French Institute
For Computer Science
and
Applied Mathematics

What can we learn from the Porgy experience?

Hélène Kirchner

WPTE 2020

29/06/2020

Overview

Porgy is a visual and interactive platform for modelling, simulating and analyzing complex systems based on graph rewriting.
https://porgy.labri.fr/
sources on github (https://github.com/bpinaud/Porgy)

Original idea: couple graph visualisation (dynamic graphs) and graph rewriting (better understand)

Joint work with teams of Maribel Fernandez from King's College London and Bruno Pinaud Bordeaux University.
The development of Porgy started in 2009 and as a plug-in of the Tulip system.

In this talk: Short presentation of Porgy (port graphs, rewrite rules, strategies) with summary of two main application domains (biology, social networks)
With the objective to explain how we have addressed a few *specific questions*

Summary

1. General presentation of the framework. An overview of Porgy with
   Concepts (port graphs, rewrite rules, strategies)
   A strategy language for computation control: *how to accurately control positions ?*

2. Application to Biological systems
   How to handle uncertain information ?
   Illustrate data analysis features

3. Application to Social networks
   How to compare and prove different models ?

4. Labeled graphs as a uniform data structure : *rules as labeled graphs*

5. To go further
   - Multilayer graphs
   - Semantics
   - Proof environment

Bibliography

[1] Strategic Port Graph Rewriting: an Interactive Modelling Framework
Maribel Fernández, Hélène Kirchner, Bruno Pinaud
Mathematical Structures in Computer Science, Cambridge University Press (CUP), 2019, 29 (5), pp.615--662. ⟨10.1017/S0960129518000270⟩

[2] Strategy-Driven Exploration for Rule-Based Models of Biochemical Systems with Porgy
Oana Andrei, Maribel Fernández, Hélène Kirchner, Bruno Pinaud
Bill Hlavacek. *Modeling Biomolecular Site Dynamics*, 1945, ⟨Springer⟩, pp 43-70, 2019, Methods in Molecular Biology, 978-1-4939-9100-6. ⟨10.1007/978-1-4939-9102-0_3⟩

[3] Labelled Graph Strategic Rewriting for Social Networks
Maribel Fernandez, Hélène Kirchner, Bruno Pinaud, Jason Vallet
Journal of Logical and Algebraic Methods in Programming, Elsevier, 2018, 96 (C), pp.12--40. ⟨10.1016/j.jlamp.2017.12.005⟩

[4] Labelled Port Graph – A Formal Structure for Models and Computations
Maribel Fernández, Hélène Kirchner, Bruno Pinaud
The 12th Workshop on Logical and Semantic Frameworks, with Applications (LSFA 2017), Sep 2017, Brasília, Brazil. pp.3 - 21, ⟨10.1016/j.entcs.2018.10.002⟩

Summary

1. **General presentation of the framework. An overview of Porgy with**
   Concepts (port graphs, rewrite rules, strategies)
   A strategy language for computation control: *how to accurately control positions ?*

2. Application to Biological systems
   How to handle uncertain information ?
   Illustrate data analysis features

3. Application to Social networks
   How to compare and prove different models ?

4. Labeled graphs as a uniform data structure : *rules as labeled graphs*

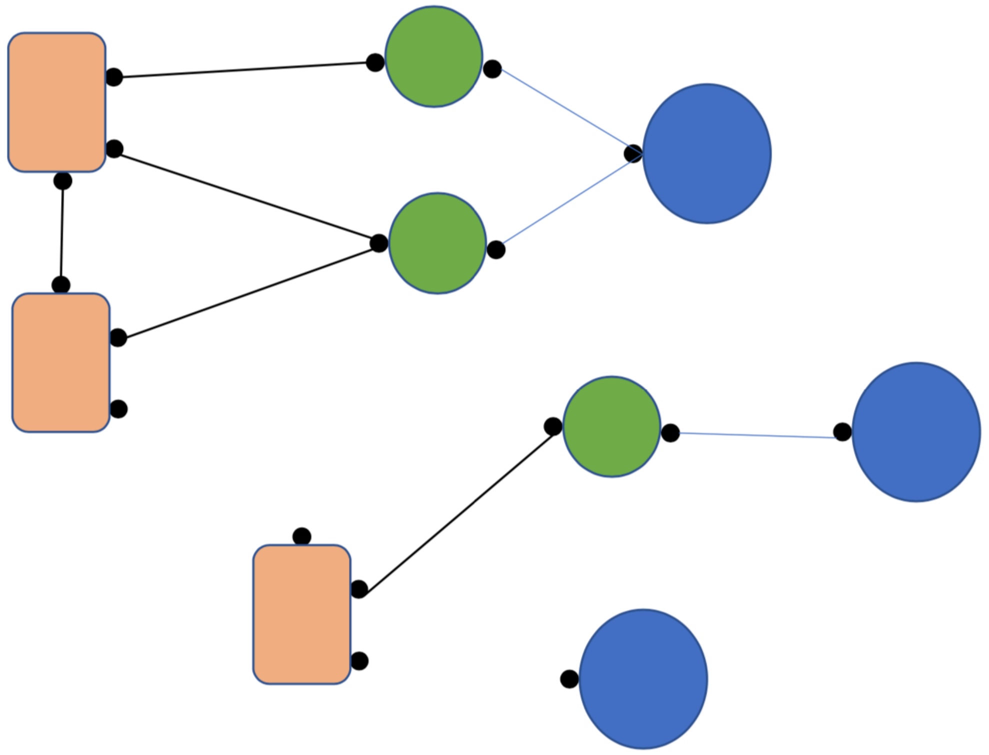
5. To go further
   - Multilayer graphs
   - Semantics
   - Proof environment

Attributed Port Graphs

Used in [IbanescuBK03], [AndreiK07], κ-calculus [DanosL04], BioNetGen [BlinovYFH05]
inspired by protein interactions;

Port graphs are graphs with multiple edges, where:
- **nodes** have explicit connection points, called **ports**
- **edges** attach to ports of nodes
- nodes, edges and ports are labeled by a **set of attributes**.

Actually equivalent to usual labeled graphs, but with more structure
Thus inherits of
- well known reasoning mechanisms based on graph notions : graph homomorphism, categorical and logical formalism
- efficient algorithms and complexity results.



An exemple of port graph

Rewriting step

Rewriting may be concurrent, probabilistic, conditional, with constraints...

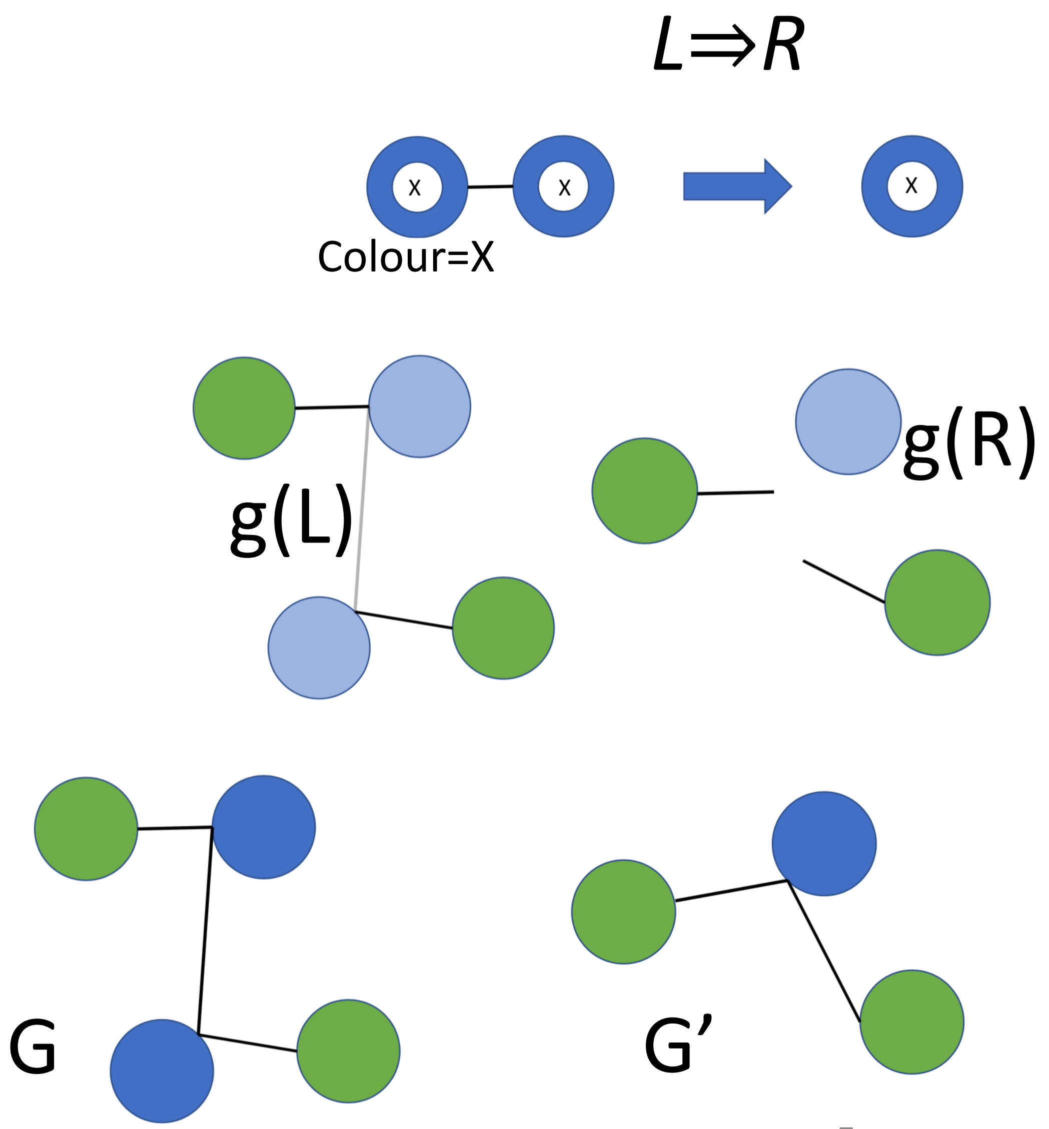
G rewrites to *G′* using the rule *L*⇒*R* if

- there is a subgraph homomorphism $g : L \to G$
  g identifies a subgraph of *H* that is equal to *L* except at positions where *L* has variables (at those positions *H* could have any value).

- and G[H]=G[g(L)] and G'=G[g(R)] *with appropriate rewiring of dangling edges*

This *rewriting step* is denoted

$$G \quad \to_{g,\, L \Rightarrow R} \quad G'$$



Graph rewriting strategies

Rewrite rules describe local transformations
Different choices for application: position(s), rule, matching substitution(s).

A strategy language to formalize the control of rewrite rule application
Which rule(s) to apply ?

Iteration : sequentially, repeat, bounded repeat
*Choice : one, all, ppick, orelse, if_then_else_*

Where to apply a rule in a graph?

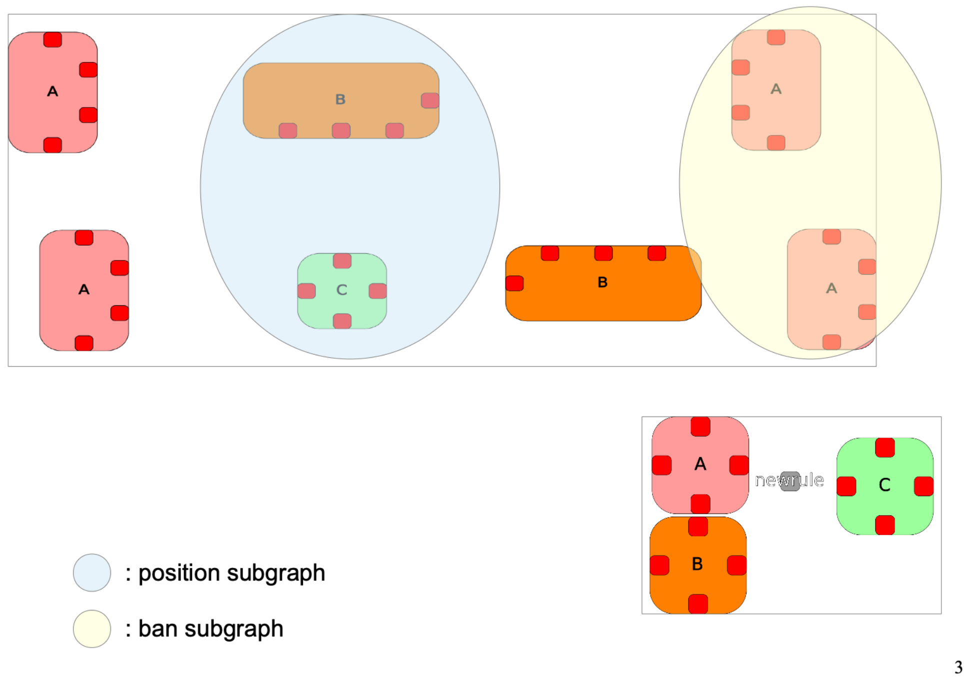
Choose position : setPos, setBan, crtGraph, crtPos, crtBan, ngb …
Select nodes with a special attribute…

➢ How to select and to ban positions, how to trace redexes along rewriting steps

Located graph and rule : a way of tracing redexes

A **located port graph** G^{PQ}
has two subgraphs P and Q called resp.
position and ban subgraph.

P and Q specify resp. the position where rewriting should take place, and the subgraph where rewriting is forbidden. Modified by rule application.



A **located rewrite rule** $L^{W} \Rightarrow R^{MN}$ specifies
- two disjoint subgraphs M and N of R resp. used to update the position P and ban Q subgraphs.
- a subgraph W of L to specify which nodes are expected to be in the position subgraph P of G.

M,N,W optional : by default R, Ø, g(L)∩P =/= Ø

Strategic Port Graph Rewriting: an Interactive Modelling Framework

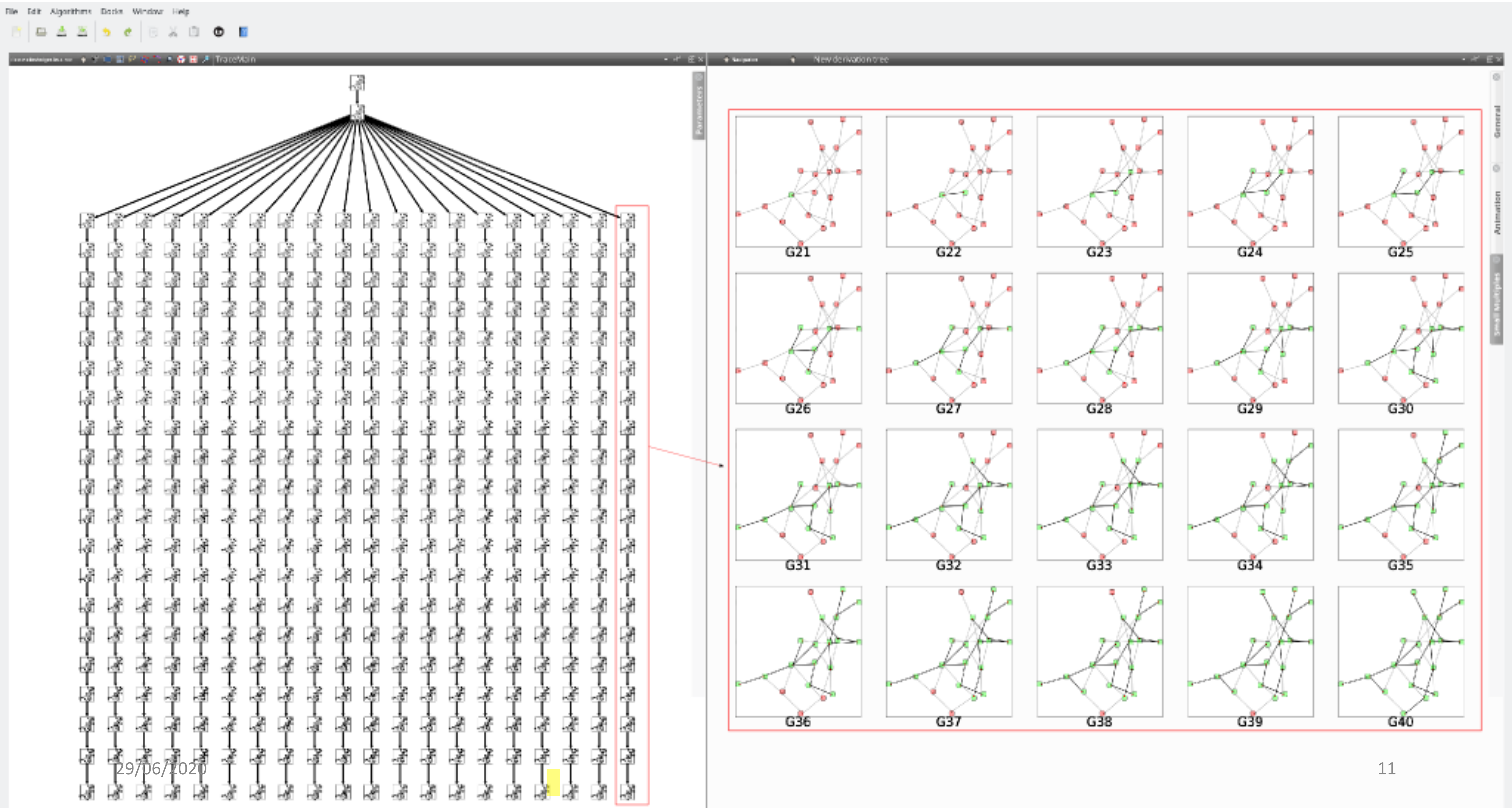
Main contributions [1]:

- Precise definitions of attributed (located) port graph, (located) rewrite rule and rewriting step

- A strategy language to formally express which rules to apply and where in the graph they may apply and are forbidden, including probabilistic primitives

- A definition of strategic graph programs [S, G^{PQ}] : a strategy and a located port graph together with examples

- A small-step operational semantics for strategic graph programs, specified by a transition system such that each strategic graph program is associated with a set of rewriting derivations, or traces, which can be represented as a derivation tree.


File
Edit
Algorithms
Docks
Window
Help
TraceMain
New derivation tree
Parameters
General
Animation
G21
G22
G23
G24
G25
G26
G27
G28
G29
G30
G31
G32
G33
G34
G35
G36
G37
G38
G39
G40

Summary

1. General presentation of the framework. An overview of Porgy with
   Concepts (port graphs, rewrite rules, strategies)
   A strategy language for computation control: *how to accurately control positions ?*

2. **Application to Biological systems**
   How to handle uncertain information ?
   Illustrate data analysis features

3. Application to Social networks
   How to compare and prove different models ?

4. Labeled graphs as a uniform data structure : *rules as labeled graphs*

5. To go further
   - Multilayer graphs
   - Semantics
   - Proof environment

Strategy-Driven Exploration for Rule-Based Models of Biochemical Systems with Porgy

Contributions in [2]

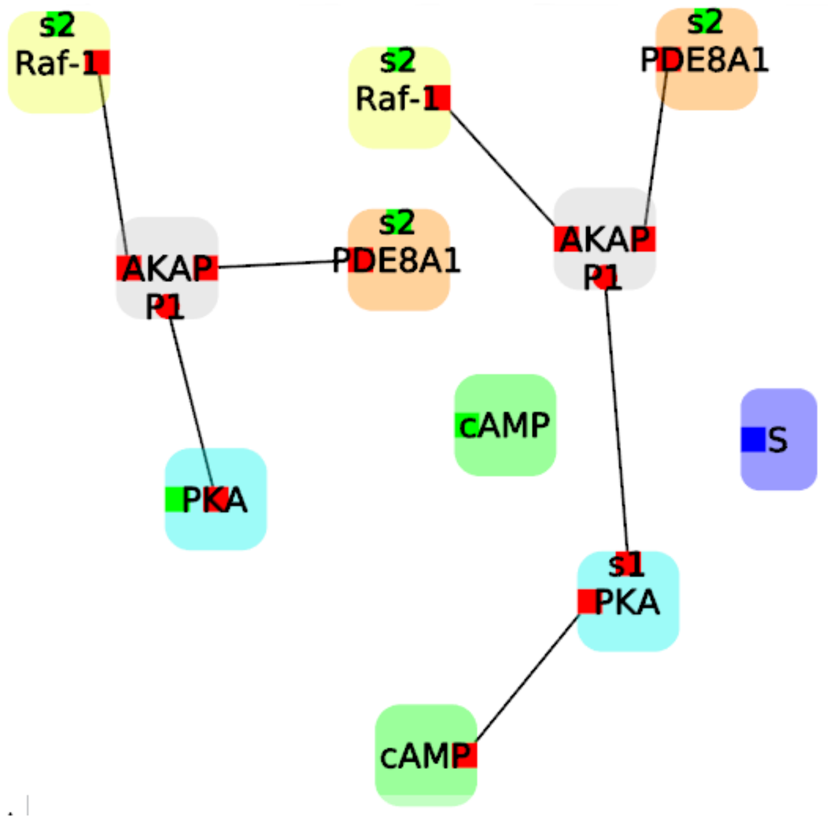
- Rule-based modeling techniques have been successfully applied in this area: a **biological pathway** is a series of interactions among molecules that leads to a specific product or a change in a cell. The state of the biochemical network at a given point in time is subject to transformations formalized by a set of conditions and rules. This methodology is supported by software tools as BioNetGen, RuleBender, Mosbie, Kappa, Maude…

- In [2], port graphs model the states, port graph rewrite rules describe the evolution of the system and strategies define explicit control over this evolution.

- Example developed in [2] of the RAF/MEK/ERK signalling pathway - that plays an important role in cell growth, prevention of apoptosis (programmed cell death), and drug resistance in cells.

Strategy-Driven Exploration for Rule-Based Models of Biochemical Systems with Porgy

The molecular species of our AKAP models are the following:
– scaffold protein AKAP with three binding sites, s1, s2, and s3
– enzyme RAF with two sites: the site s1 bound to AKAP's site s2 and the phosphorylation site s2
– enzyme PDE with one site s1 for binding to the scaffold's site s3 and one phosphorylation site s2 (PDE is more active when the site s2 is phosphorylated)
– nucleotide AMP with one binding site s1
– protein PKA with one site s1 bound to AKAP's site s1 and one site s2 for binding to AMP's site s1 (PKA is active when bound to AMP)
– activation signal S, an artificially introduced entity whose role is to signal when PKA is de-activated and AMP degraded



Strategy-Driven Exploration for Rule-Based Models of Biochemical Systems with Porgy

- Main expectations of biologists and Porgy's advantages:

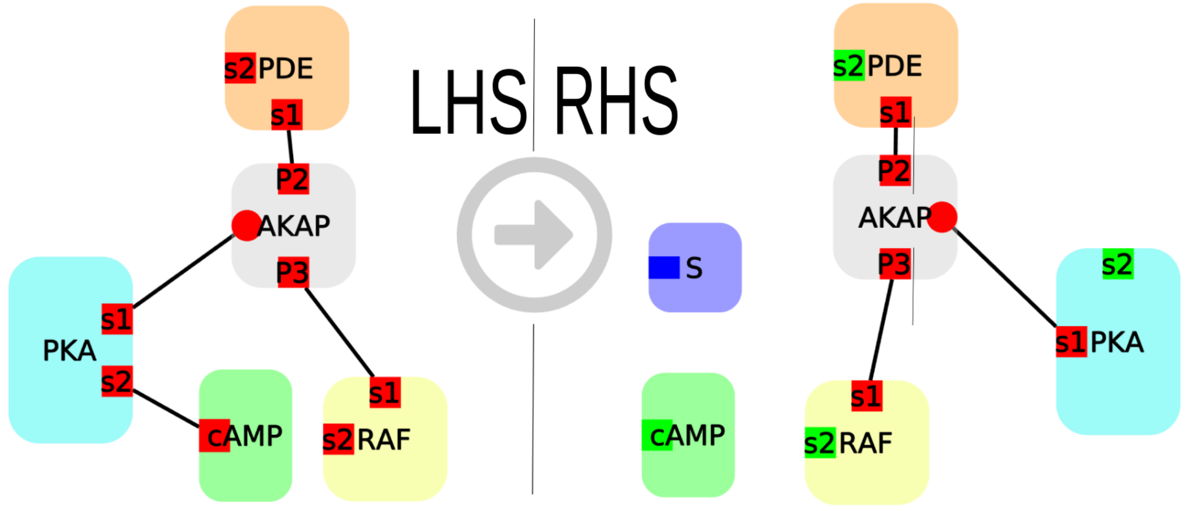
- Tools to facilitate the analysis of the system's evolution.
  - Get access to all traces (i.e., sequences of transformation steps)
  - Origin tracking
  - Plotting signal molecules

- Deal with uncertain information:
  - priorities over rule applications when exact reaction rates are unknown
  - experiments with different strategies
  - uncertainties formalized with probabilities and random choice

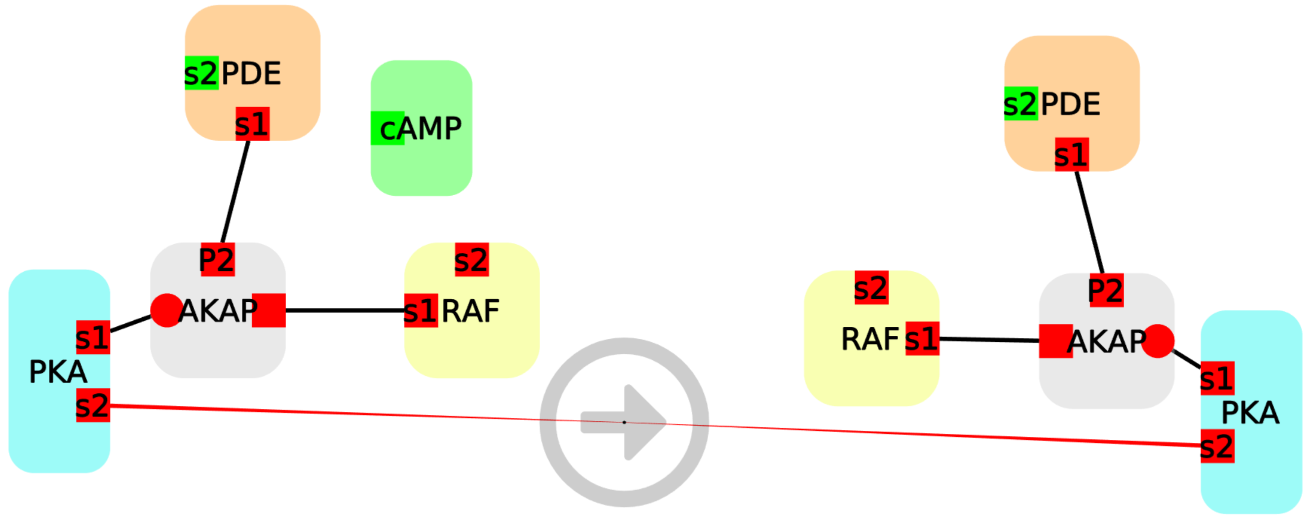
In the current implementation, the *ppick* command (random choice) has two options to specify probabilities

- Either a constant probability is associated to each rule
- Or a probability distribution is associated to a set of rules.

R6:



R10:

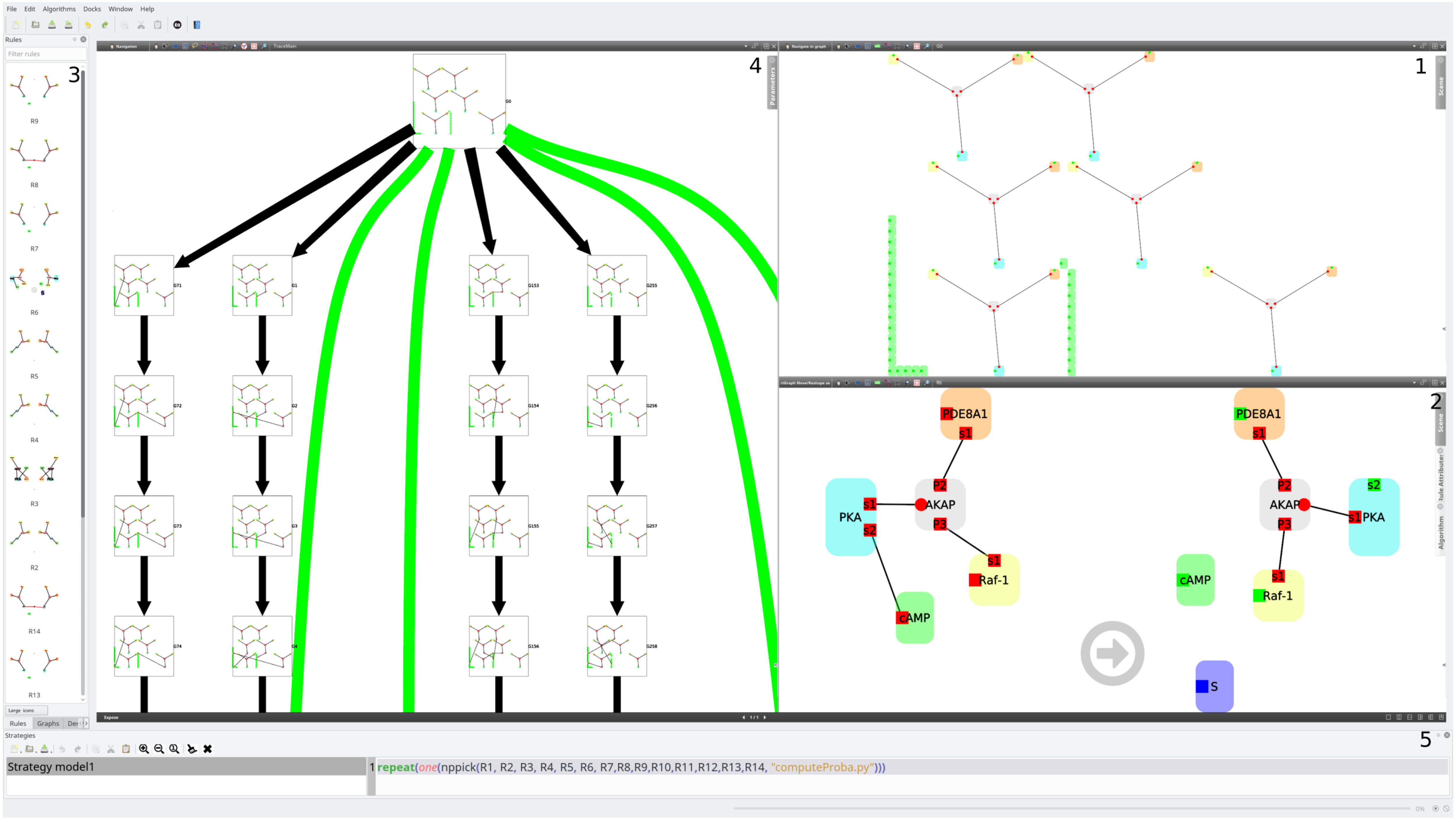


First strategy

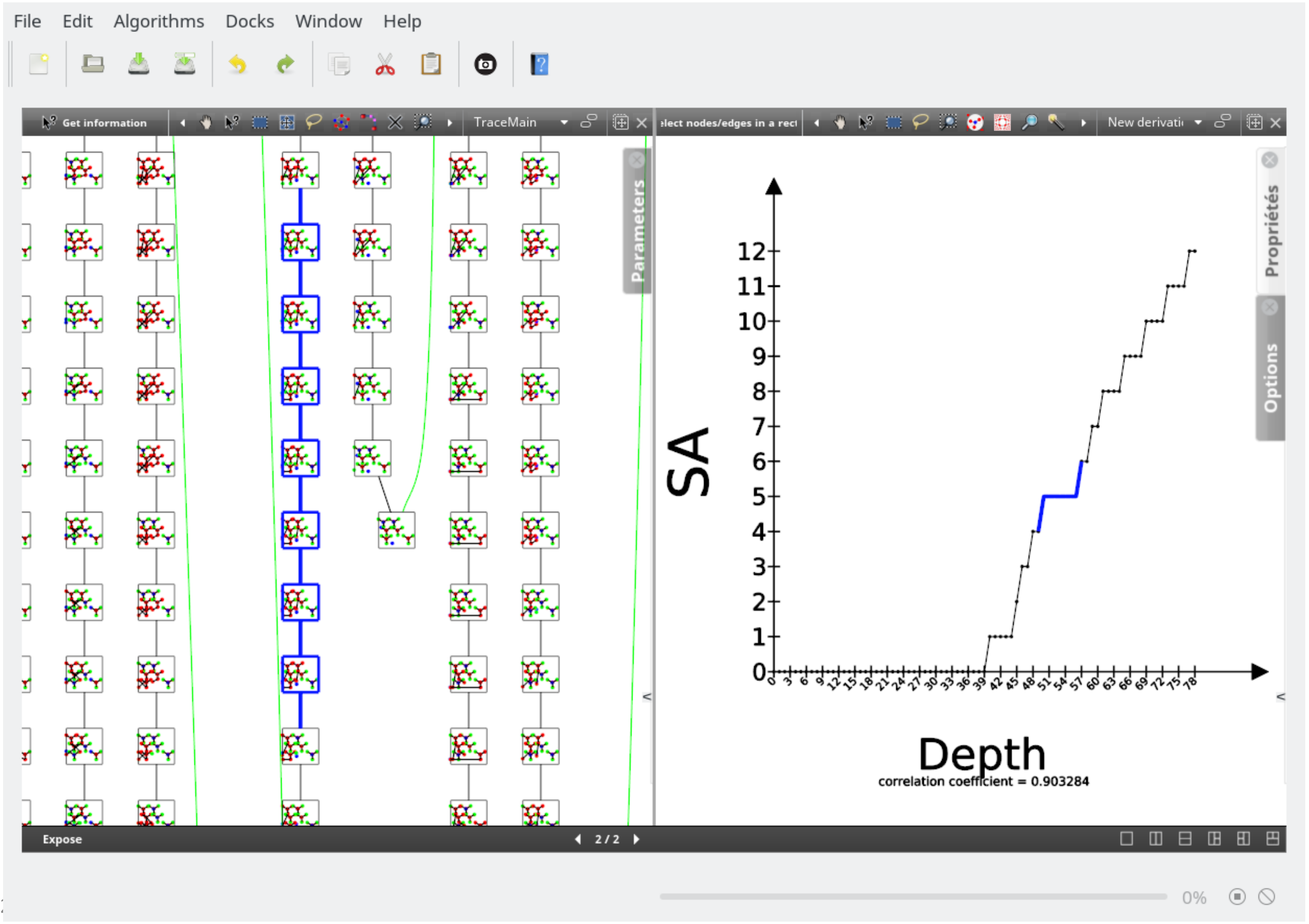
repeat(one(
ppick(R1, R2, R3, R4, R5, R6, R7, R8,
R9, R10, R11, R12, R13, R14,“ComputeProba.py”)
))

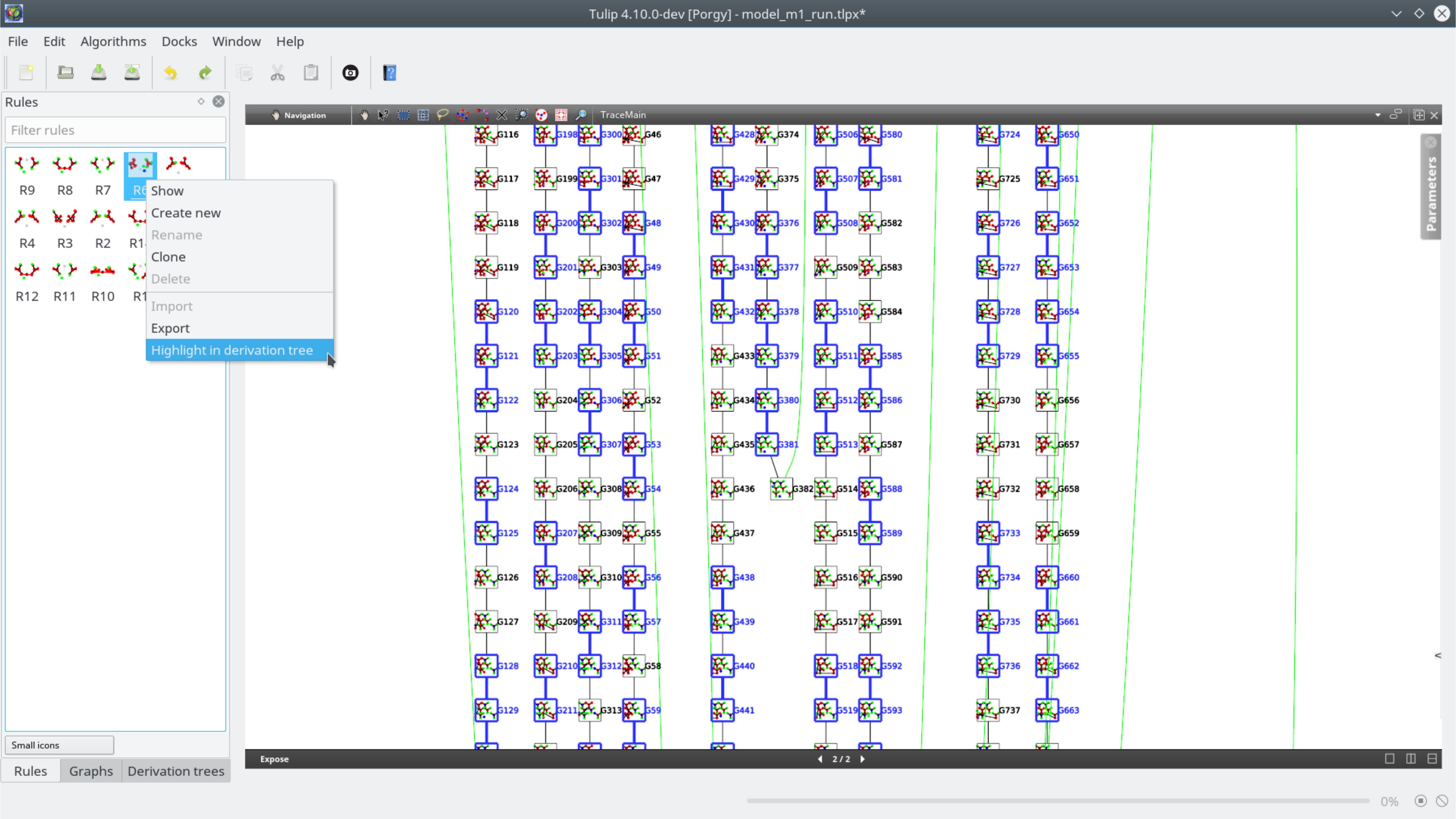
Updated strategy where the degradation process (rule R6) occurs after other rules applications.

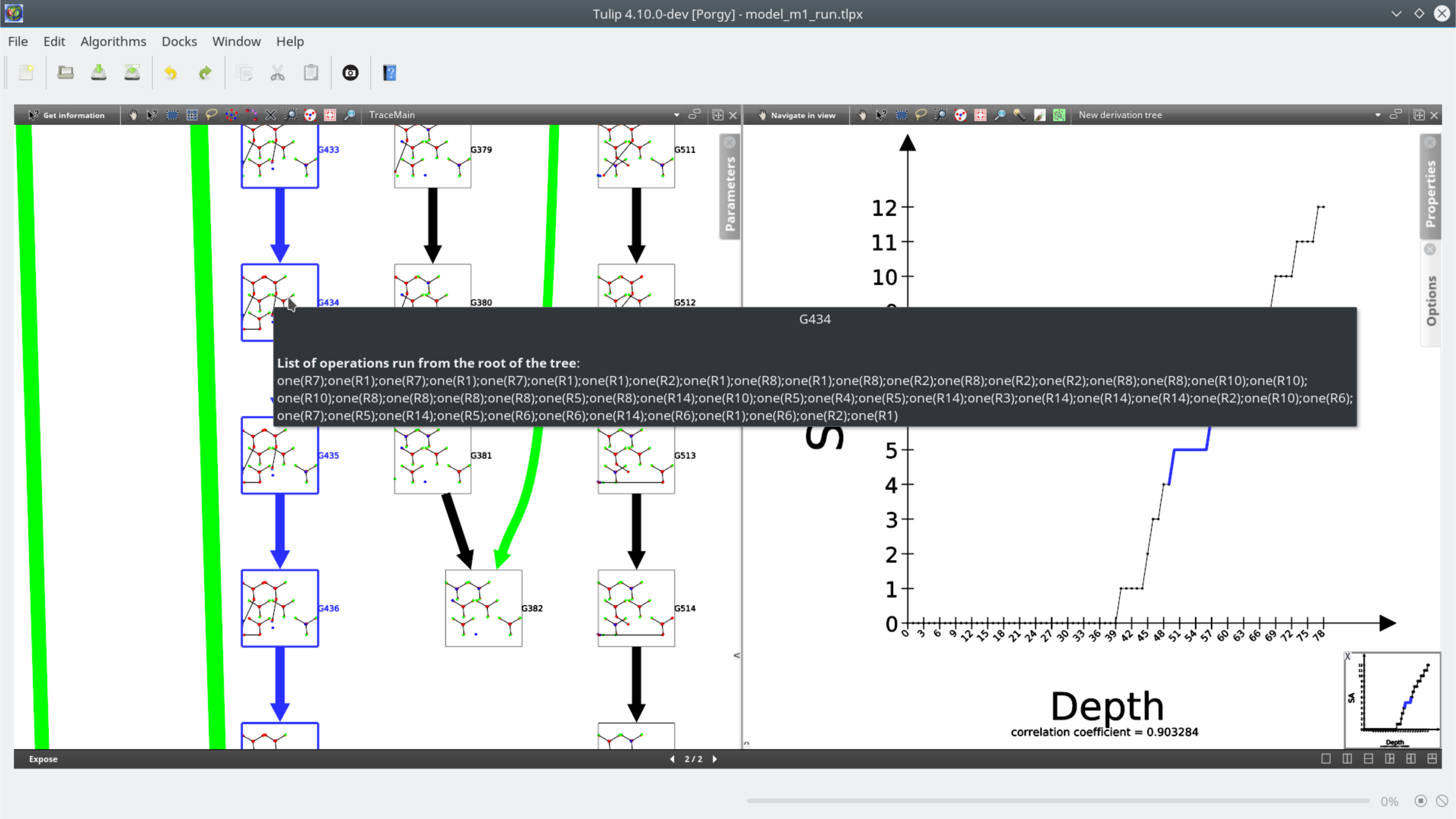
repeat(one(
ppick(R1, R2, R3, R4, R5, R7, R8,
R9, R10, R11, R12, R13, R14,“ComputeProba.py”)
);
try(one(R6)))



```
repeat(one(nppick(R1, R2, R3, R4, R5, R6, R7,R8,R9,R10,R11,R12,R13,R14, "computeProba.py")))
```


File
Edit
Algorithms
Docks
Window
Help
Get information
TraceMain
Parameters
elect nodes/edges in a rect
New derivati
Propriétés
Options
SA
Depth
0
1
2
3
4
5
6
7
8
9
10
11
12
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78
correlation coefficient = 0.903284
Expose
2 / 2
0%


Tulip 4.10.0-dev [Porgy] - model_m1_run.tlpx*
File
Edit
Algorithms
Docks
Window
Help
Rules
Filter rules
R9
R8
R7
R6
R4
R3
R2
R1
R12
R11
R10
R1
Show
Create new
Rename
Clone
Delete
Import
Export
Highlight in derivation tree
Small icons
Rules
Graphs
Derivation trees
Navigation
TraceMain
Parameters
Expose
2 / 2
0%
G116 G198 G300 G46 G428 G374 G506 G580 G724 G650
G117 G199 G301 G47 G429 G375 G507 G581 G725 G651
G118 G200 G302 G48 G430 G376 G508 G582 G726 G652
G119 G201 G303 G49 G431 G377 G509 G583 G727 G653
G120 G202 G304 G50 G432 G378 G510 G584 G728 G654
G121 G203 G305 G51 G433 G379 G511 G585 G729 G655
G122 G204 G306 G52 G434 G380 G512 G586 G730 G656
G123 G205 G307 G53 G435 G381 G513 G587 G731 G657
G124 G206 G308 G54 G436 G382 G514 G588 G732 G658
G125 G207 G309 G55 G437 G515 G589 G733 G659
G126 G208 G310 G56 G438 G516 G590 G734 G660
G127 G209 G311 G57 G439 G517 G591 G735 G661
G128 G210 G312 G58 G440 G518 G592 G736 G662
G129 G211 G313 G59 G441 G519 G593 G737 G663



G434

List of operations run from the root of the tree:
one(R7);one(R1);one(R7);one(R1);one(R7);one(R1);one(R1);one(R2);one(R1);one(R8);one(R1);one(R8);one(R2);one(R8);one(R2);one(R2);one(R8);one(R8);one(R10);one(R10);
one(R10);one(R8);one(R8);one(R8);one(R8);one(R5);one(R8);one(R14);one(R10);one(R5);one(R4);one(R5);one(R14);one(R3);one(R14);one(R14);one(R14);one(R2);one(R10);one(R6);
one(R7);one(R5);one(R14);one(R5);one(R6);one(R6);one(R14);one(R6);one(R1);one(R6);one(R2);one(R1)

correlation coefficient = 0.903284

Summary

1. General presentation of the framework. An overview of Porgy with
   Concepts (port graphs, rewrite rules, strategies)
   A strategy language for computation control: *how to accurately control positions ?*

2. Application to Biological systems
   How to handle uncertain information ?
   Illustrate data analysis features

3. **Application to Social networks**
   How to compare and prove different models ?

4. Labeled graphs as a uniform data structure : *rules as labeled graphs*

5. To go further
   - Multilayer graphs
   - Semantics
   - Proof environment

Labeled Graph Strategic Rewriting for Social Networks

Contributions in [3]

An algebraic approach, based on labeled graph strategic rewriting,
for the study of social networks, specifically network generation and propagation mechanisms.

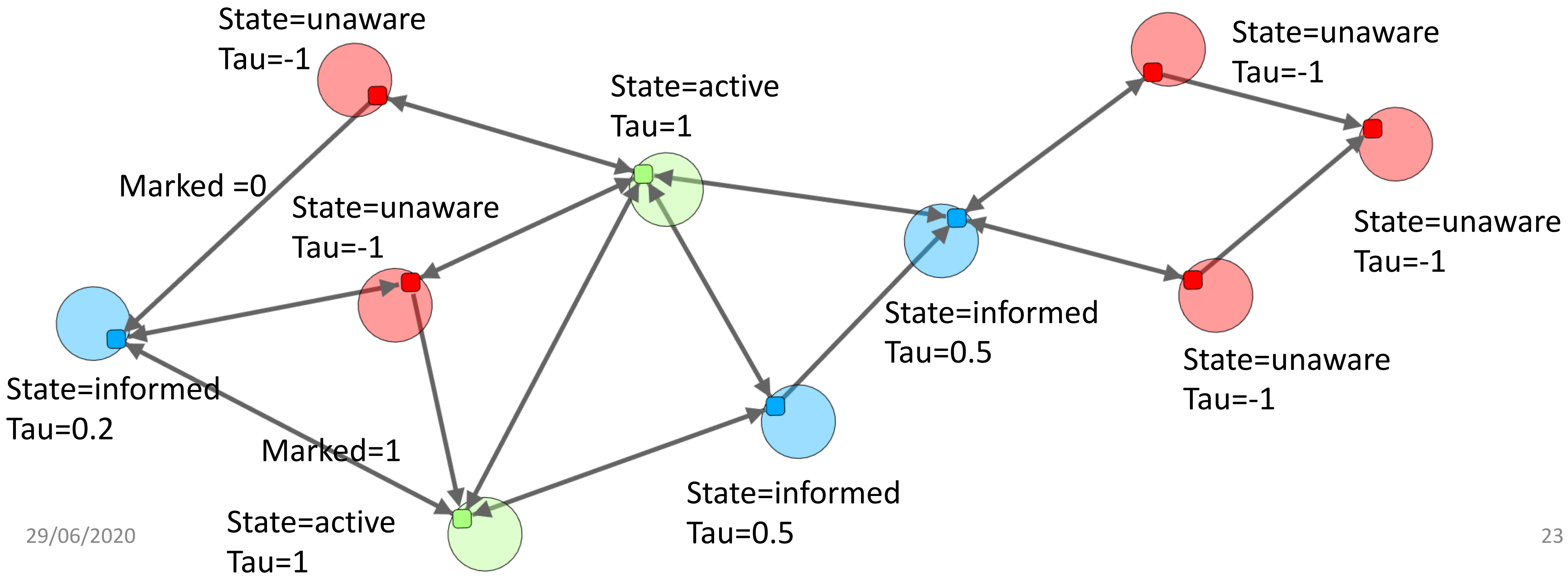
These techniques have been used

- to generate random networks that are suitable for social network analysis (small world),
- to simulate different propagation mechanisms, and
- to analyse and compare propagation models by extracting common rules and differences,

thus leading to improved algorithms.

A toy social network for propagation models

Users perform a specific action (such as relaying information, sharing a video clip…), thus becoming **active**. Their **unaware** neighbours are then **informed** and, when they are sufficiently **influenced**, become active themselves while performing the same action.
The process then reiterates and propagates the activation throughout the whole network.



Labeled Graph Strategic Rewriting for Social Networks

In [3], we considered three different propagation algorithms:

- ✓ the independent cascade model IC (with probabilistic influence)
- ✓ the linear threshold model LT (with neighbours' combined influence)
- ✓ the Riposte model RP, where the propagation process ensures that users' opinions are not disclosed to observers: being active (endorsing information) is different from disseminating information.

- The specification using strategic rewriting shows that the two models IC and LT are instances of the same strategic graph program with different attributes

- The strategic rewriting definition of RP can be seen as an improved version of IC with more rules, different attributes and strategies

- We proposed a new propagation model: a privacy-aware version of LT.

Propagation process IC

N_k be the set of active nodes at step k, and ξ_k be the set of ordered pairs (n,n′) subjected to a propagation from n (active) towards n′ (inactive).

Properties which must be satisfied at each step k where an active node n is selected:

IC.1 : n is given a single chance to activate each inactive neighbour n’

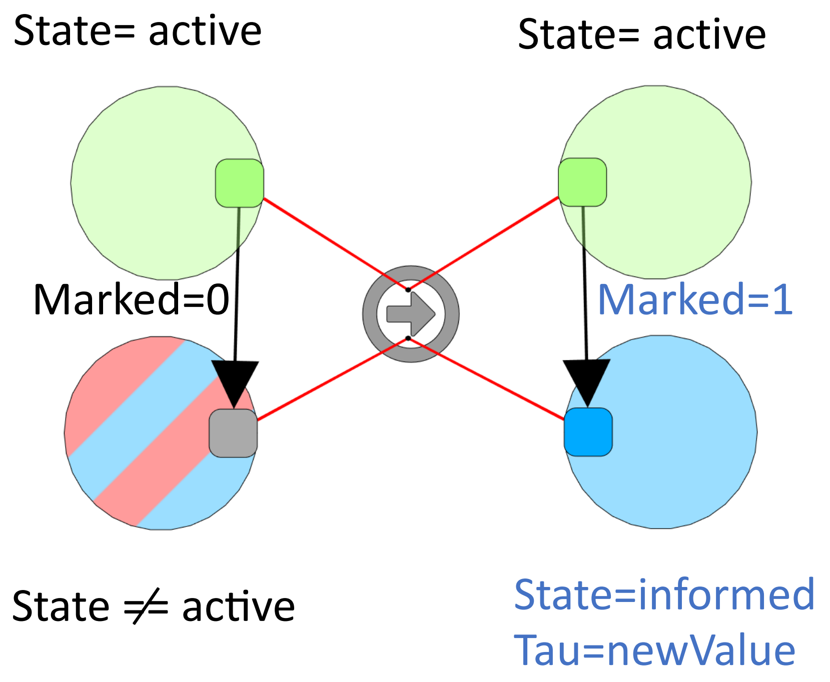
IC.2 : n succeeds in activating n’ with a probability $p_{n,n'}$

IC.3 : attempts of n to activate its inactive neighbours are performed in arbitrary order

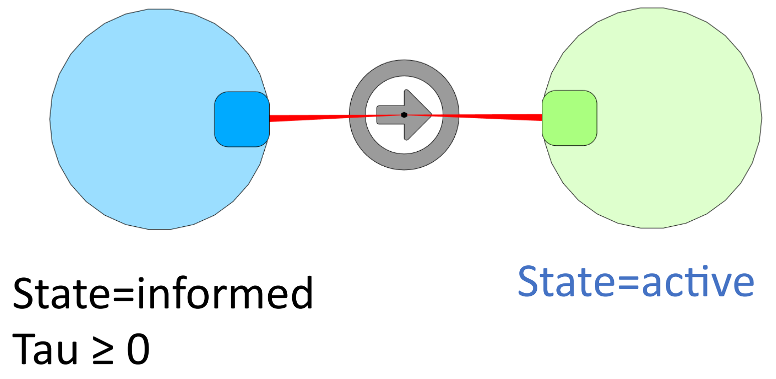
IC.4 : if n succeeds in activating n’ at step k, n’ must be considered as an active node in step k+1

IC.5 : the process ends if no more activations are possible.

Rules used to express the Independent Cascade model



(a) IC influence trial: influence from an active neighbour on an inactive node (either unaware or just informed).



(b) IC activate: an informed node becomes active
when sufficiently influenced.

Strategy for IC propagation

```
Strategy IC

repeat(
    setPos(all(property(crtGraph, node, State == active)));
    one(IC influence trial);
    try(one(IC activate))
     )
```

Theorem

The strategic rewrite program given by the rules IC influence trial and IC activate, under Strategy IC terminates and correctly implements the IC model.

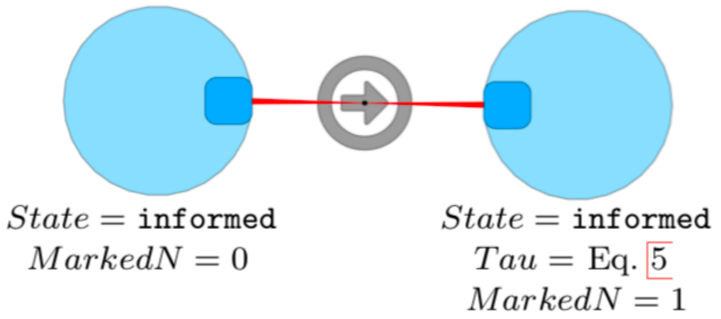
Proof based on

- Correctness of rules w.r.t. mathematical model (properties IC1-IC5)
- Termination taking into account the strategy: the semantics of repeat guarantees that if a command inside the body fails, the loop terminates.

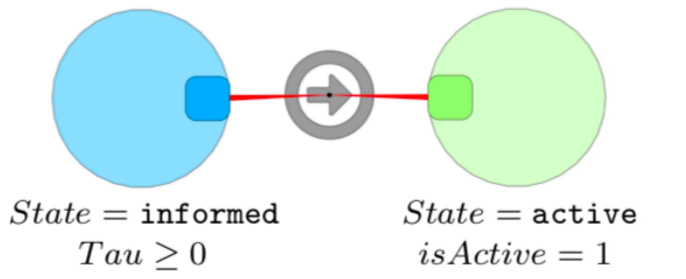
Rules for IC-Riposte

Being active different from sending information

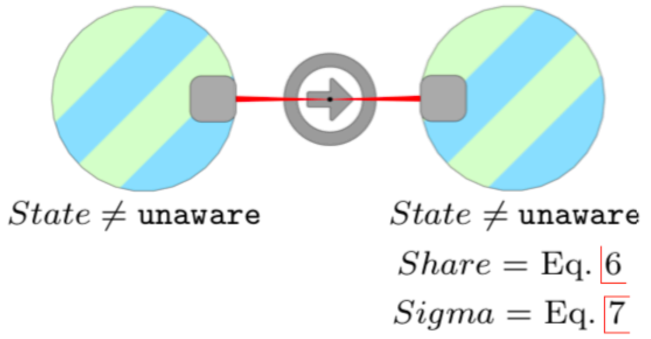
Same idea applied with LT: LT-Riposte



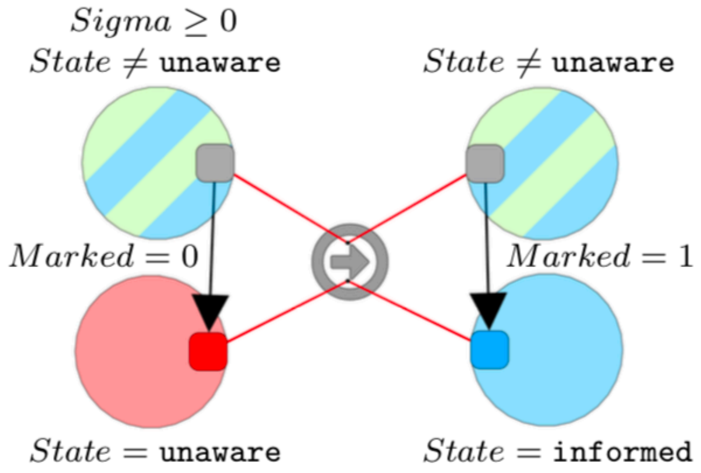
(a) *RP initialisation*: this rule is used to initialise an **informed** node aware of the information being spread.



(b) *RP activate*: an **informed** node becomes **active** if its *Tau* attribute is greater or equal to 0.



(c) *RP share trial*: whether a node is **active** or **informed**, the **RP** model can decide to use it to spread the information to others.


$Sigma \geq 0$
$State \neq \texttt{unaware}$
$State \neq \texttt{unaware}$
$Marked = 0$
$Marked = 1$
$State = \texttt{unaware}$
$State = \texttt{informed}$

(d) *RP inform*: a node aware of the information (**active** or **informed**), and selected to share its knowledge, informs an **unaware** neighbour.

Further topics for modeling social networks

Challenge: a drastic change of scale (millions of nodes and edges)

✓ size and complexity : needs to address
data storage capacities (similar to graph data bases)
subgraph matching algorithms –either exact or approximate– for finding one or all solutions
parallel or stochastic issues for matching and rewriting

✓ data structuring and visualization :
abstraction patterns for clusters
focusing on points of interests
hierarchies and views (for instance, through multi-layer graphs).
All these notions need a precise and logical definition

Orthogonal questions: adequation of model to real data, compliance to properties like privacy preserving

Summary

1. General presentation of the framework. An overview of Porgy with
   Concepts (port graphs, rewrite rules, strategies)
   A strategy language for computation control: *how to accurately control positions ?*

2. Application to Biological systems
   How to handle uncertain information ?
   Illustrate data analysis features

3. Application to Social networks
   How to compare and prove different models ?

4. **Labeled graphs as a uniform data structure :** ***rules as labeled graphs***

5. To go further
   - Multilayer graphs
   - Semantics
   - Proof environment

Labeled Port Graph – A Formal Structure for Models and Computations

Contribution in [4]

All ingredients of a graph transformation system can be specified as labeled port graphs.

- ✓ Port graph rewrite rules with a special node (the arrow node)
- ✓ Located graphs with labels to specify the rewriting position and banned positions
- ✓ Rewriting derivations whose nodes are labelled by graphs and edges by rules/strategies.

A framework whose all components are represented using a unique concept (namely, labeled port graphs), has advantages both from a theoretical and a practical point of view:

- there is one main data structure to design and implement, so the implementation efforts can focus on this task,
- since the rewrite rules are themselves graphs, the formalism is by construction reflective.

Reflection is a key property in logical frameworks and facilitates the design of extensions (see, e.g., rewriting logic).

Labeled port graph rewrite rule

A labeled port graph rewrite rule L ⇒ R is a graph with an **arrow node** with

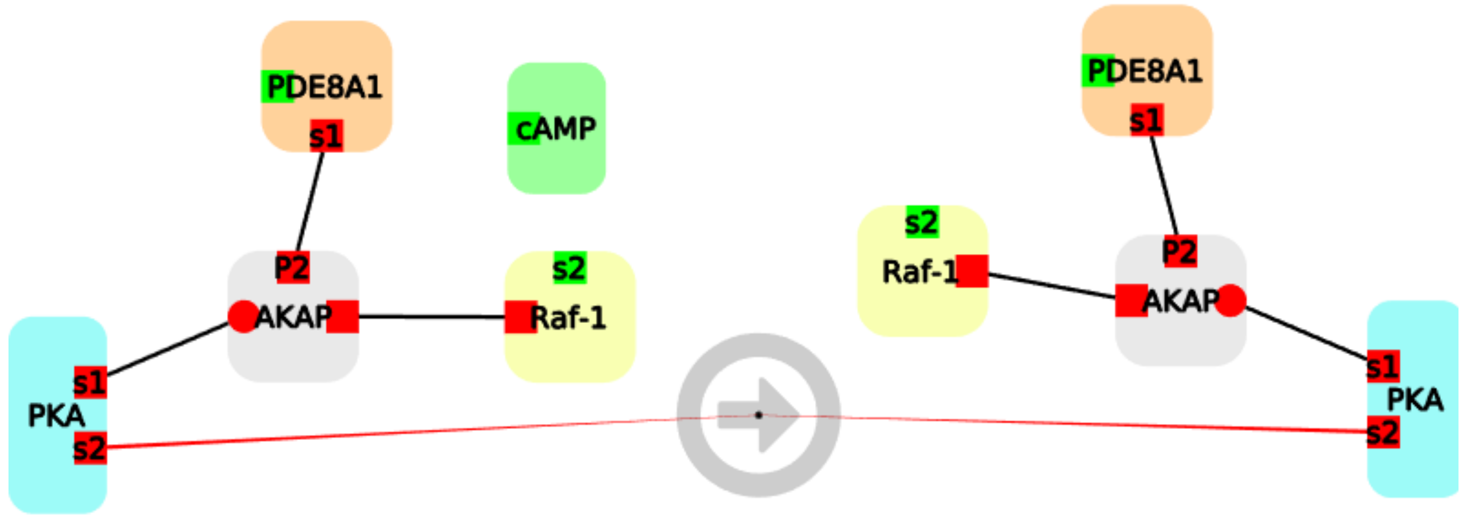
- Two implicit ports, labeled Left and Right, and edges connecting Left to all nodes in L and Right to all nodes in R. *Separate left and right-hand sides*

- A set of **rewiring ports** and edges connecting them to ports in L or R.

Each port has a label Type that can have one of three different values: **bridge, wire and blackhole.** The value indicates how a rewriting step using this rule should affect the edges that connect the redex to the rest of the graph. *Guide rewiring dangling edges*

(1) A port of type bridge has edges connecting it to L and to R (one to L and one or more to R): it thus connects a port from L to ports in R.
(2) A port of type wire has exactly two edges connecting to L and no edge connecting to R.
(3) A port of type blackhole has edges connecting it only to L (one edge or more).

- Two predefined labels: *Express conditions*
  - **Where** that expresses a condition to trigger rule application,
  - **Saturated**, whose value is the list of ports in L that are not connected to a bridge, wire or blackhole port, also used as a condition to select morphism



This rule has red edges connecting the port s2 in PKA to a bridge port in the arrow node.
In the graph where this rule is applied, there could be other edges arriving to s2 from the outside.

However, no edges can be connected to AMP since its port is not connected to the arrow node: if an edge arrives to AMP from the outside the Saturated condition fails and the rule does not apply.

Summary

1. General presentation of the framework. An overview of Porgy with
   Concepts (port graphs, rewrite rules, strategies)
   A strategy language for computation control: *how to accurately control positions ?*

2. Application to Biological systems
   How to handle uncertain information ?
   Illustrate data analysis features

3. Application to Social networks
   How to compare and prove different models ?

4. Labeled graphs as a uniform data structure : *rules as labeled graphs*

5. **To go further**
   - **Multilayer graphs**
   - **Semantics**
   - **Proof environment**

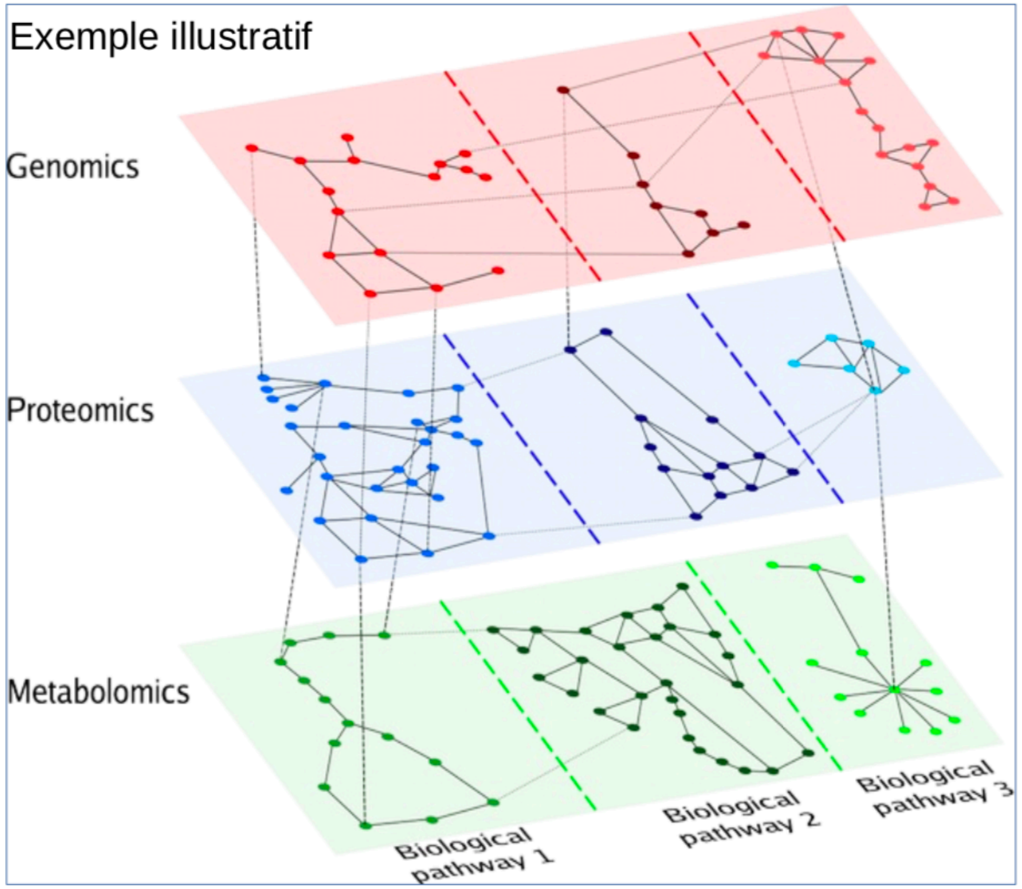
Multilayer Networks

A hierarchical view for analyzing complex systems: interactions are classified into layers according to their characteristics.

A **multilayer network** is a set of graphs (networks) with edges (connections) between nodes in different layers.
Layers represent features that characterize nodes and intra-layer edges
Inter-layer edges connect nodes in different layers.

In labeled graphs,
- a same layer label can be given to nodes, ports and edges to characterize this layer. The same set of attributes is shared at a given layer.
- different edge label can identify a relation between nodes or ports of layer A and layer B, possibly coupling two shared nodes or ports but with different sets of attributes in A and B.

Multilayer networks have been used already in ecology (how same plant species interact with pollinators and herbivores), in biology (molecular interactions between different human tissues), in transports (intermodal mobility in a city or region), human brain (time-varying connectivity at different scales), economy (interbank network).



The State of the Art in Multilayer Network Visualization.
F. Mcgee, M. Ghoniem, G. Melançon, B. Otjacques, B. Pinaud.
Computer Graphics Forum, Wiley, 2019

Semantics of port graph rewriting

- Definition of rewrite rule and rewriting step

[1] follows the single pushout approach (Ehrig et al., 1997b) and (Löwe et al., 1993)
for attributed graph structures:
a rewrite rule is a pair of graphs (the left and right-hand sides) with a partial morphism that relates them (specified via the arrow node).
Port graph morphisms take attributes and their values into account.

OK for a large class of port graph rewrite rules but not in full generality.

Explore (Corradini et al)'s approach to algebraic graph rewriting
and the PBPO (Pullback-Pushout) for attributed graph transformation.

Debugging and verification

Strategic graph programming raises many questions :

- termination analysis, confluence analysis, conflicting rules detection
- cycle detection, fairness analysis
- detection of unwanted patterns, error detection and correction
- correctness and reachability proofs

Already possible:

- use the derivation tree first to get an intuition: termination, cycle detection,
- detection of unwanted pattern (use a rule P=>fail)

To go further:

- Adapt Floyd-Hoare logic to the strategy language
- Exploit the derivation tree for reachability proofs and traces equivalence
- Memory and Backtracking : reason on strategies with backtracking and history

Long-term ambition : visual environment for debugging, verifying and certifying strategic graph programs

Thank you !   Questions ?