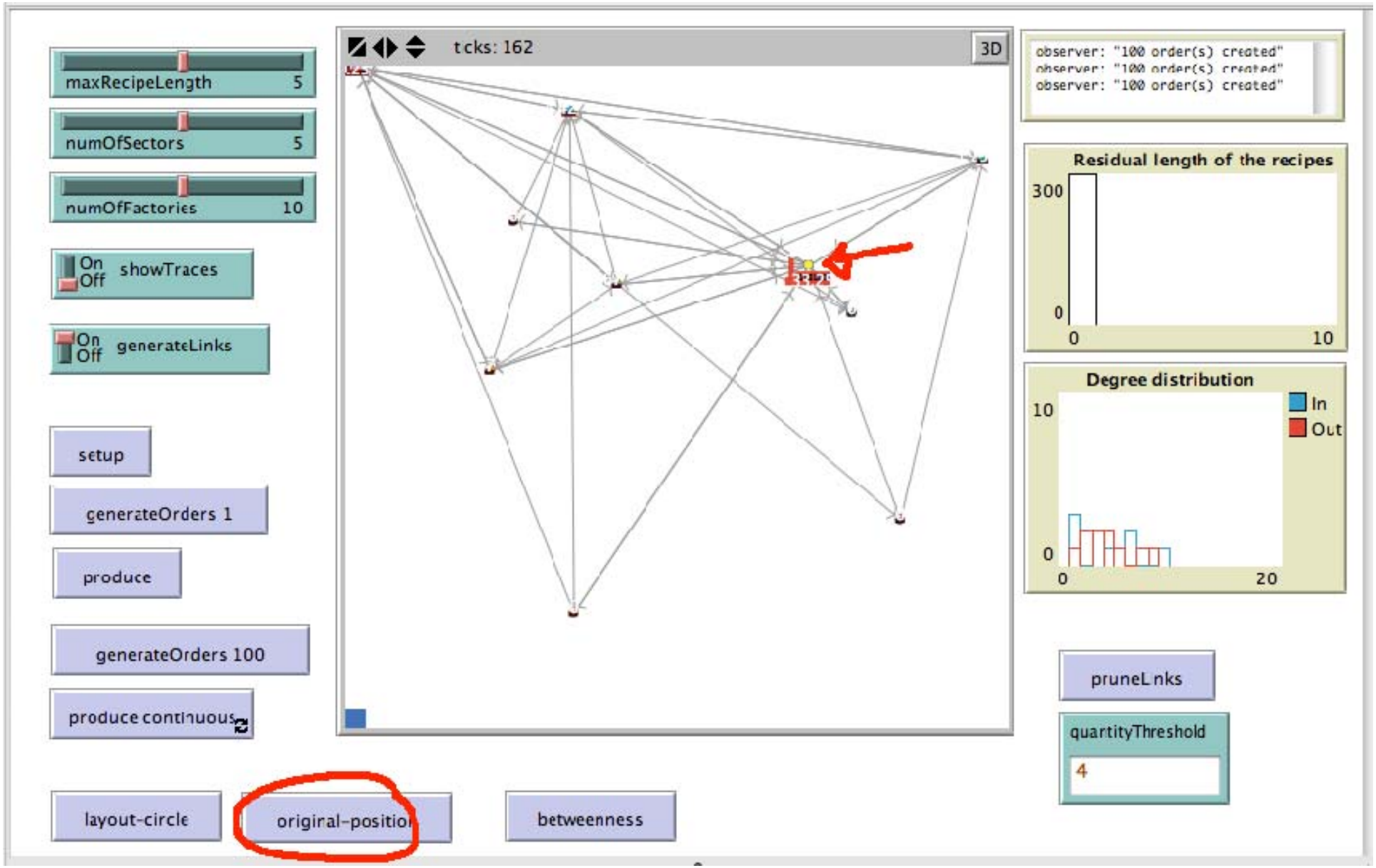

maxRecipeLength 5
numOfSectors 5
numOfFactories 10
On Off showTraces
On Off generateLinks
setup
generateOrders 1
produce
generateOrders 100
produce continuous
layout-circle
original-position
betweenness
ticks: 161
3D
observer: "100 order(s) created"
observer: "100 order(s) created"
observer: "100 order(s) created"
Residual length of the recipes
300
0
0
10
Degree distribution
10
0
0
20
In
Out
pruneLinks
quantityThreshold
4


maxRecipeLength 5
numOfSectors 5
numOfFactories 10
On Off showTraces
On Off generateLinks
setup
generateOrders 1
produce
generateOrders 100
produce continuous
layout-circle
original-position
betweenness
ticks: 162
3D
observer: "100 order(s) created"
observer: "100 order(s) created"
observer: "100 order(s) created"
Residual length of the recipes
300
0
0
10
Degree distribution
10
0
0
20
In
Out
pruneLinks
quantityThreshold
4

Calculations are made using the new NW NetLogo extension
https://github.com/NetLogo/NW-Extension



Calculations are made using the new NW NetLogo extension
https://github.com/NetLogo/NW-Extension



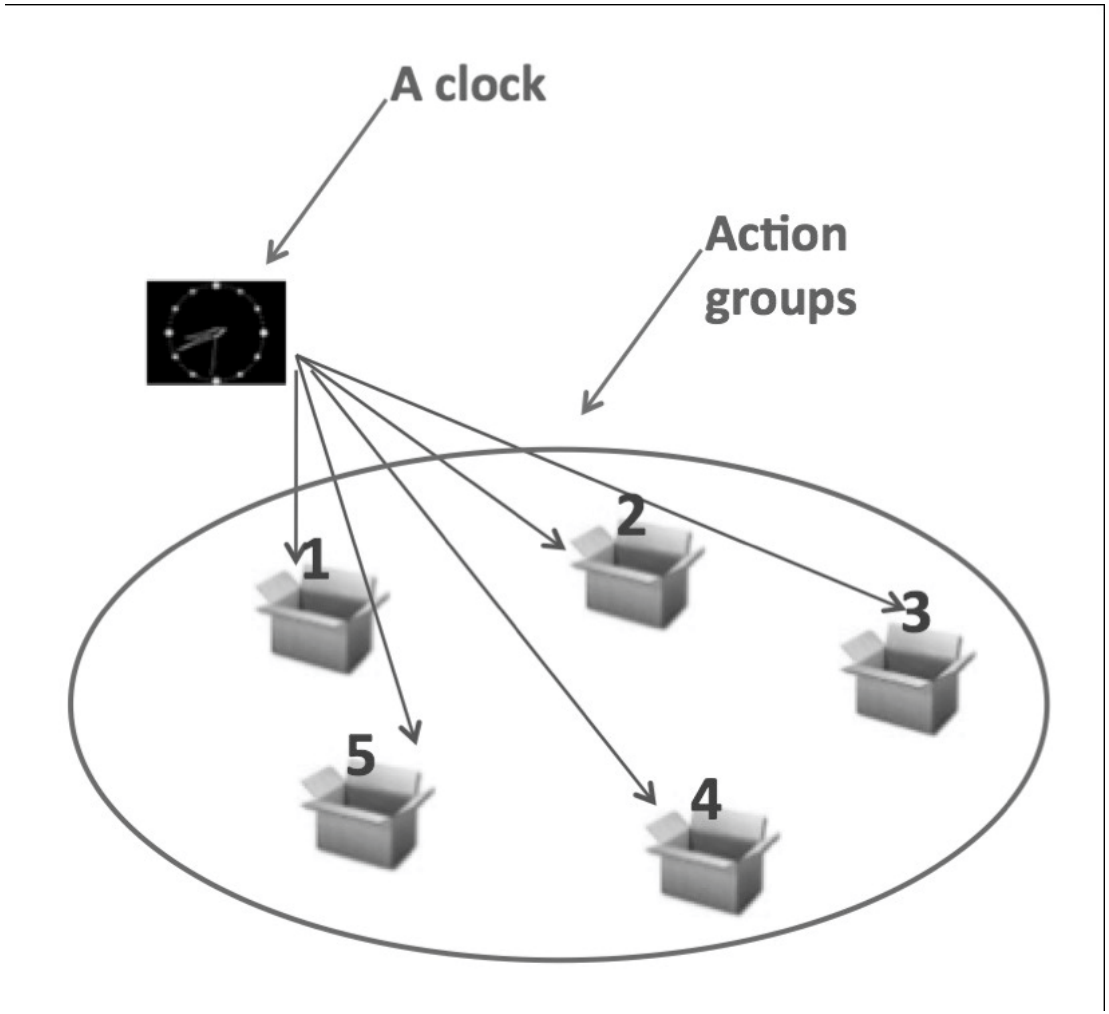
What is making “special” this result is that, in this context, agents are activated (following their internal rules and capabilities) by the events.

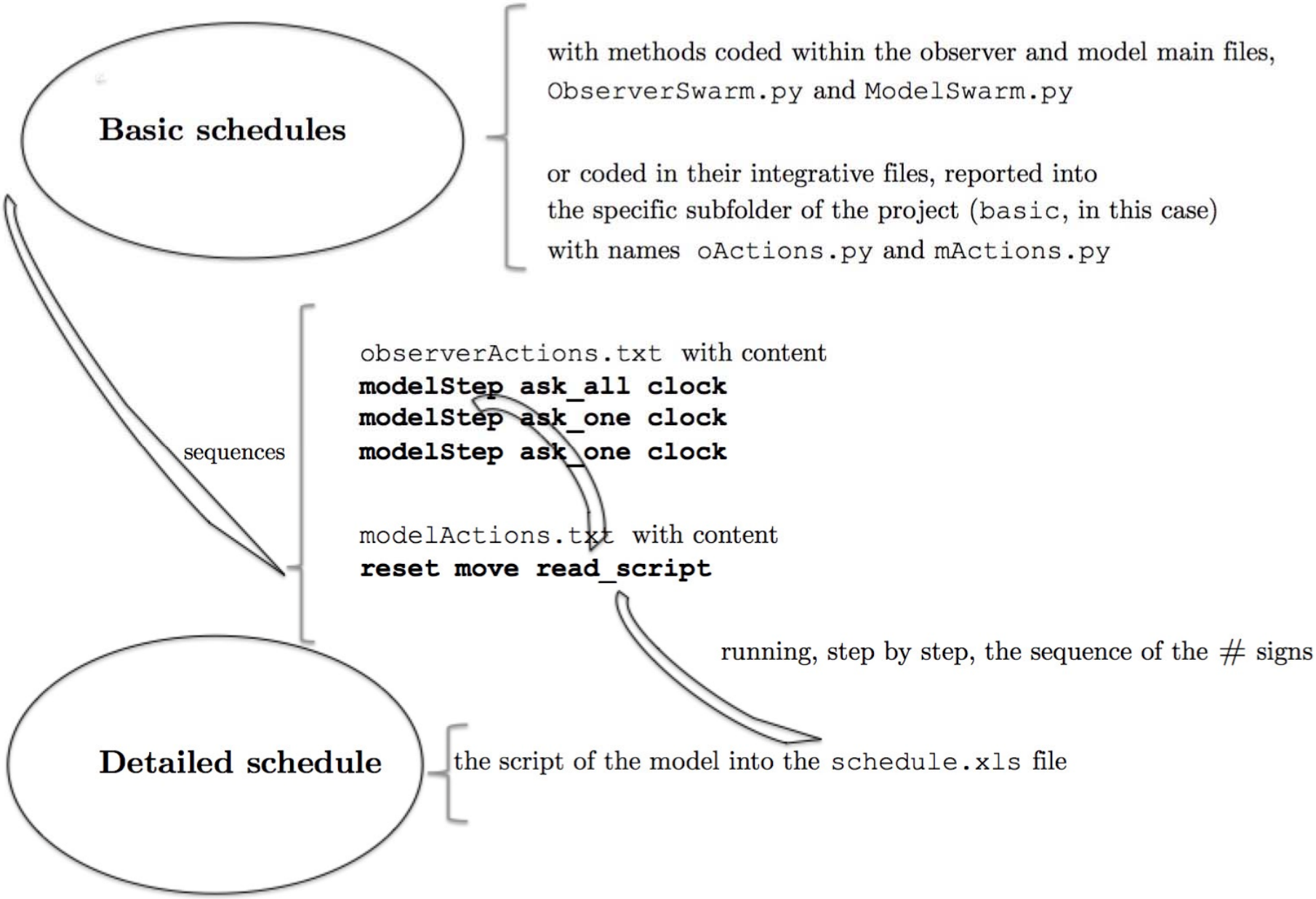
The network emerges as a side effect, as in the real world.

A quick look to SLAPP

Swarm-Like Agent Protocol in Python

A basic example to introduce the Schedule


A clock
Action groups
1
2
3
4
5


Basic schedules
with methods coded within the observer and model main files, ObserverSwarm.py and ModelSwarm.py
or coded in their integrative files, reported into the specific subfolder of the project (basic, in this case) with names oActions.py and mActions.py
sequences
observerActions.txt with content
modelStep ask_all clock
modelStep ask_one clock
modelStep ask_one clock
modelActions.txt with content
reset move read_script
running, step by step, the sequence of the # signs
Detailed schedule
the script of the model into the schedule.xls file

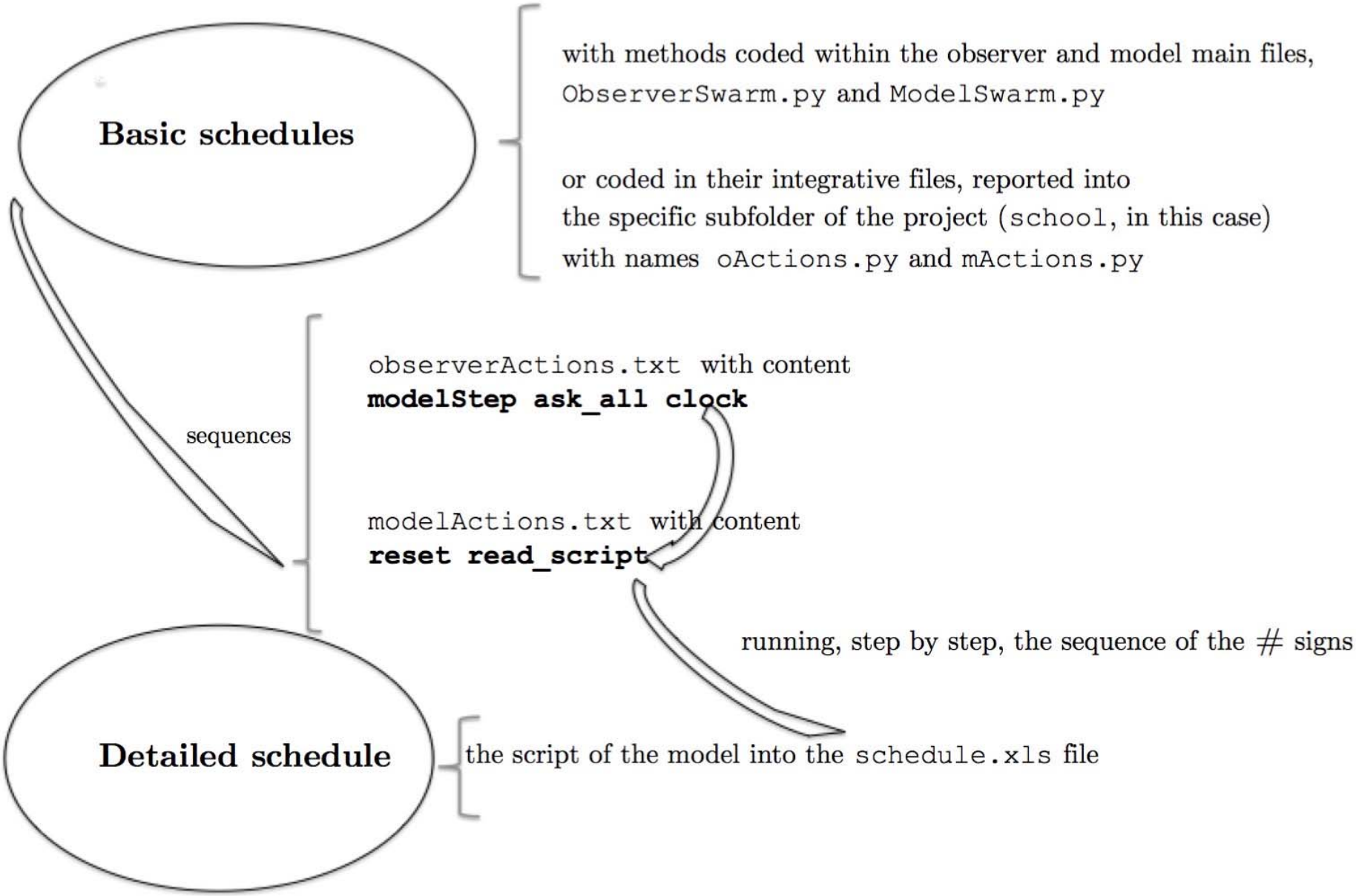
	A	B	C	D
1	#	1		
2	bland	eat		
3	bland	dance		
4	#	2		
5	#	4		
6	all	0.5	dance	
7	tasteC	eat		
8	#	5		
9	all	eat		
10	all	dance		
11	#	7		
12	tasteA	0.5	dance	
13	#	8		
14	tasteB	dance		

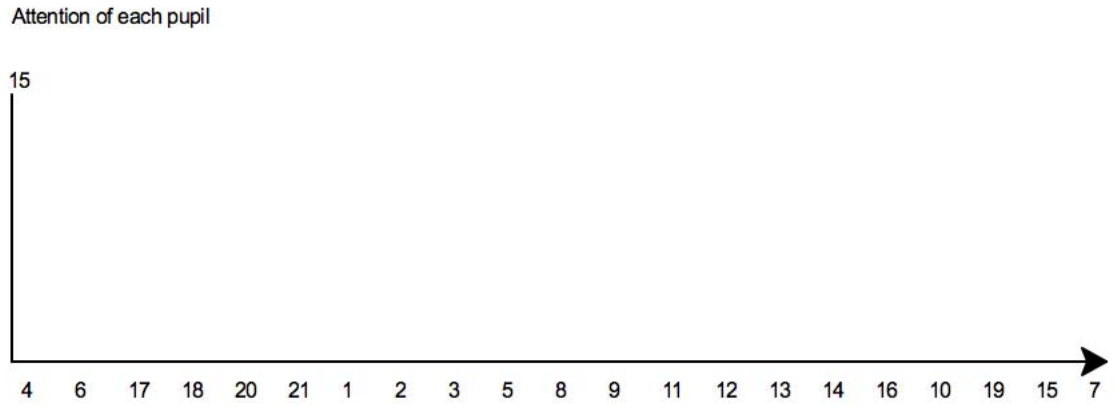
School example (turtle based)

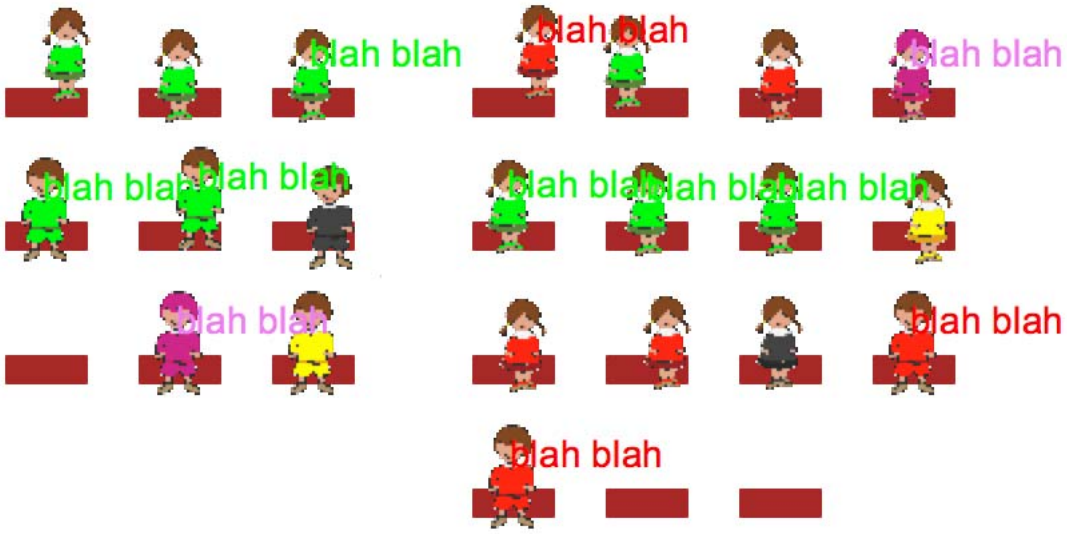
	A	B	C	D	E	F	G	H
1					the first sheet has to be named 'schedule',			
2					the other ones have the names of the macros			
3	#	1			that you are using			
4	macro	transitionPhase						
5	#	2						
6	macro	checkToObtainAttention						
7								

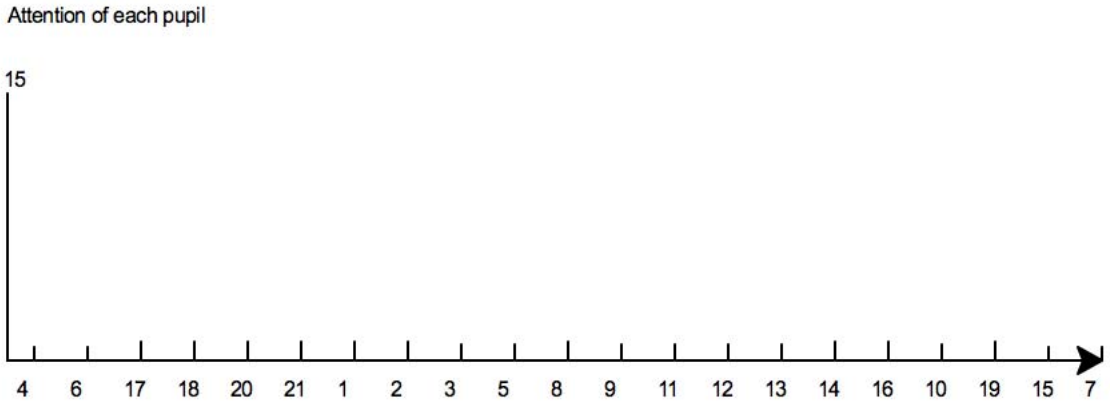
	A	B	C
1	all	0.5	sitDownNotWell
2	all	0.5	talk
3			

	A	B	C
1	all	payAttention	
2	all	sitDownWell	
3	yellowPupil	sitDownNotWell	
4	greenPupil	-1	sitDownNotWell
5	all	0.3	fidget
6	saPupil	0.8	shake
7	greenPupil	-1	shake
8	threeGreen	talkClose	
9	r4r	talkClose	
10	saPupil	0.5	answerBad
11	all	0.15	answerWell
12			


Basic schedules
with methods coded within the observer and model main files,
ObserverSwarm.py and ModelSwarm.py
or coded in their integrative files, reported into
the specific subfolder of the project (school, in this case)
with names oActions.py and mActions.py
sequences
observerActions.txt with content
modelStep ask_all clock
modelActions.txt with content
reset read_script
running, step by step, the sequence of the # signs
Detailed schedule
the script of the model into the schedule.xls file


Attention of each pupil
15
4 6 17 18 20 21 1 2 3 5 8 9 11 12 13 14 16 10 19 15 7

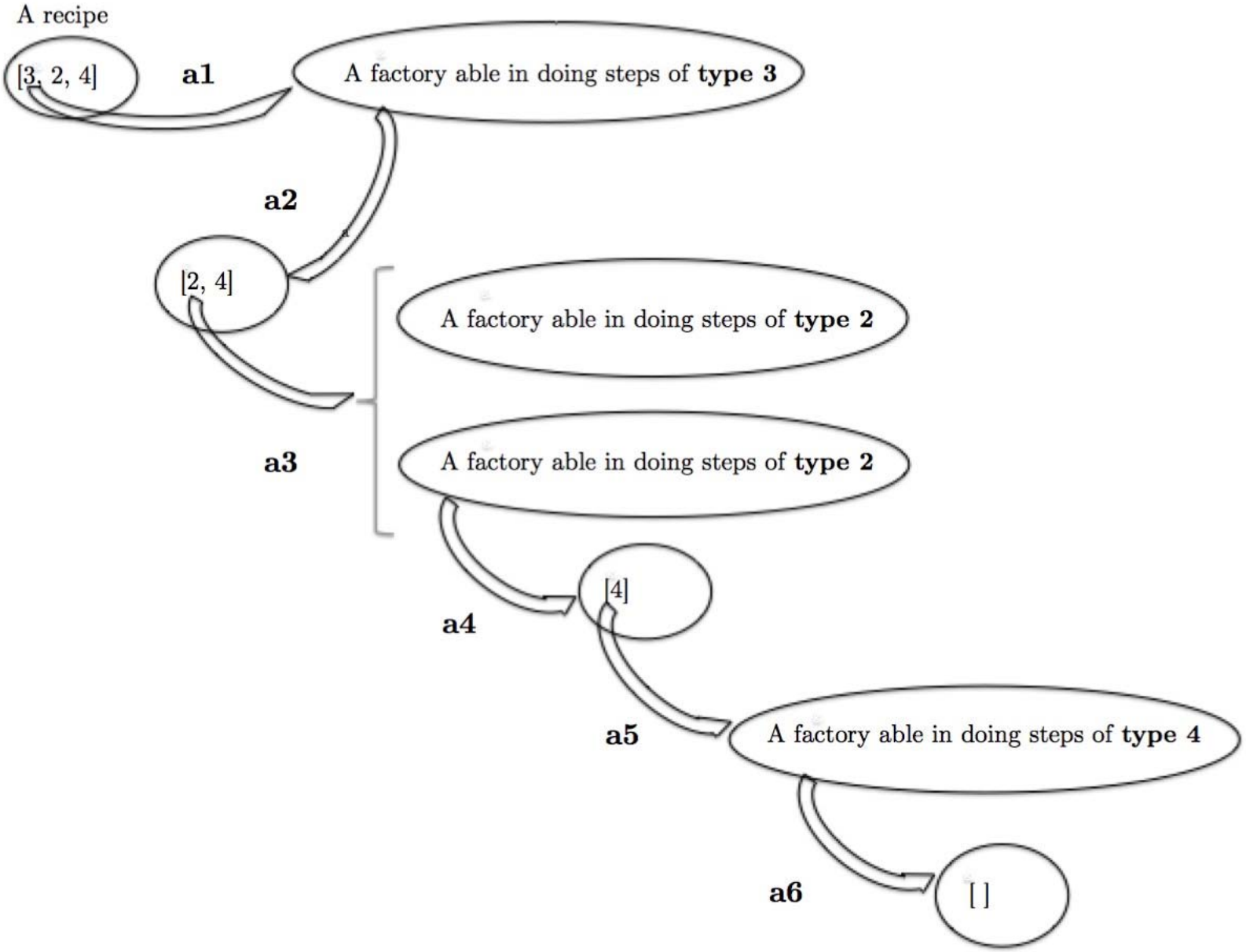

blah blah
blah blah
blah blah
blah blah
blah blah
blah blah
blah blah
blah blah
blah blah
blah blah


Attention of each pupil
15
4 6 17 18 20 21 1 2 3 5 8 9 11 12 13 14 16 10 19 15 7


pss pss pss pss pss pss pss pss
Shaking
answ ok
Shaking

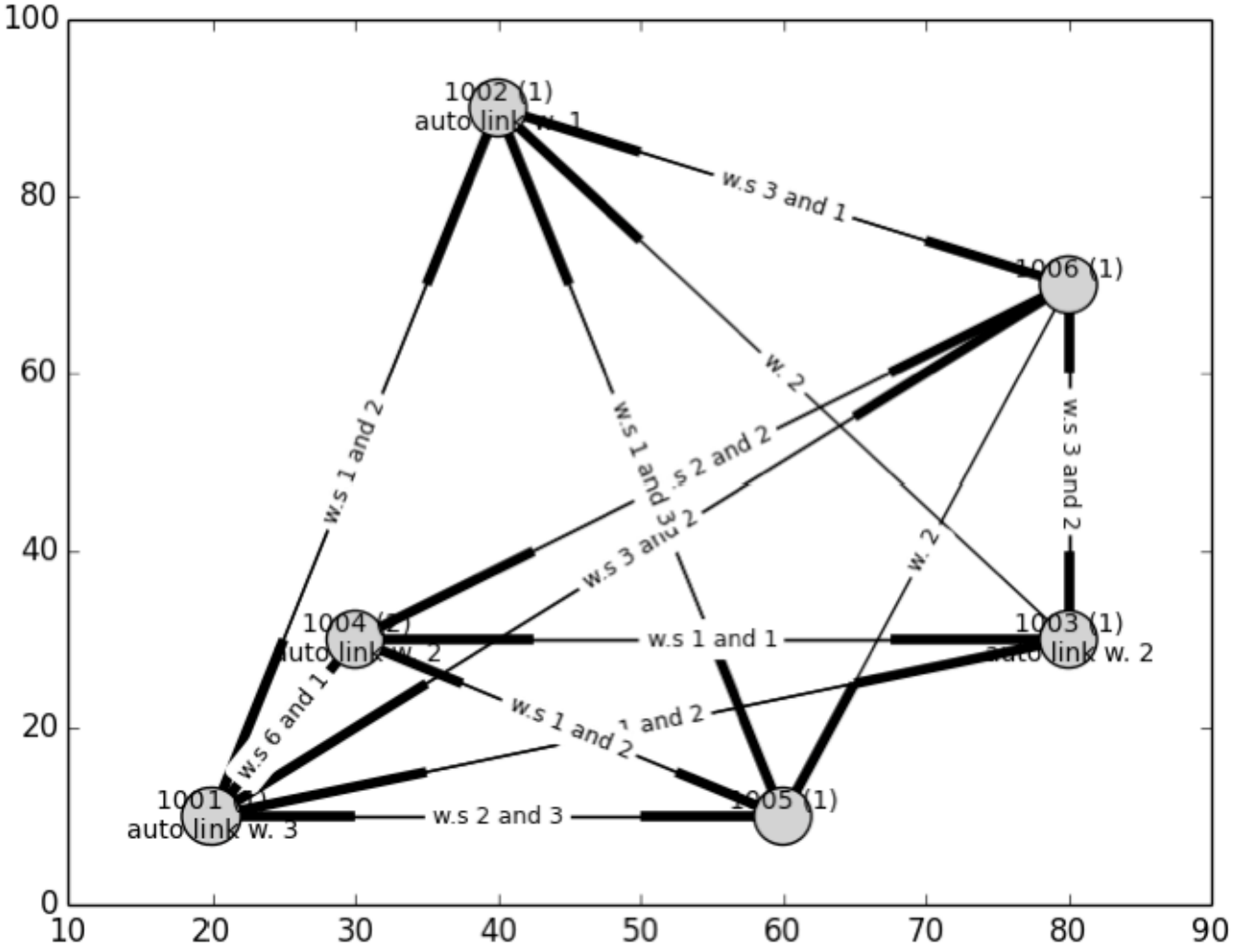
	A	B	C
1	all	payAttention	
2	all	sitDownWell	
3	yellowPupil	sitDownNotWell	
4	greenPupil	-1	sitDownNotWell
5	all	0.3	fidget
6	saPupil	0.8	shake
7	verdePupil	-1	shake
8	saPupil	shakeIf_greenPupil	
9	threeGreen	talkClose	
10	r4r	talkClose	
11	saPupil	0.5	answerBad
12	all	0.15	answerWell
13			

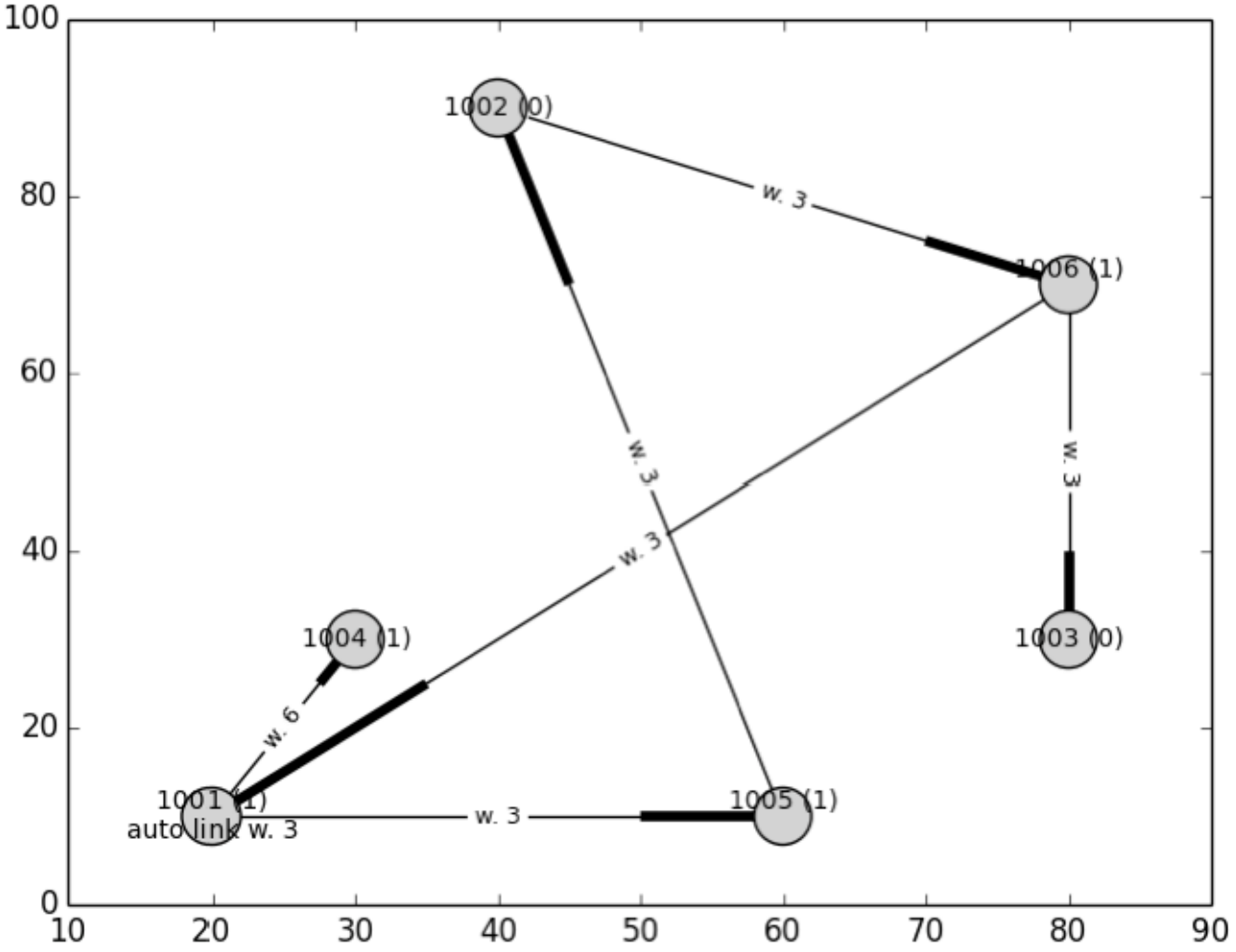
Production example (network analysis based)

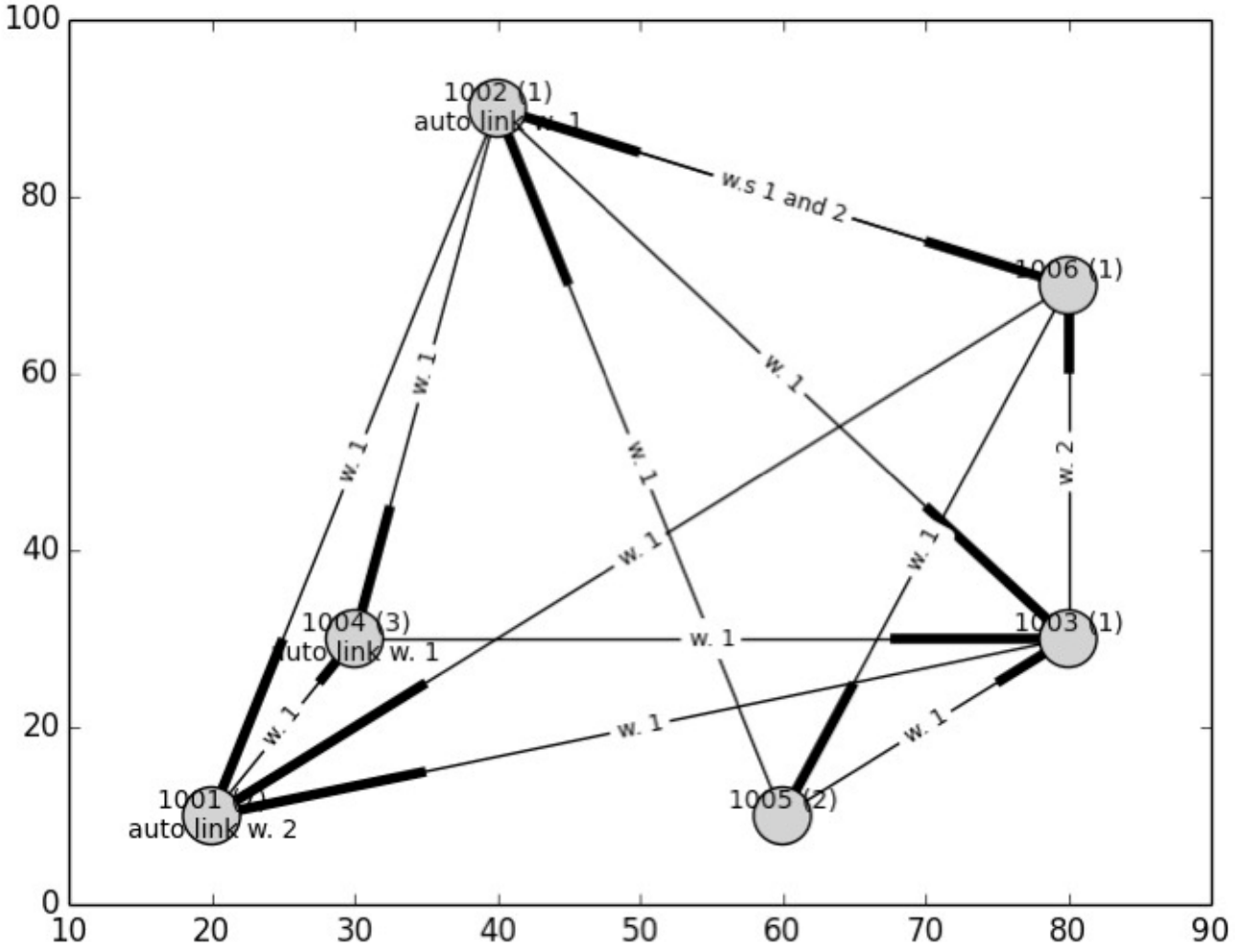

A recipe
[3, 2, 4]
a1
A factory able in doing steps of type 3
a2
[2, 4]
A factory able in doing steps of type 2
a3
A factory able in doing steps of type 2
[4]
a4
a5
A factory able in doing steps of type 4
a6
[ ]

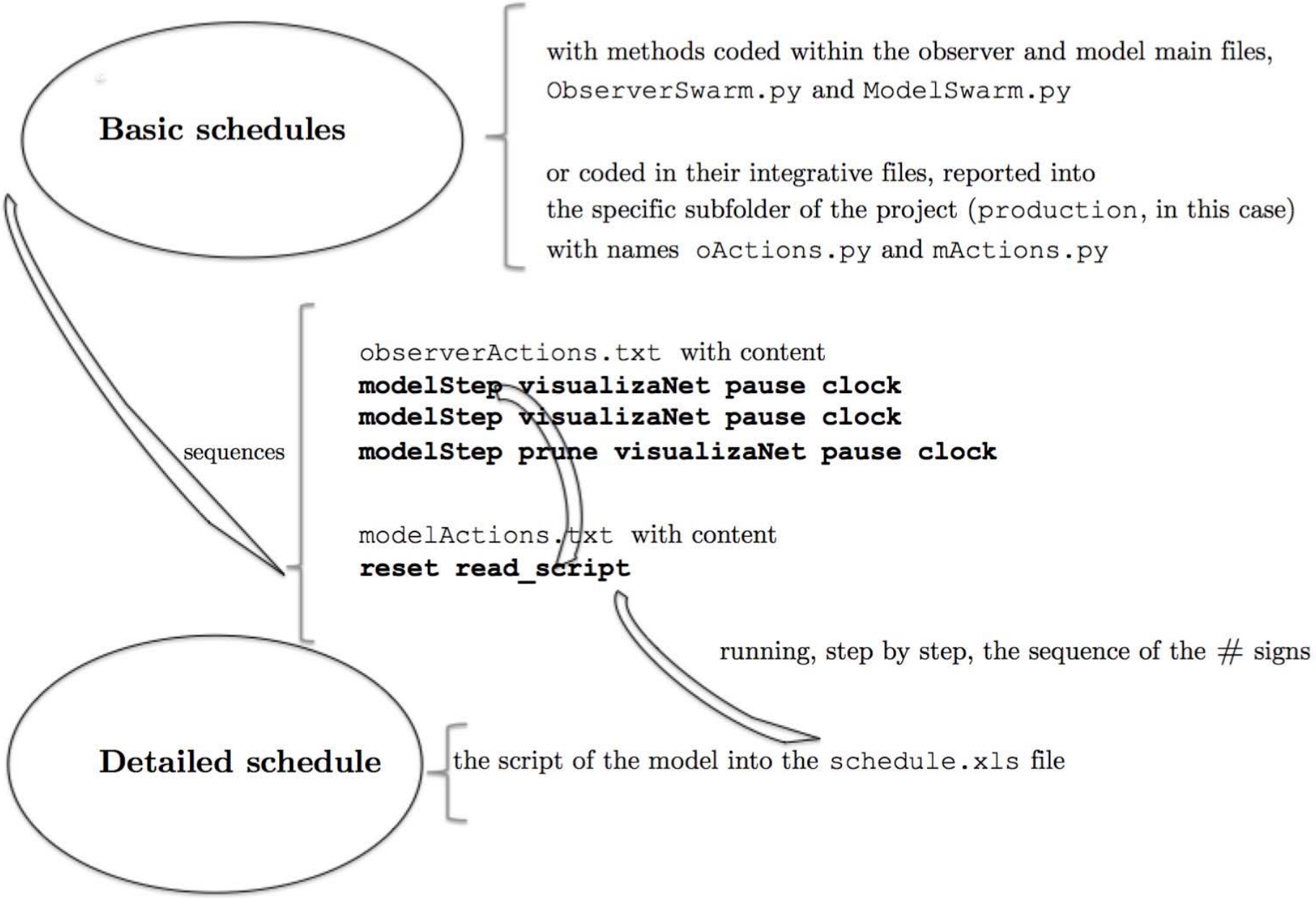
	A	B	C
1	#	1	20
2	recipes	setRecipeContent	
3	recipes	searchForSector	
4	factories	produce	
5			

	A	B	C
1	#	1	20
2	recipes	setRecipeContent	
3	recipes	searchForSector	
4	factories	0.2	produce
5	sector2	produce	
6	sector5	produce	


100
80
60
40
20
0
10
20
30
40
50
60
70
80
90
1002 (1)
auto link w. 1
w.s 3 and 1
1006 (1)
w. 2
w.s 1 and 2
w.s 1 and 3
w.s 2 and 2
w.s 3 and 2
w.s 3 and 2
w. 2
w.s 3 and 2
1004 (2)
auto link w. 2
w.s 1 and 1
1003 (1)
auto link w. 2
w.s 6 and 1
w.s 1 and 2
w.s 1 and 2
1001 (1)
auto link w. 3
w.s 2 and 3
1005 (1)


100
80
60
40
20
0
10
20
30
40
50
60
70
80
90
1002 (0)
w. 3
1006 (1)
w. 3
w. 3
w. 3
1004 (1)
1003 (0)
w. 6
1001 (1)
auto link w. 3
w. 3
1005 (1)

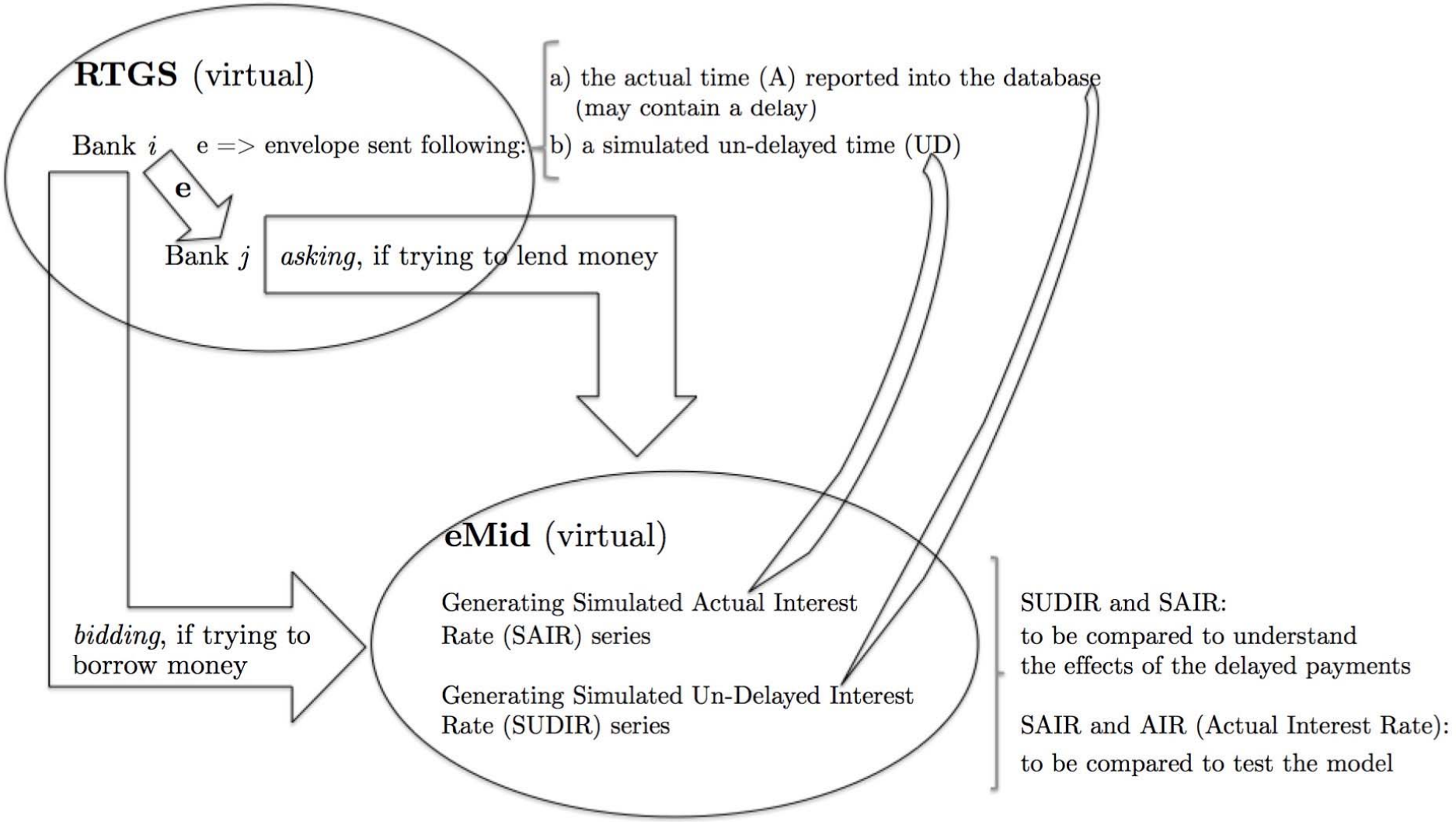

100
80
60
40
20
0
10
20
30
40
50
60
70
80
90
1002 (1)
auto link w. 1
w.s 1 and 2
1006 (1)
w. 1
w. 1
w. 1
w. 2
w. 1
w. 1
w. 1
1004 (3)
auto link w. 1
w. 1
1003 (1)
w. 1
w. 1
w. 1
w. 1
1001 (2)
auto link w. 2
1005 (2)

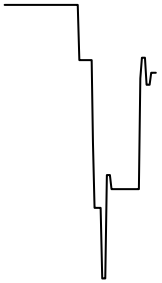

Basic schedules
with methods coded within the observer and model main files,
ObserverSwarm.py and ModelSwarm.py
or coded in their integrative files, reported into
the specific subfolder of the project (production, in this case)
with names oActions.py and mActions.py
sequences
observerActions.txt with content
modelStep visualizaNet pause clock
modelStep visualizaNet pause clock
modelStep prune visualizaNet pause clock
modelActions.txt with content
reset read_script
running, step by step, the sequence of the # signs
Detailed schedule
the script of the model into the schedule.xls file

	A	B	C
1	macro	repeat	
2	#	3	
3	factories	-2	addAFactory
4	#	6	
5	factories	0,2	addAFactory
6	#	9	
7	factories	-3	removeItself

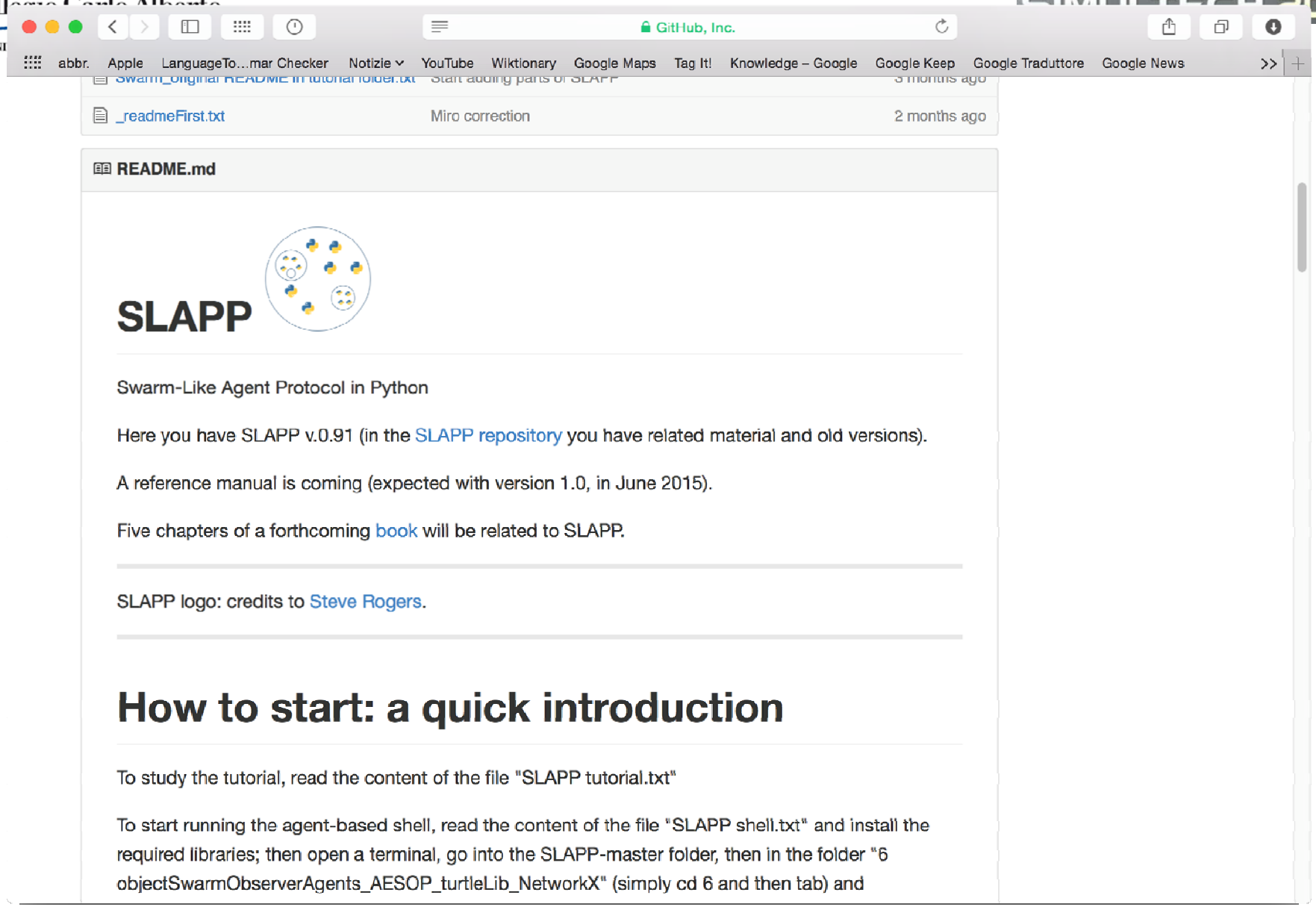
	A	B	C
1	#	1	20
2	recipes	setRecipeContent	
3	recipes	searchForSector	
4	factories	produce	
5			

Payments example (using R)


RTGS (virtual)
Bank i
e => envelope sent following:
a) the actual time (A) reported into the database (may contain a delay)
b) a simulated un-delayed time (UD)
e
Bank j
asking, if trying to lend money
bidding, if trying to borrow money
eMid (virtual)
Generating Simulated Actual Interest Rate (SAIR) series
Generating Simulated Un-Delayed Interest Rate (SUDIR) series
SUDIR and SAIR:
to be compared to understand
the effects of the delayed payments
SAIR and AIR (Actual Interest Rate):
to be compared to test the model



https://github.com/terna/SLAPP/


GitHub, Inc.
abbr. Apple LanguageTo...mar Checker Notizie YouTube Wiktionary Google Maps Tag It! Knowledge – Google Google Keep Google Traduttore Google News
_readmeFirst.txt
Miro correction
2 months ago
README.md
SLAPP
Swarm-Like Agent Protocol in Python
Here you have SLAPP v.0.91 (in the SLAPP repository you have related material and old versions).
A reference manual is coming (expected with version 1.0, in June 2015).
Five chapters of a forthcoming book will be related to SLAPP.
SLAPP logo: credits to Steve Rogers.
How to start: a quick introduction
To study the tutorial, read the content of the file "SLAPP tutorial.txt"
To start running the agent-based shell, read the content of the file "SLAPP shell.txt" and install the required libraries; then open a terminal, go into the SLAPP-master folder, then in the folder "6 objectSwarmObserverAgents_AESOP_turtleLib_NetworkX" (simply cd 6 and then tab) and

Thanks

Pietro Terna
pietro.terna@unito.it