PORGY: A Visual Graph Rewriting Environment for Complex Systems

Bruno Pinaud, Guy Melançon, Jonathan Dubois

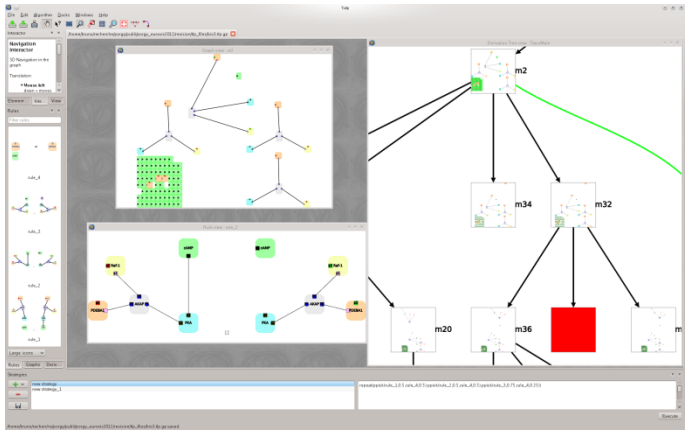
Visu 2012, 25/09/2012,
Telecom ParisTech

LaBRI
UNIVERSITÉ DE BORDEAUX
Inria
INVENTEURS DU MONDE NUMÉRIQUE

Modeling Complex Systems

Domain experts

Visualization



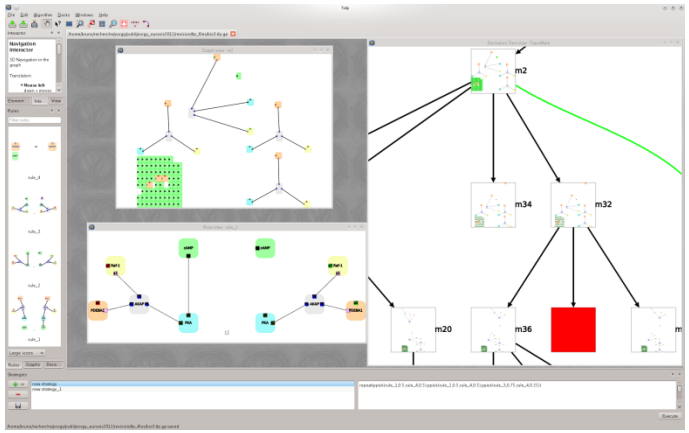
Modeling experts

Modeling Complex Systems

Biologists

See video 1

Visualization



Graph rewriting experts

Modeling Complex Systems

- Overall goal: understand role of pathway *interaction* in the regulation of cell proliferation, transformation and survival
- More specific goals:
  - Explain the overall dynamics using a small set of rules prescribing how molecules bind
  - Assess the validity of the model by showing how system parameters obey expected ‘laws’

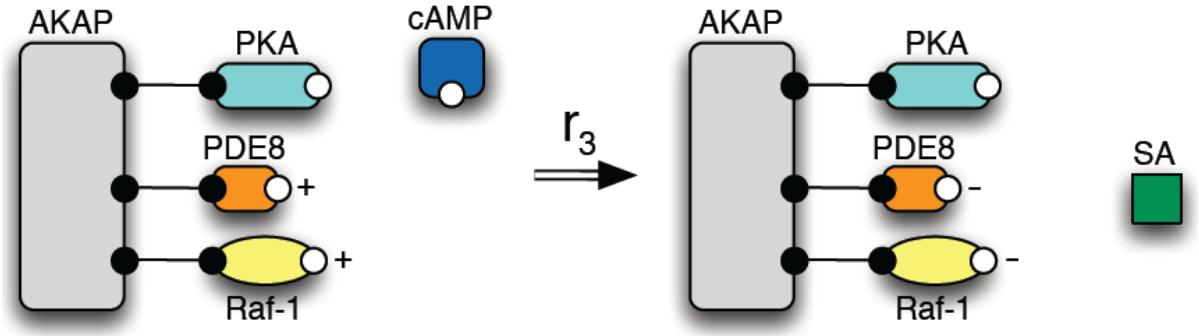
Modeling Complex Systems

- In simple terms: model design and validation can be seen as a « *what if* ? » game
  - *What if* we have this rule governing how molecule of type A interacts
  - *What if* each rule R only applies with probability p?
  - *What if* … ?

See video 2

Modeling Complex Systems

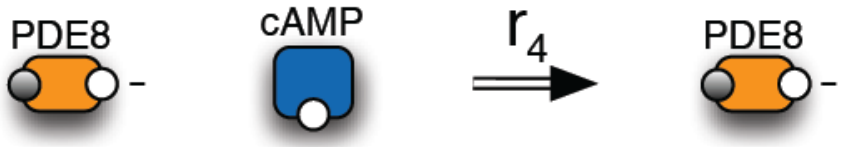
- The method:
  - Design simple binding rules



  - Run simulations, perform measurements
  - Adapt (change) rules
  - Keep track of past simulations

Modeling Complex Systems

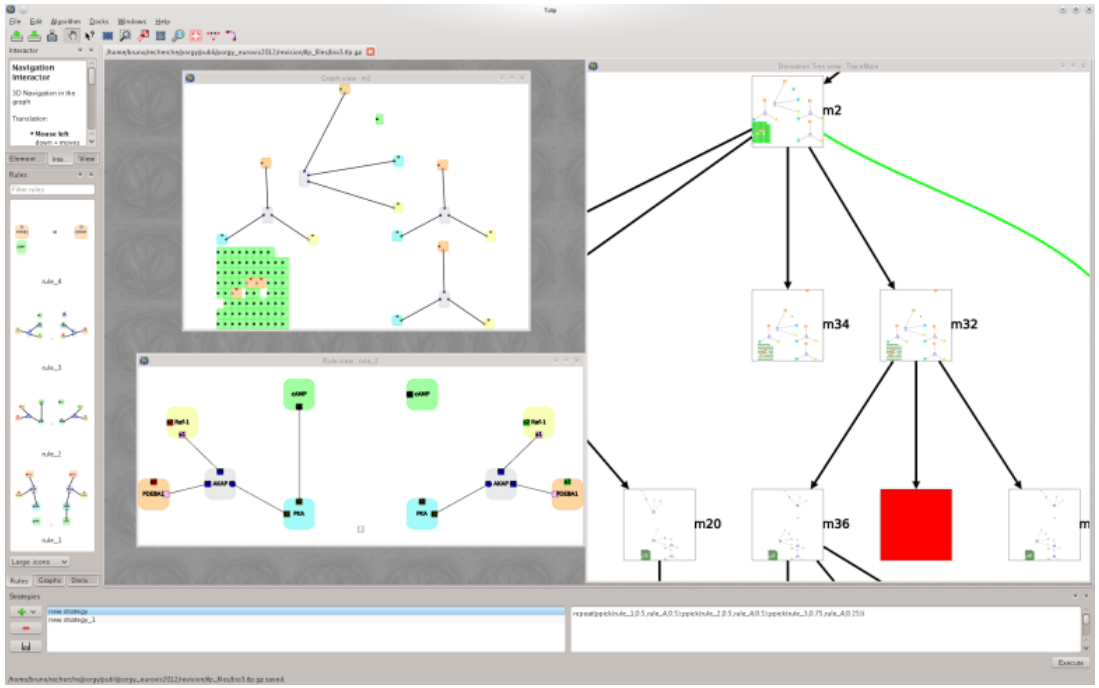
- The formalism:
  - Model molecules as graphs



  - Binding rules become local graph transformations
  - The whole thing becomes a Graph Rewriting System (GRS)

Modeling Complex Systems

- The challenge:
  - Design a visual environment to support rule-based modeling



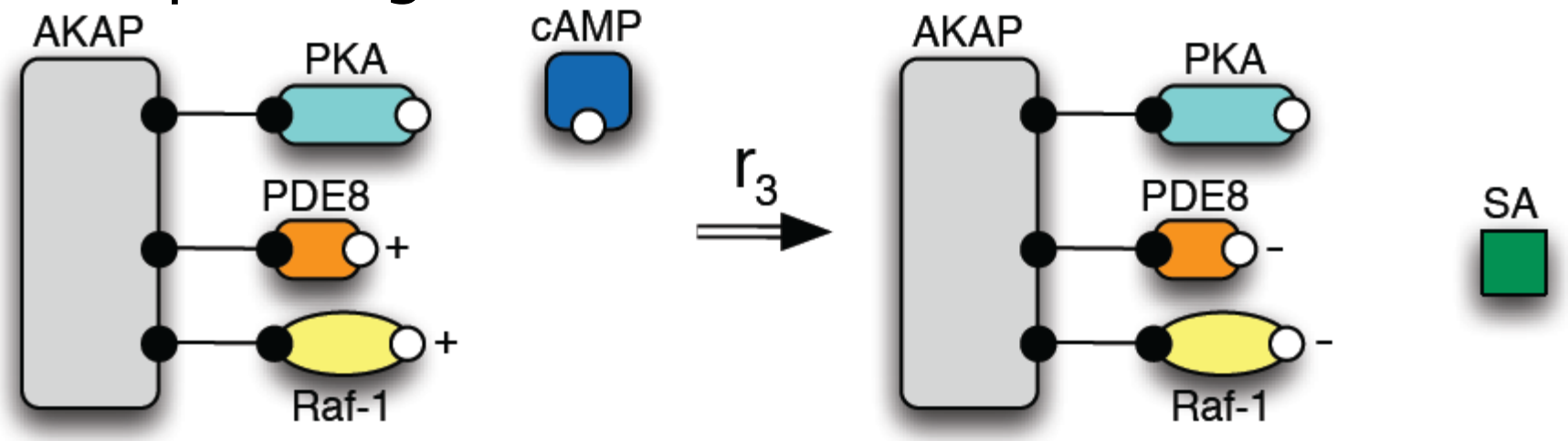
PORGY: Port Graph Rewriting

Graph rewriting systems

- GRS have a long history in theoretical computer science
  - Powerful graphical computation paradigm
  - GRS formal languages [Progress (Schür et al. 1997), Fujaba (Nickel et al. 2000), GP (Plump 2009), ...]
- Growing number of applications
  - Security policies, model driven software engineering, biomolecular interactions

Graph rewriting systems

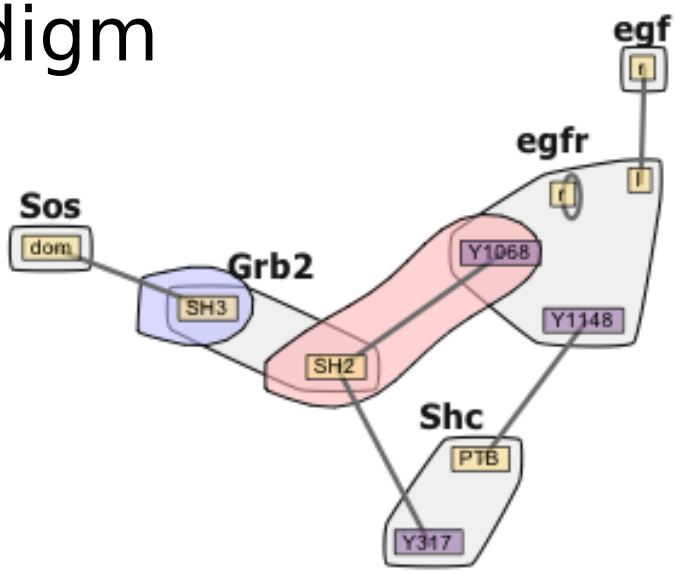
- GRS have a long history in theoretical computer science
  - Powerful graphical computation paradigm



Biomolecular interactions & calculus [Andrei , Calder, Kirchner 2008, 2010]

Graph rewriting systems

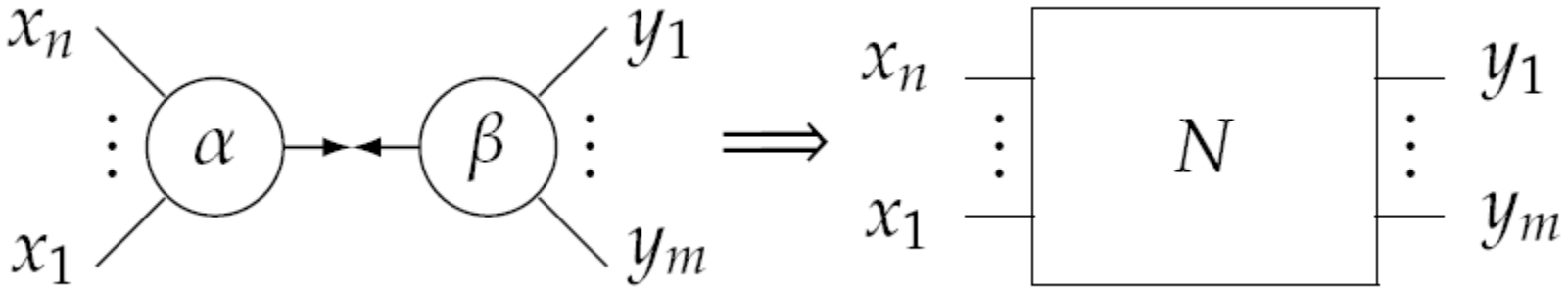
- GRS have a long history in theoretical computer science
  - Powerful graphical computation paradigm



Biomolecular interactions & calculus [RuleBender – Smith et al. 2011]

Graph rewriting systems

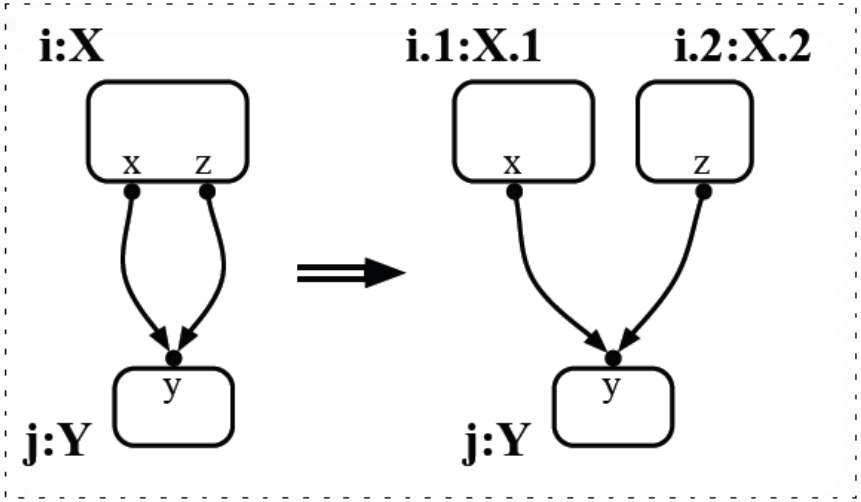
- GRS have a long history in theoretical computer science
  - Powerful graphical computation paradigm



Arithmetic calculus [Lafont 1990; Fernandez Mackie 1998]

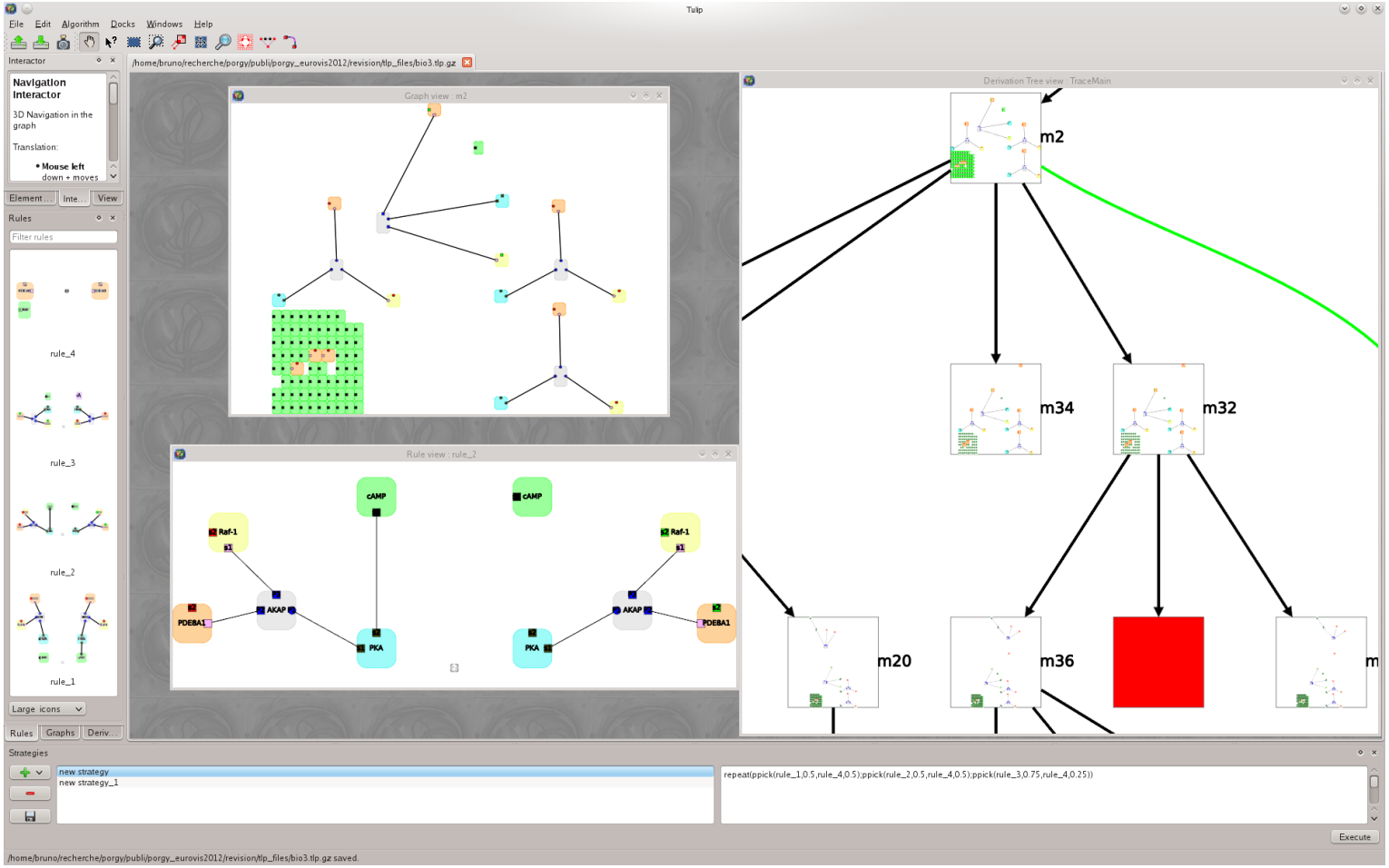
Graph rewriting systems

- GRS have a long history in theoretical computer science
  - Powerful graphical computation paradigm



Autonomous systems [Andrei, Kirchner 2009]

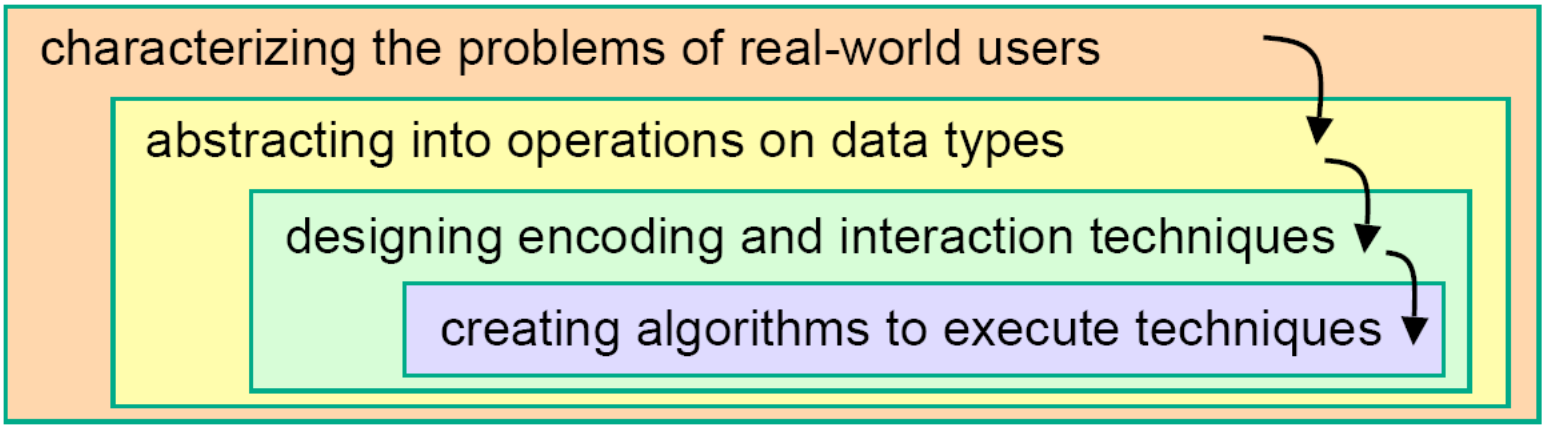
Supporting rule-based modeling



Visualization system design

- Adopt a methodology according to

T. MUNZNER: **A nested process model for visualization design and validation**.
IEEE Trans. on Visualization and Computer Graphics 15, 6 (2009), 921–928



3-year project

Biologists (O. Andrei), GRS experts (Fernandez, Kirchner)

Visualization system design

M System Modeling	G Graph Rewriting Questions	R Task Abstractions Design Requirements	A Visual / Interaction / Algorithms
M1 Design elementary molecular interactions	**G1** Design the ruleset and the initial graph G_0	**R1** Build/Edit G_0/rules	**A1** Support port graph editing
M2 Define an evolution scenario	**G2** Define a rewriting strategy	**R2** Build/Edit a rewriting strategy (*e.g.*, reg. exp.)	**A2** Drag and drop graphical entities between different views
M3 Observe / Study system evolution	**G3** Iterate rule applications **G4** Visualize their effects	**R3** Grab a strategy or rule and trigger a computation **R4** Replay rule applications	**A3** Compute subgraph isomorphism **A4** Show graph transformations (Animations / Small multiples)
M4 Keep track of computations / Allow backtracking to check, adjust and/or modify model	**G5** Check for convergence or termination/premature end of computation **G6** Eventually fix the ruleset	**R5** Trigger computations from any node of the derivation tree **R6** Show dead-end situations in the derivation tree	**A5** Build and layout derivation tree **A6** Tree drawing issues: grow branch from internal node, insure stability of layout
M5 Query for the presence of certain molecules at different stages **M6** Study behavior of given parameters as system evolves **M7** Alternate between parameter behavior and model	**G7** Local inspection/query of graph items (nodes, edges, subgraphs) **G8** Compute attributes or structural properties of graphs as rules are applied	**R7** Select (group of) nodes **R8** Test for the presence of identical groups of nodes **R9** Plot parameter evolution **R10** Synchronize the plot with the derivation tree **R11** Allow selection of graph items from the plot	**A7** Emphasize selected items wherever shown **A8** Manage subgraph hierarchies **A9** Compute subgraph isomorphism **A10** Synchronize underlying data structures (coordinated views)
M8 Study model computational / structural properties	**G9** Check for confluence of computation **G10** Inquire about structure underlying ruleset	**R12** Check whether graphs obtained from different rule sequences are the same **R13** Check how/where graphs are the same on an alternate view	**A11** Compute subgraph isomorphism **A12** Build quotient graph from equivalence relation (isomorphism) between graphs **A13** Maintain coherence with synchronized views

B. Pinaud, G. Melançon and J. Dubois, PORGY: A Visual Graph Rewriting Environment for Complex Systems, Eurovis 2012

Visualization system design

System modeling
Define elementary molecule interactions
Define an evolution scenario

GRS questions
Define *rhs/lhs* subgraphs
Define a *rewriting strategy* [Fernandez, Kirchner, Namet]

- System modeling tasks/questions correspond to ‘pure’ GRS questions
  - Therefore indicating that genericity is achievable

Visualization system design

System modeling
Heading towards model validation
Query for the presence of molecules
Study model parameters

GRS questions
Iterate rule applications
Local inspection of graph items
Compute graph structural properties (metrics)

Visualization system design

Task requirements	Visual / interaction / Algorithms
Build/Edit rules or graphs, rewriting strategy	Graph editing
Trigger computations	Show graph transformations
Show dead-end situations	Drag & drop entites between views
Selection	Subgraph isomorphism heuristics
View synchronization	

More involved tasks

System modeling
Keep track of computations Allow backtracking to check, adjust and/or modify model
Study model computational / structural properties

GRS questions
Check for convergence or termination / premature end of computation Eventually fix the ruleset
Check for confluence of Computation
Inquire about structure of underlying ruleset

Keeping track of computation

System modeling

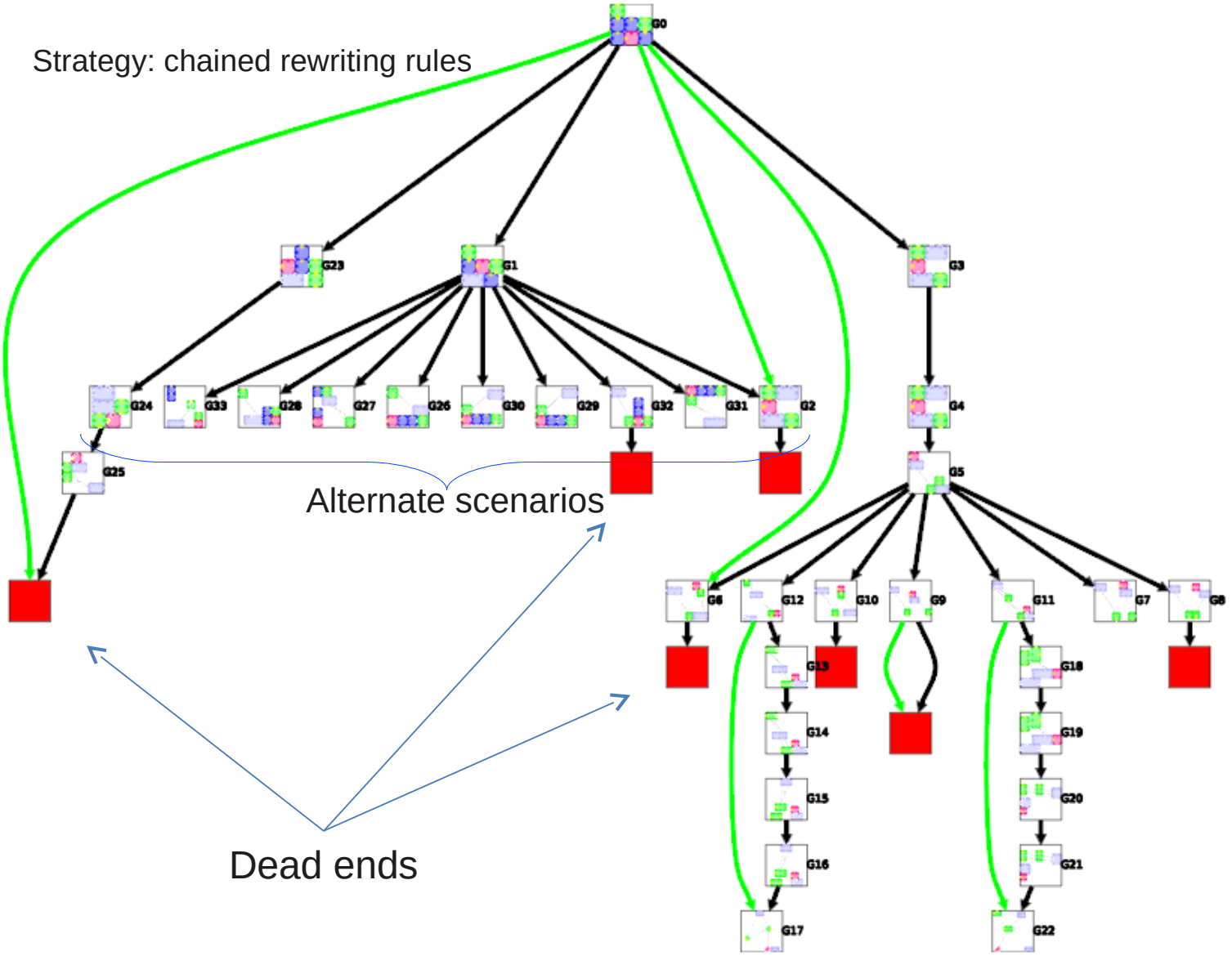
Keep track of computations
Allow backtracking to check, adjust and/or modify model

GRS questions

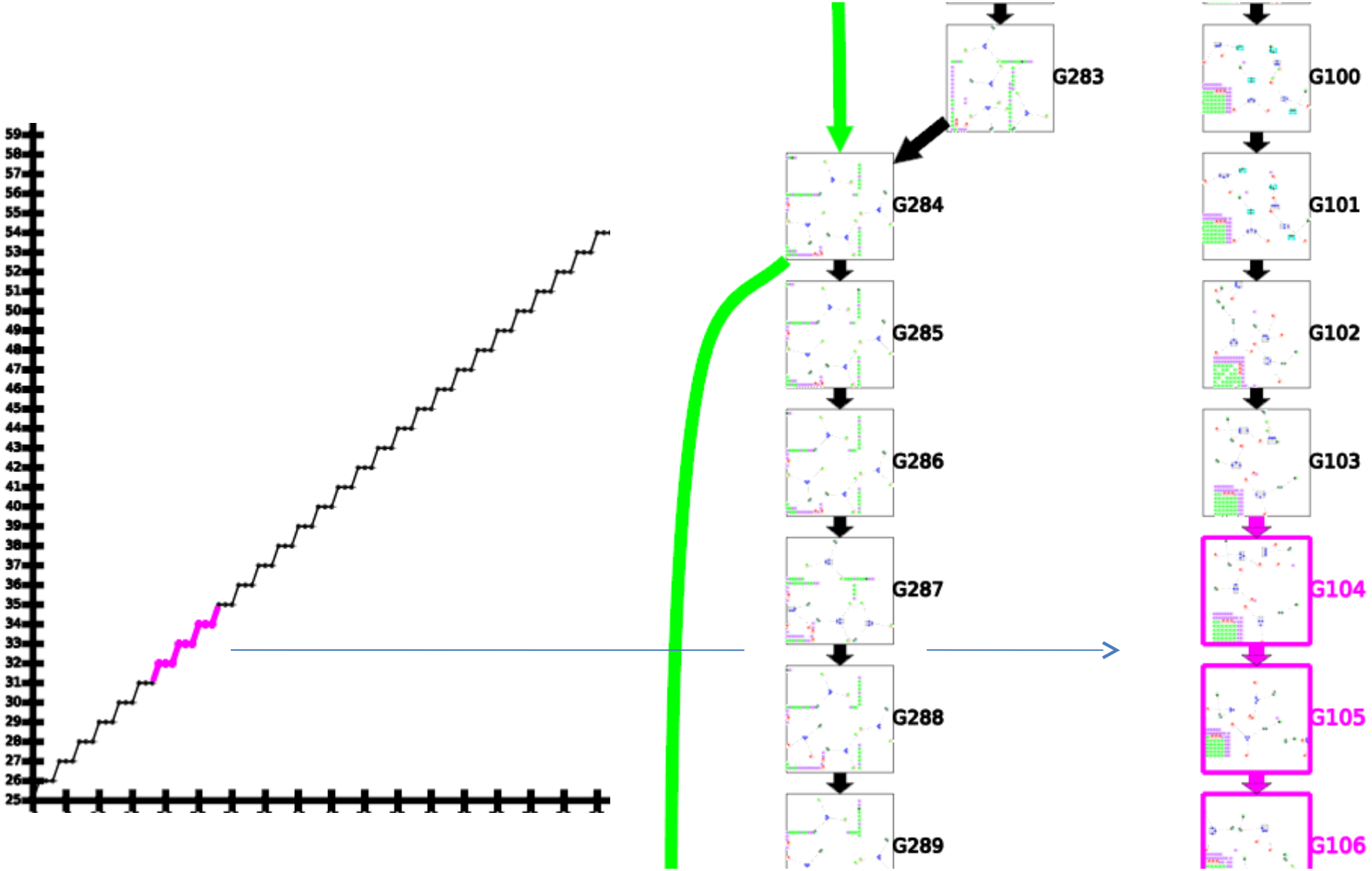
Check for *convergence* or termination / premature end of computation
Eventually fix the ruleset

- Derivation *tree* as a history mechanism and data structure
  - Rule application is *non-deterministic*

Keeping track of computation



Track model parameter



More involved tasks

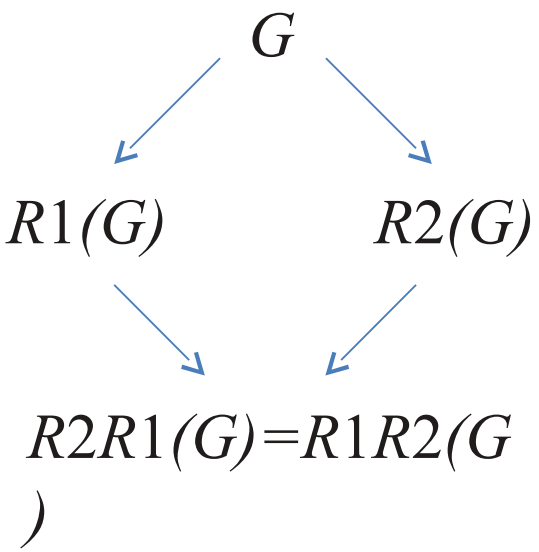
System modeling

Study model computational / structural properties

Inquire about structure underlying ruleset

Check for *confluence* of Computation

Inquire about structure of underlying ruleset



- *Confluence* means rules ‘commute’

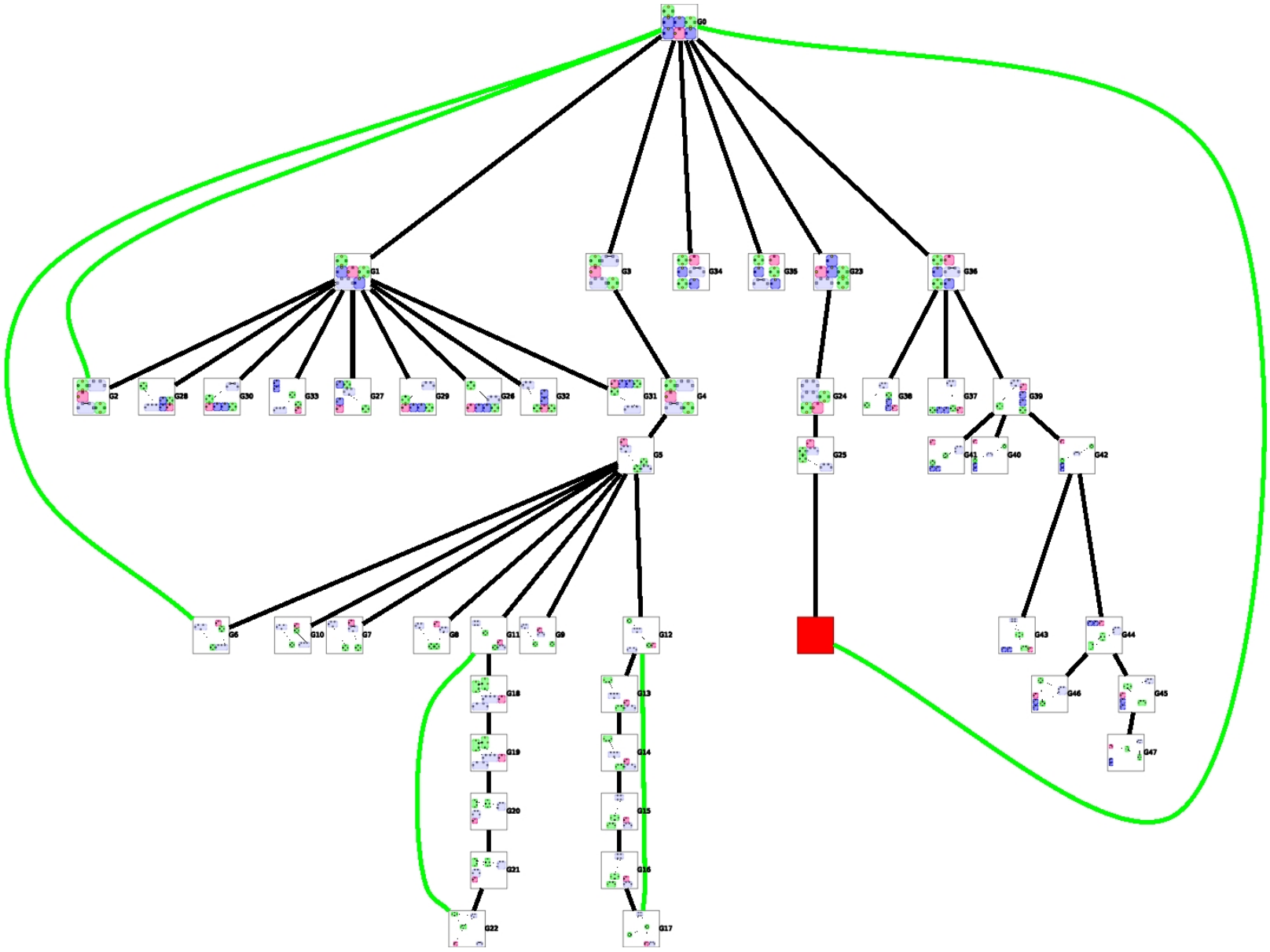
Confluence

- *Confluence* is heavy duty stuff – both conceptually and computationally
  - When combined with convergence it means « All roads lead to Rome »
  - Testing confluence requires identifying isomorphic copies of a graph

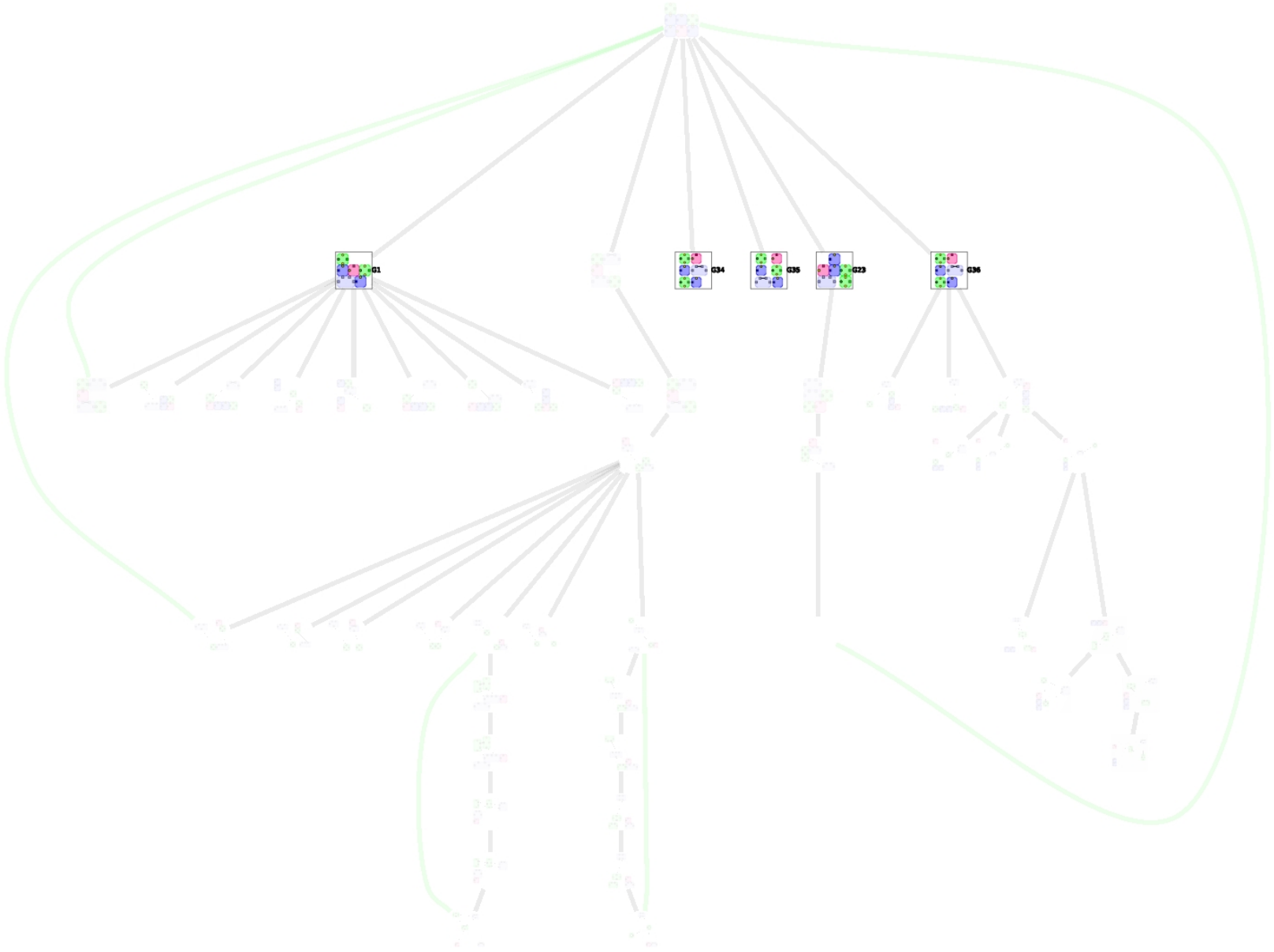
Confluence

- *Confluence* is heavy duty stuff – both conceptually and computationally
  - After identifying isomorphic copies, the derivation tree may be *folded* into a graph
  - Confluence is studied through pattern identification in the *folded* graph

Confluence

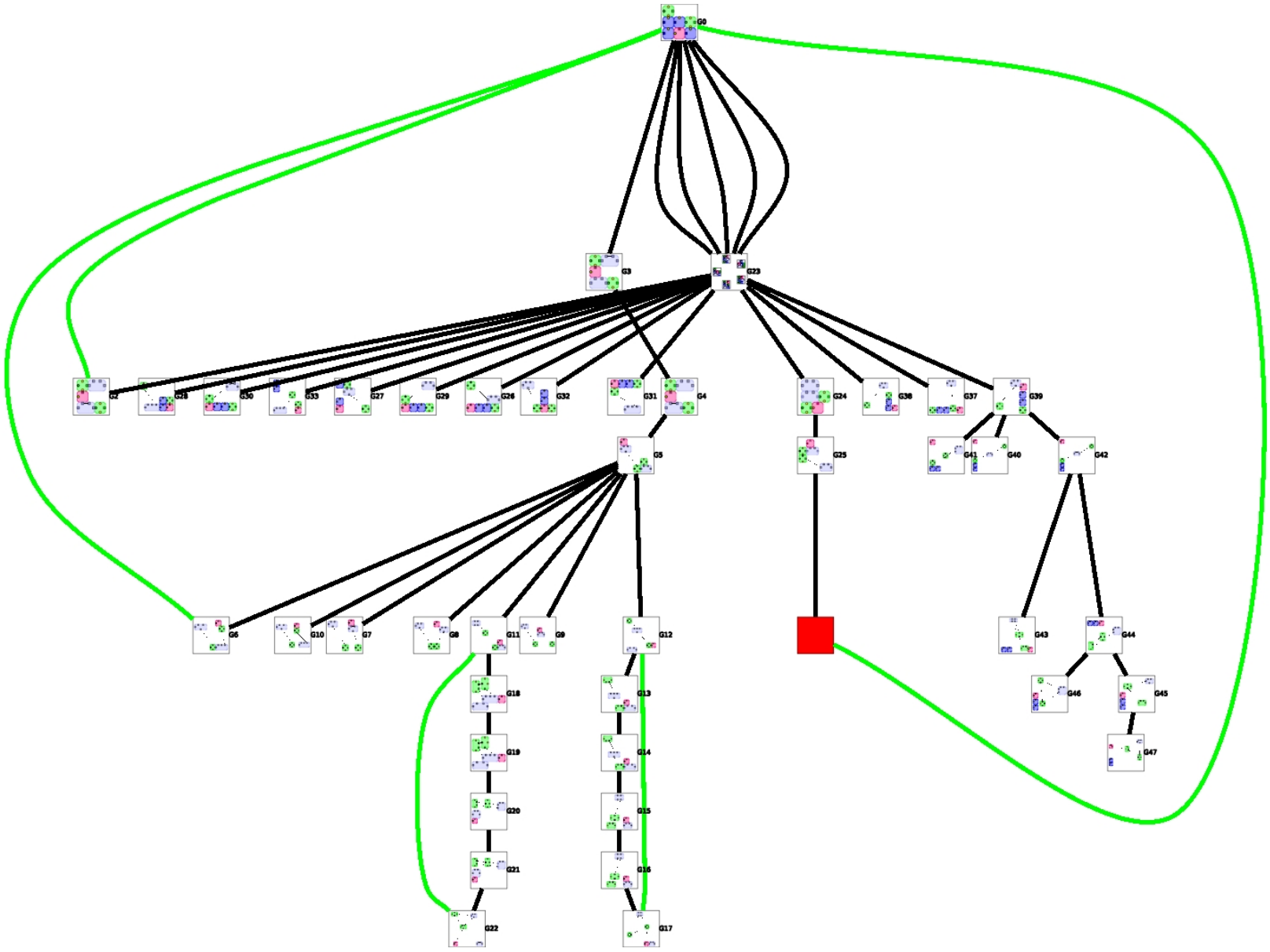

G0
G1
G3
G34
G35
G23
G36
G2
G28
G30
G33
G27
G29
G26
G32
G31
G4
G24
G38
G37
G39
G5
G25
G41
G40
G42
G6
G10
G7
G8
G11
G9
G12
G43
G44
G18
G13
G46
G45
G19
G14
G47
G20
G15
G21
G16
G22
G17

Confluence

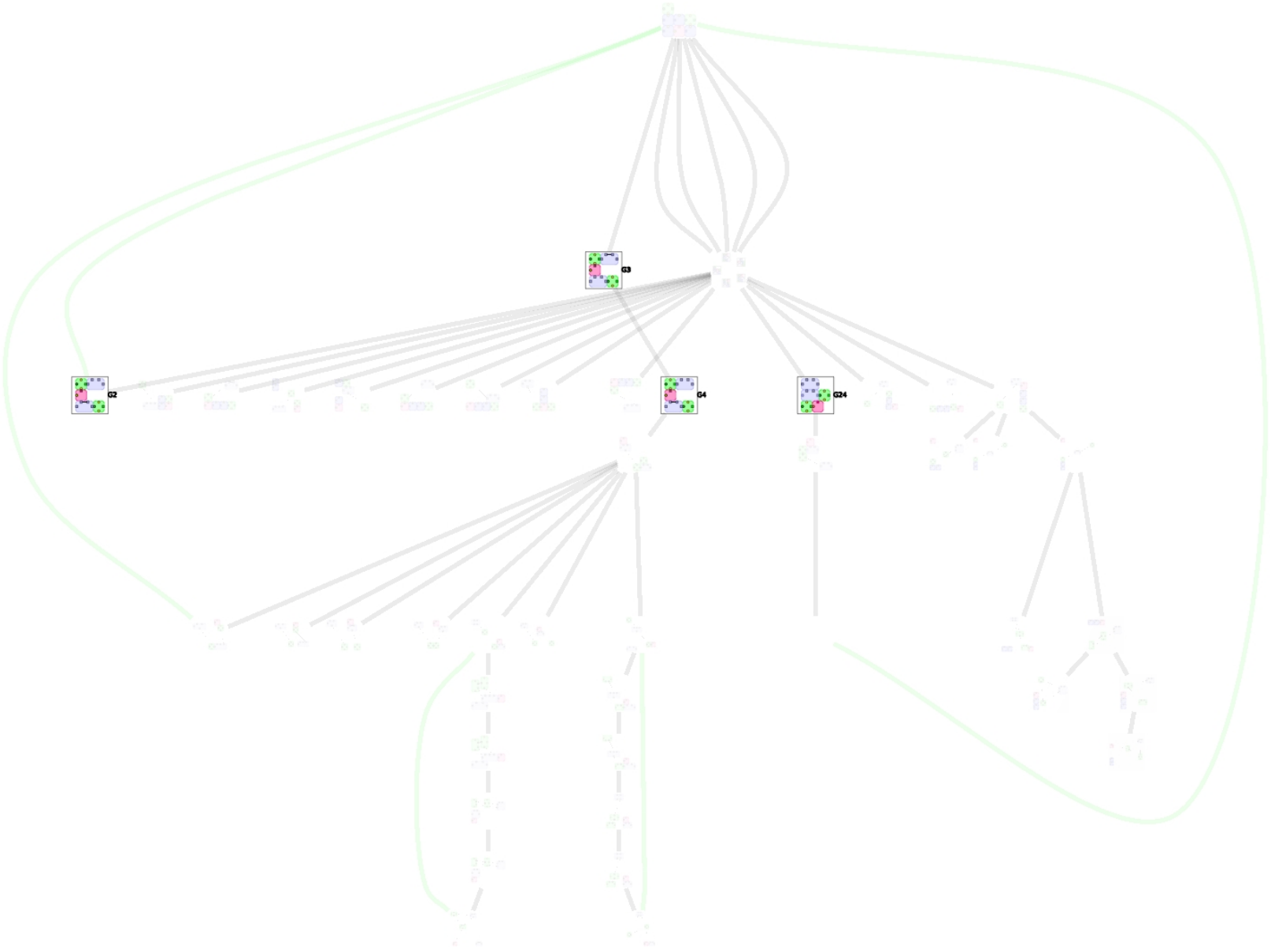


Confluence

Confluence

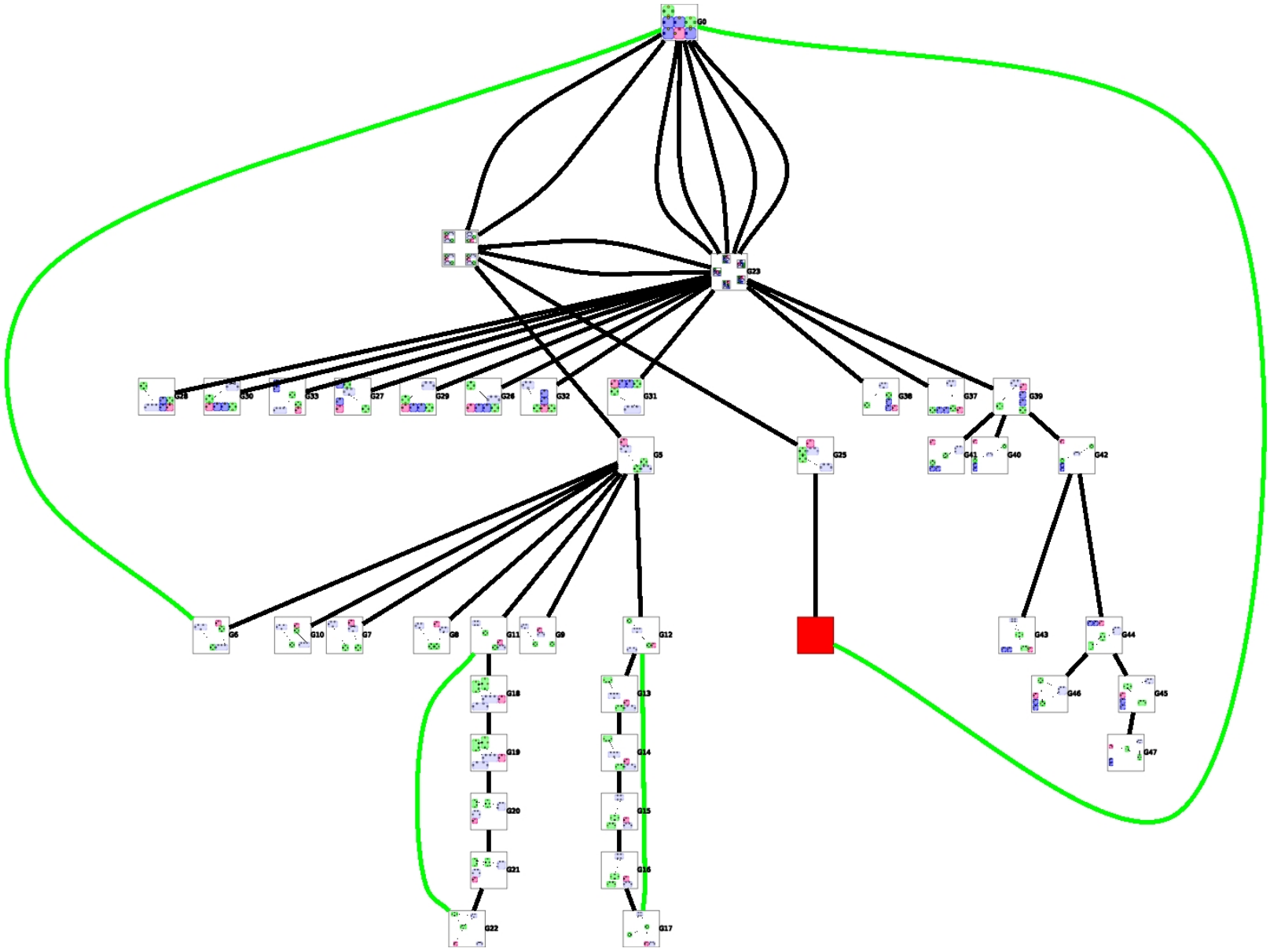

G0
G3
G23
G2
G28
G30
G33
G27
G29
G26
G32
G31
G4
G24
G36
G37
G39
G5
G25
G41
G40
G42
G6
G10
G7
G8
G11
G9
G12
G43
G44
G18
G13
G46
G45
G19
G14
G47
G20
G15
G21
G16
G22
G17

Confluence



Confluence

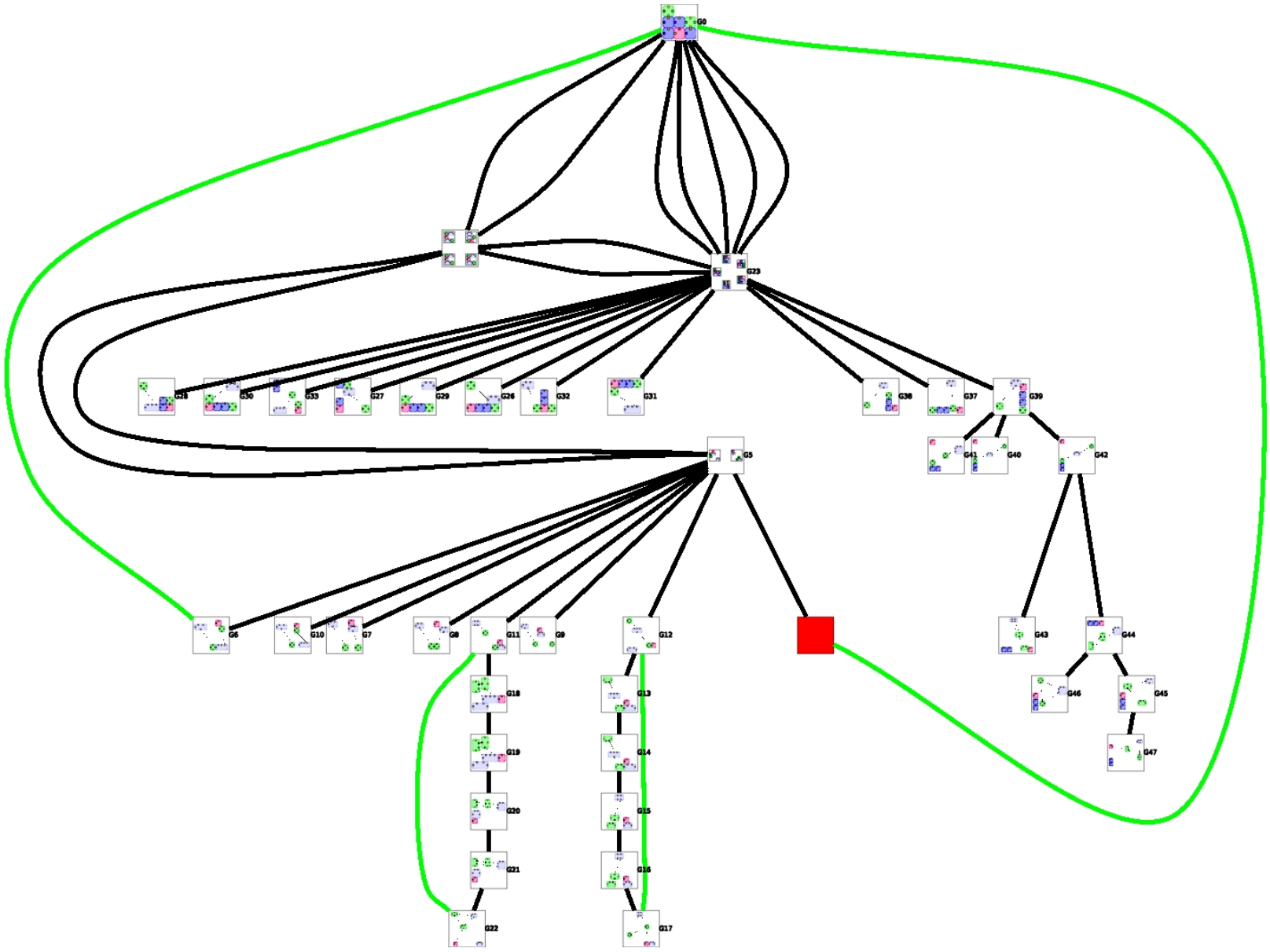
Confluence


G0
G23
G28
G30
G33
G27
G29
G26
G32
G31
G36
G37
G39
G5
G25
G41
G40
G42
G6
G10
G7
G8
G11
G9
G12
G43
G44
G18
G13
G46
G45
G19
G14
G47
G20
G15
G21
G16
G22
G17

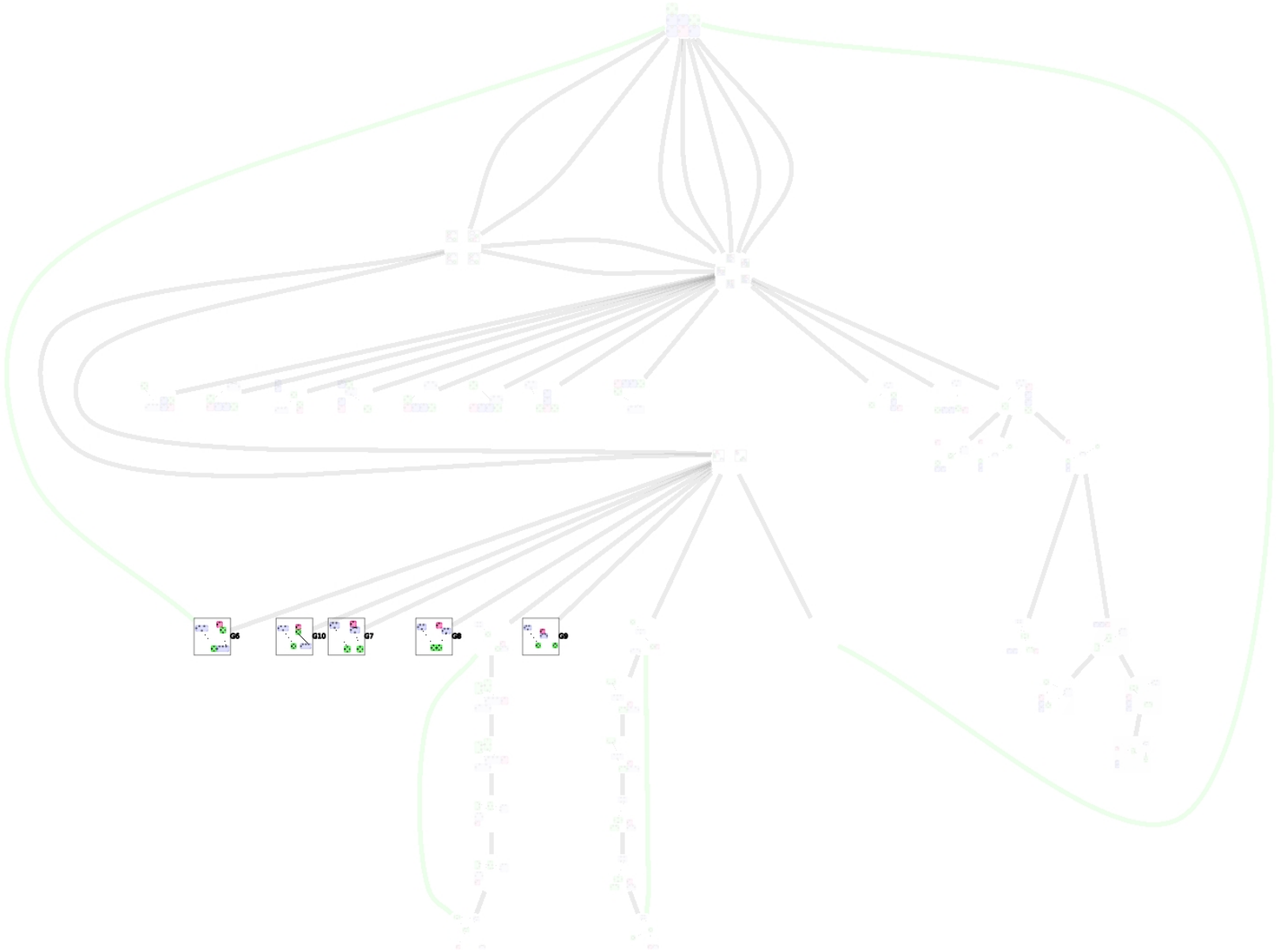
Confluence

Confluence

Confluence

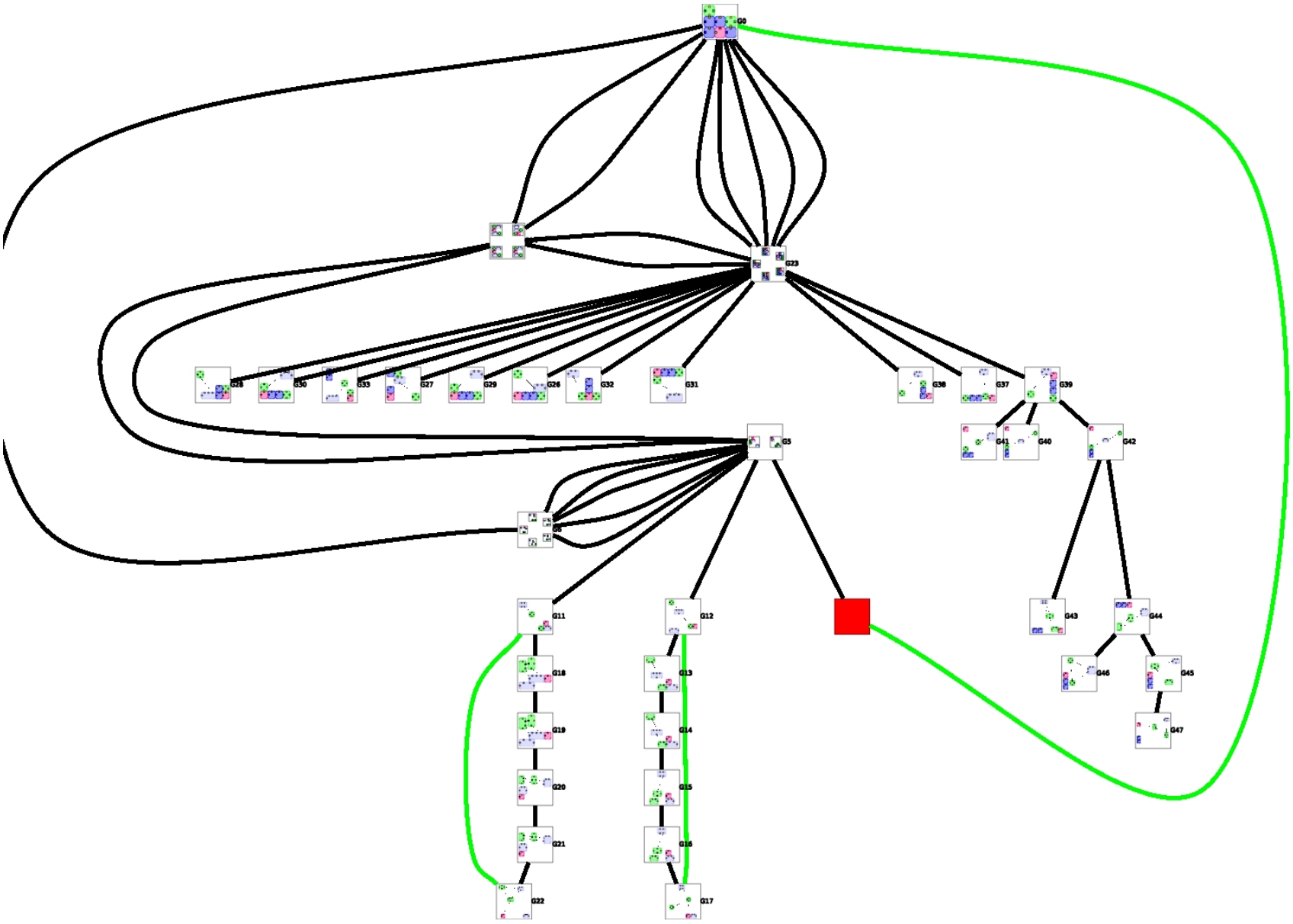

G0
G23
G28
G30
G33
G27
G29
G26
G32
G31
G36
G37
G39
G41
G40
G42
G5
G6
G10
G7
G8
G11
G9
G12
G43
G44
G18
G13
G46
G45
G19
G14
G47
G20
G15
G21
G16
G22
G17

Confluence



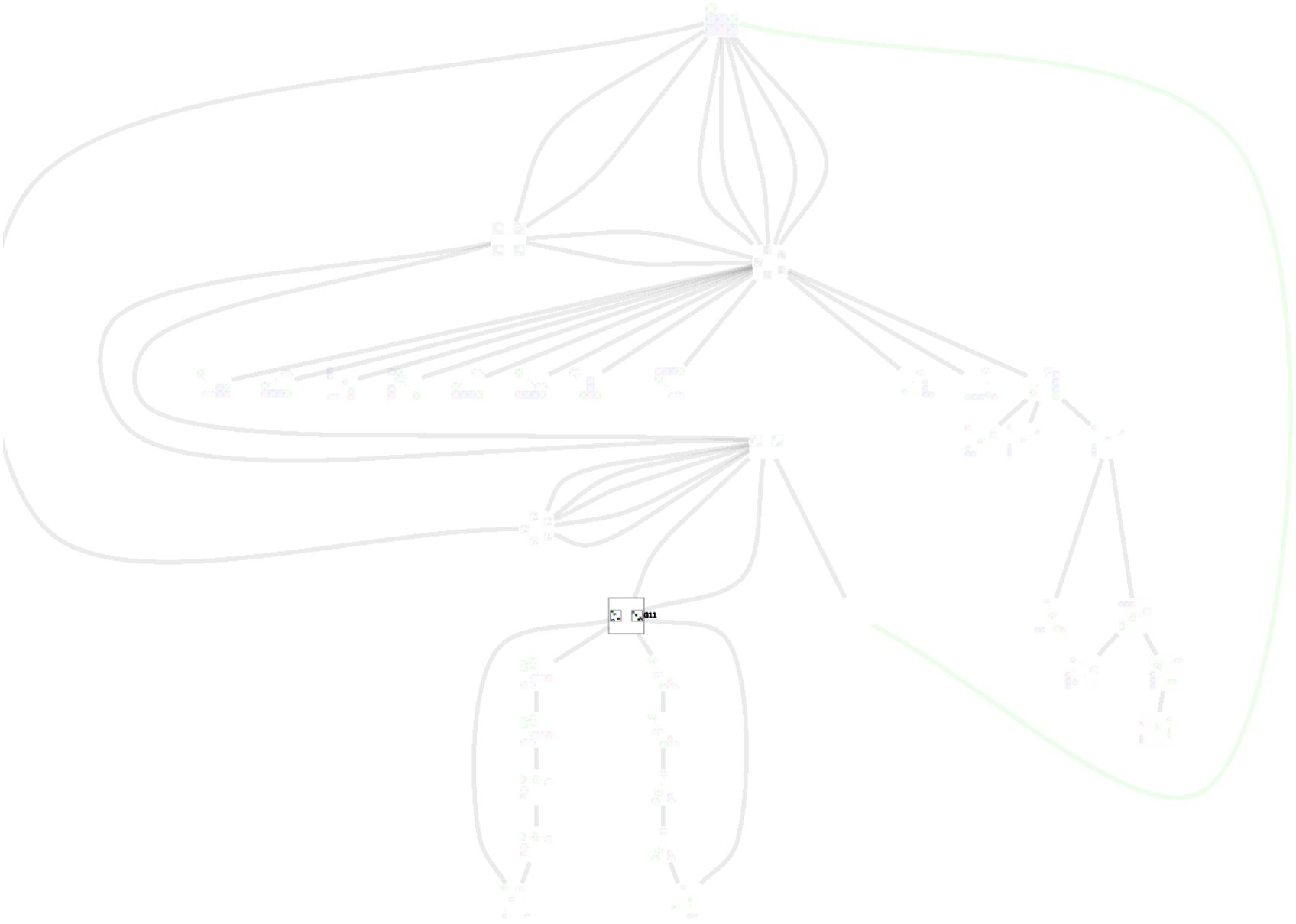
Confluence

Confluence

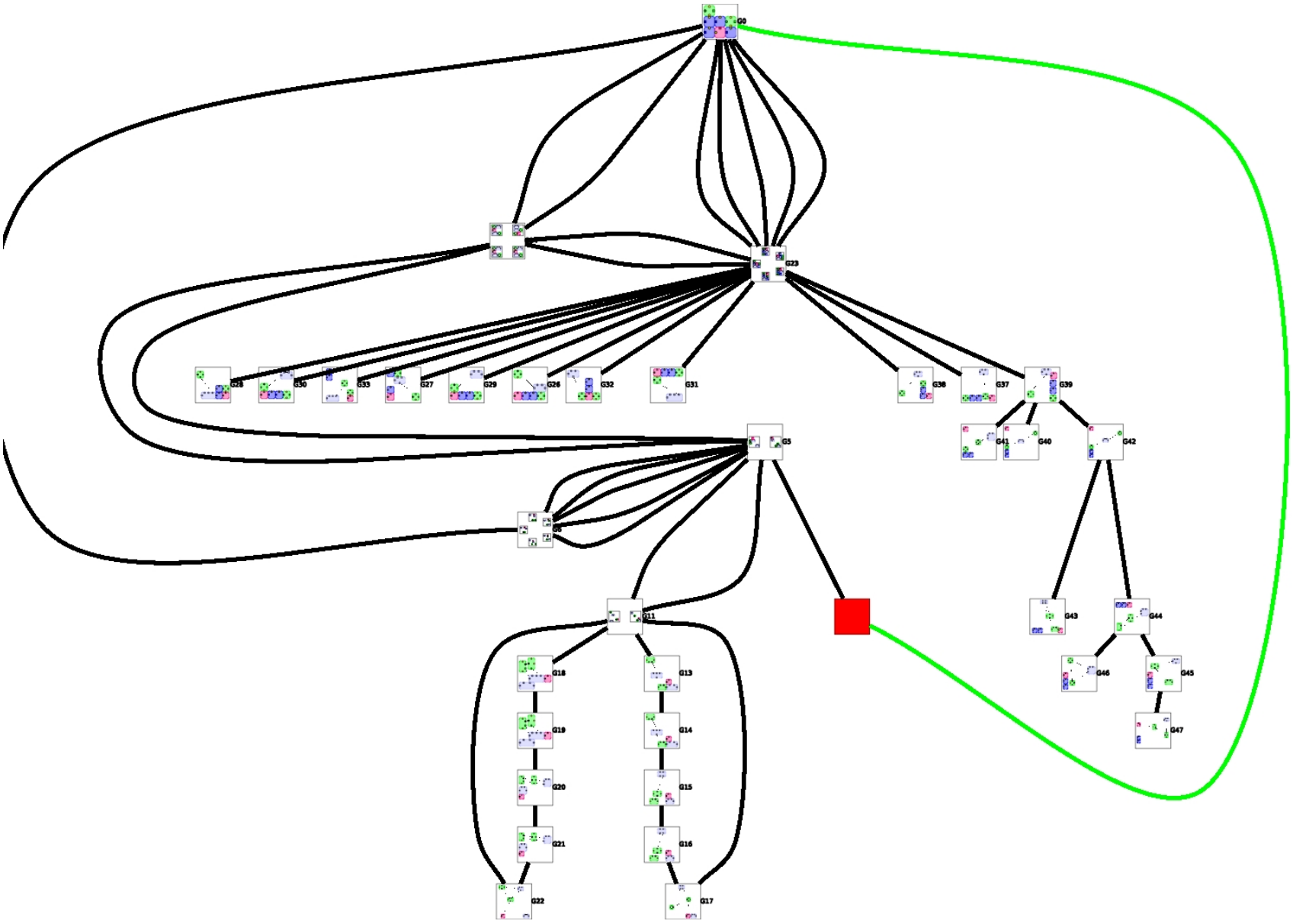

G0
G23
G28
G30
G33
G27
G29
G26
G32
G31
G36
G37
G39
G41
G40
G42
G5
G6
G11
G12
G43
G44
G18
G13
G46
G45
G19
G14
G47
G20
G15
G21
G16
G22
G17

Confluence

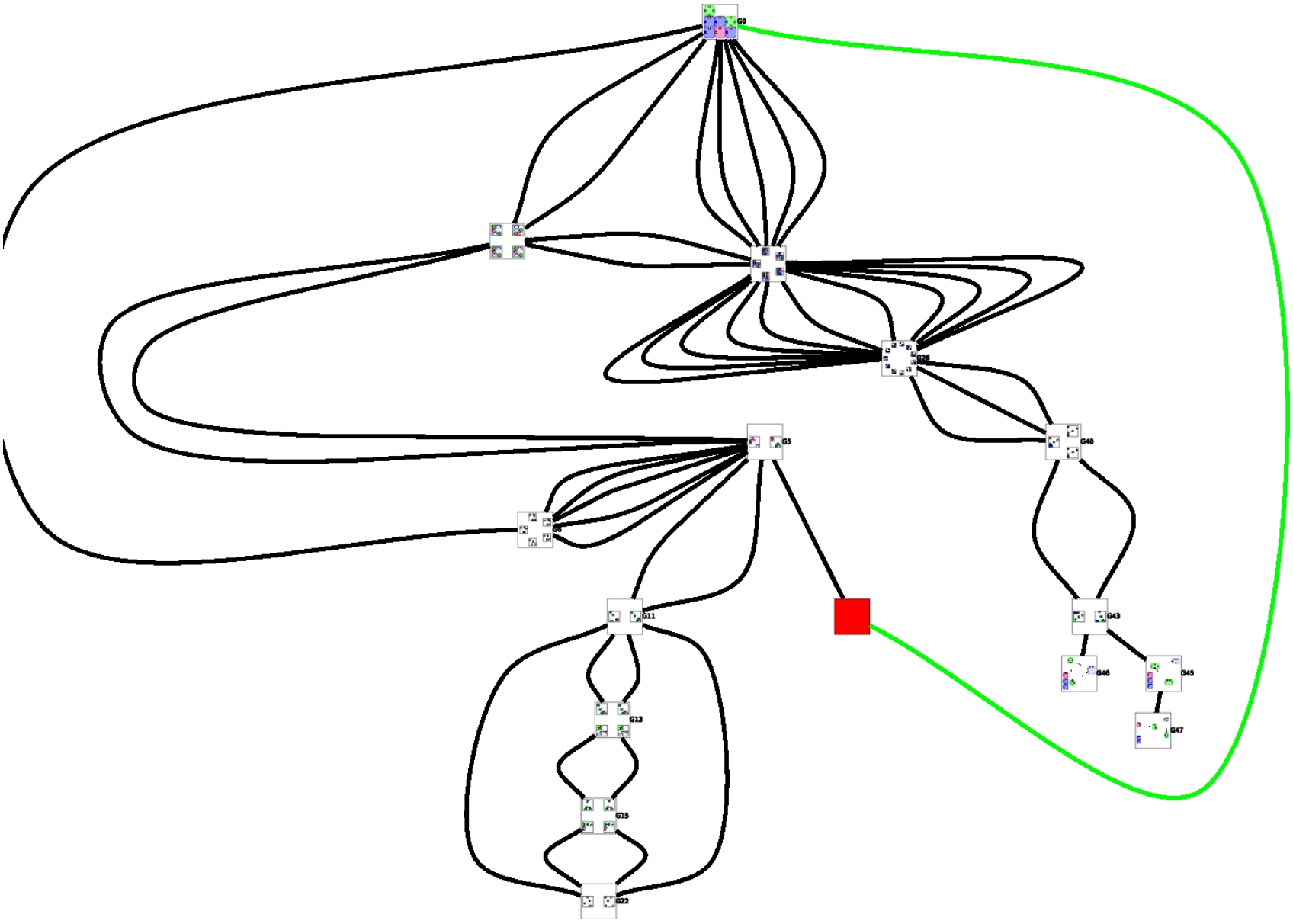
Confluence



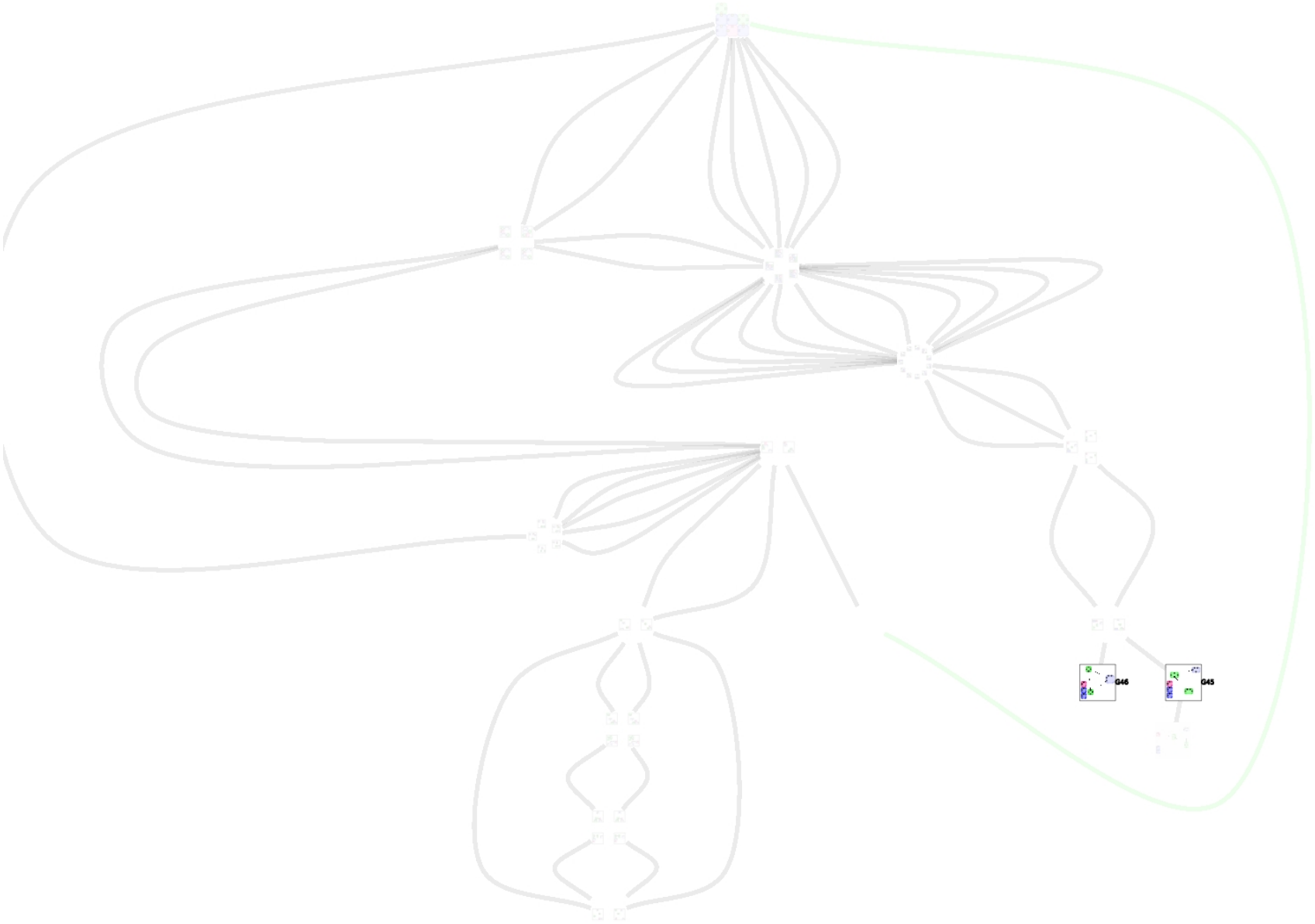
Confluence


G0
G23
G28
G30
G33
G27
G29
G26
G32
G31
G36
G37
G39
G41
G40
G42
G5
G6
G11
G43
G44
G46
G45
G47
G18
G19
G20
G21
G22
G13
G14
G15
G16
G17

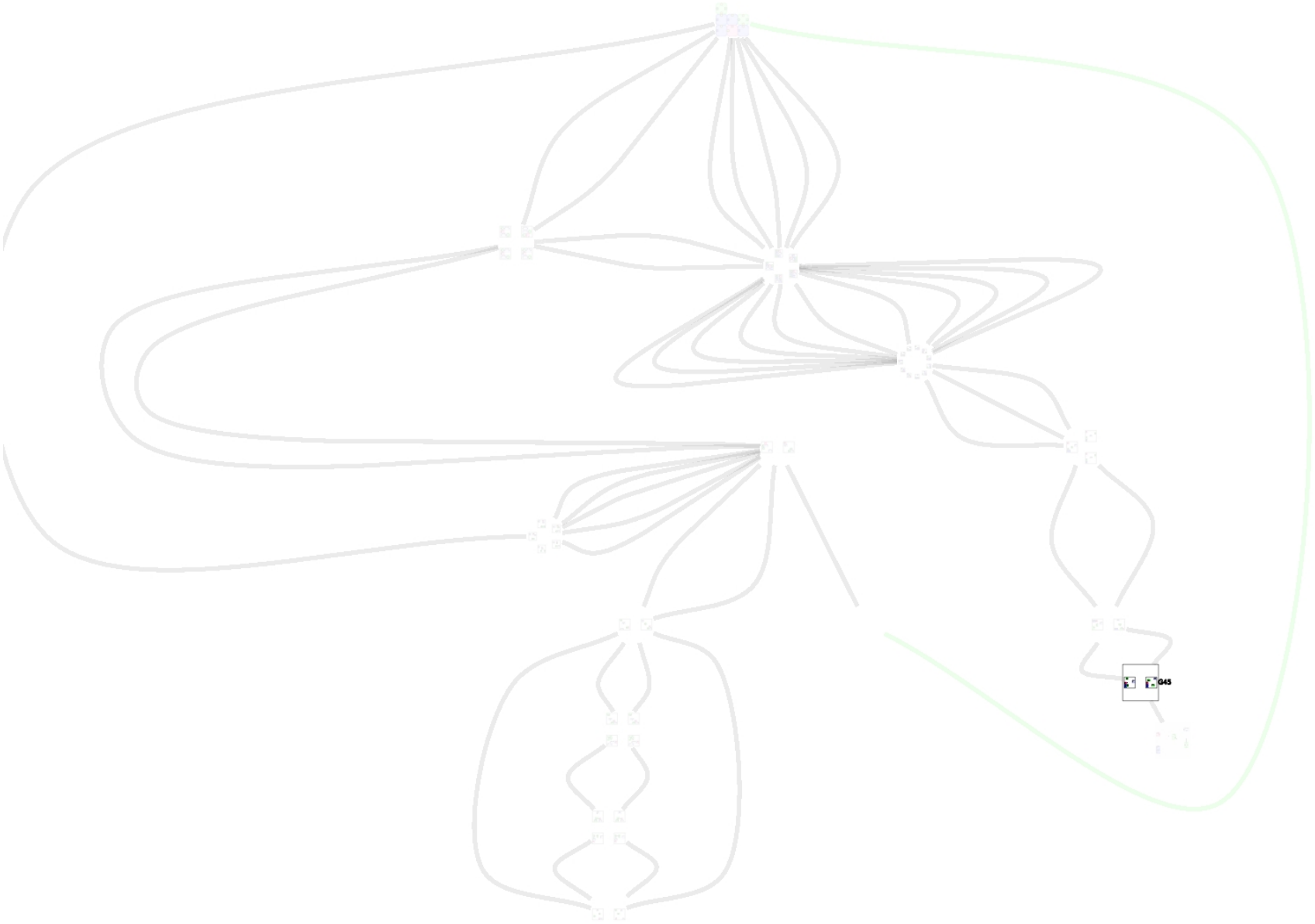
Confluence



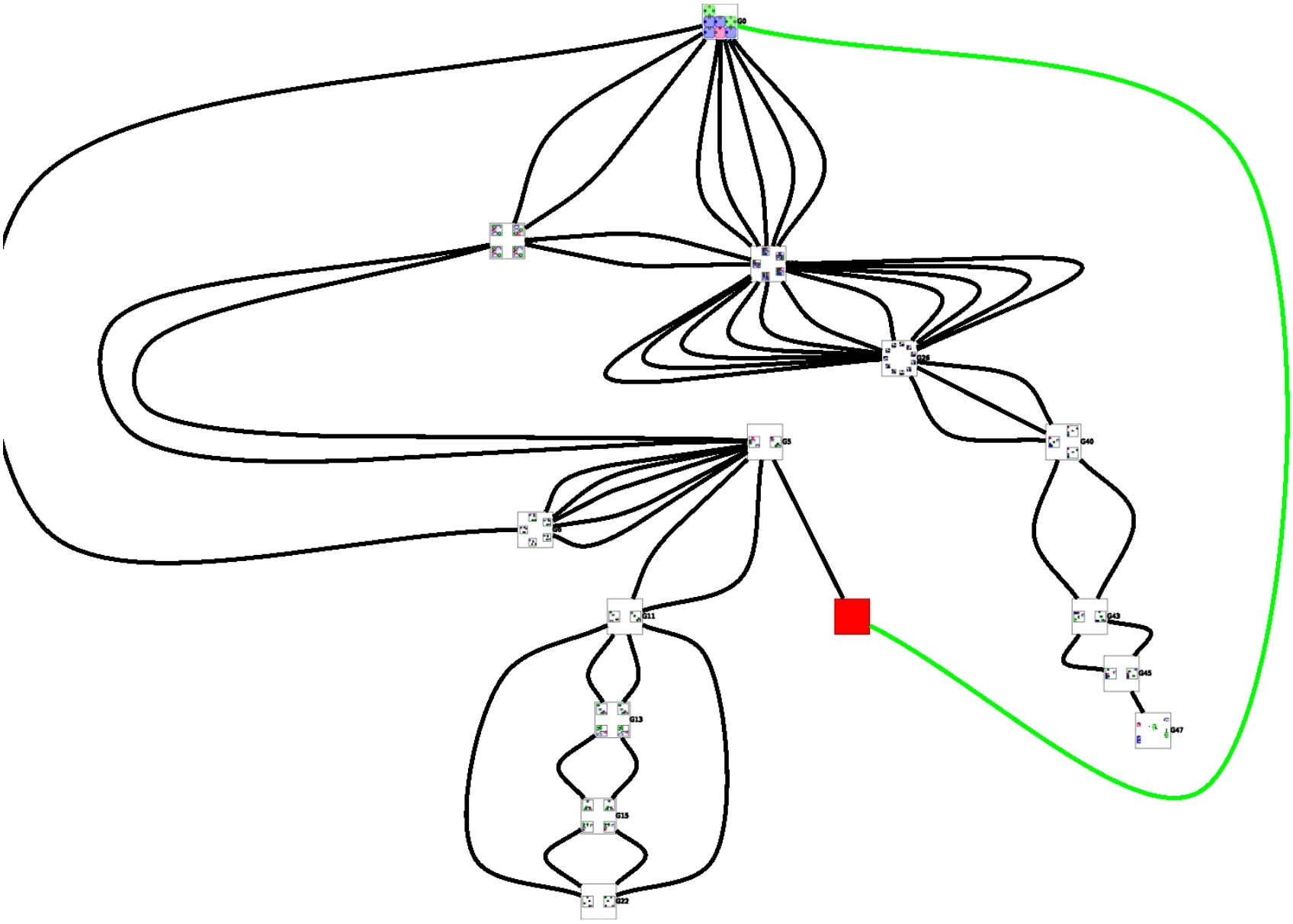
Confluence



Confluence



Confluence

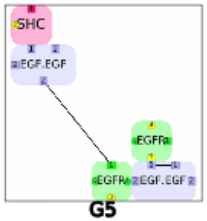


Visual encodings

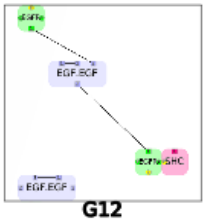
- Node-link diagrams came as an obvious choice due to strong graphical conventions (from both user communities)
  - For graphs
  - For rules
- Similarly, the derivation tree is drawn using a classical top-down hierarchical layout

Visual encodings

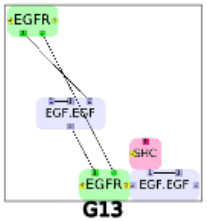
- We introduced small multiples as an option to graph animation [Archambault et al. 2011]



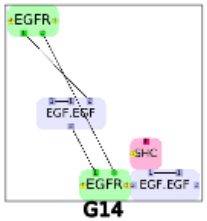
G5



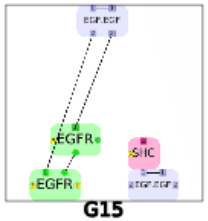
G12



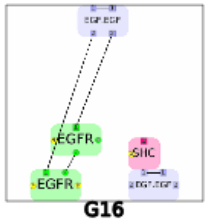
G13



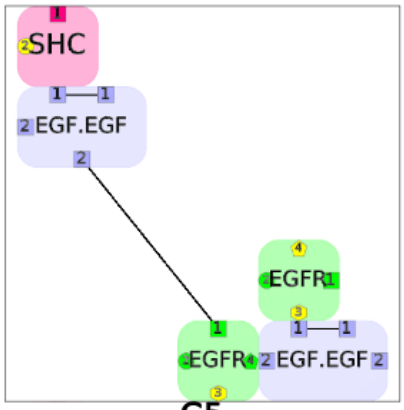
G14



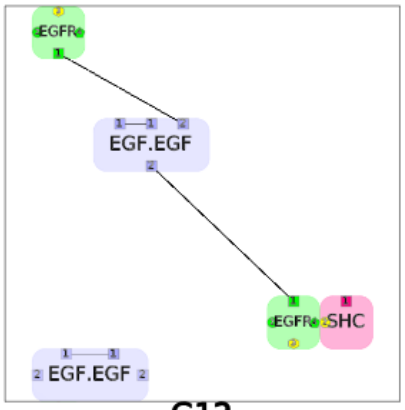
G15



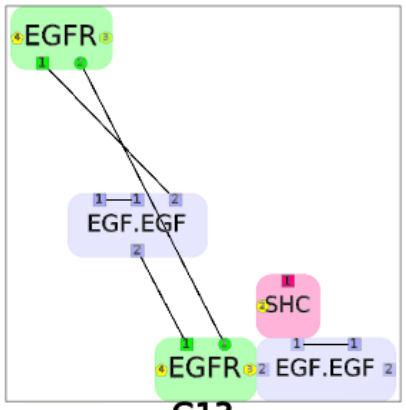
G16



G5



G12



G13

Interaction

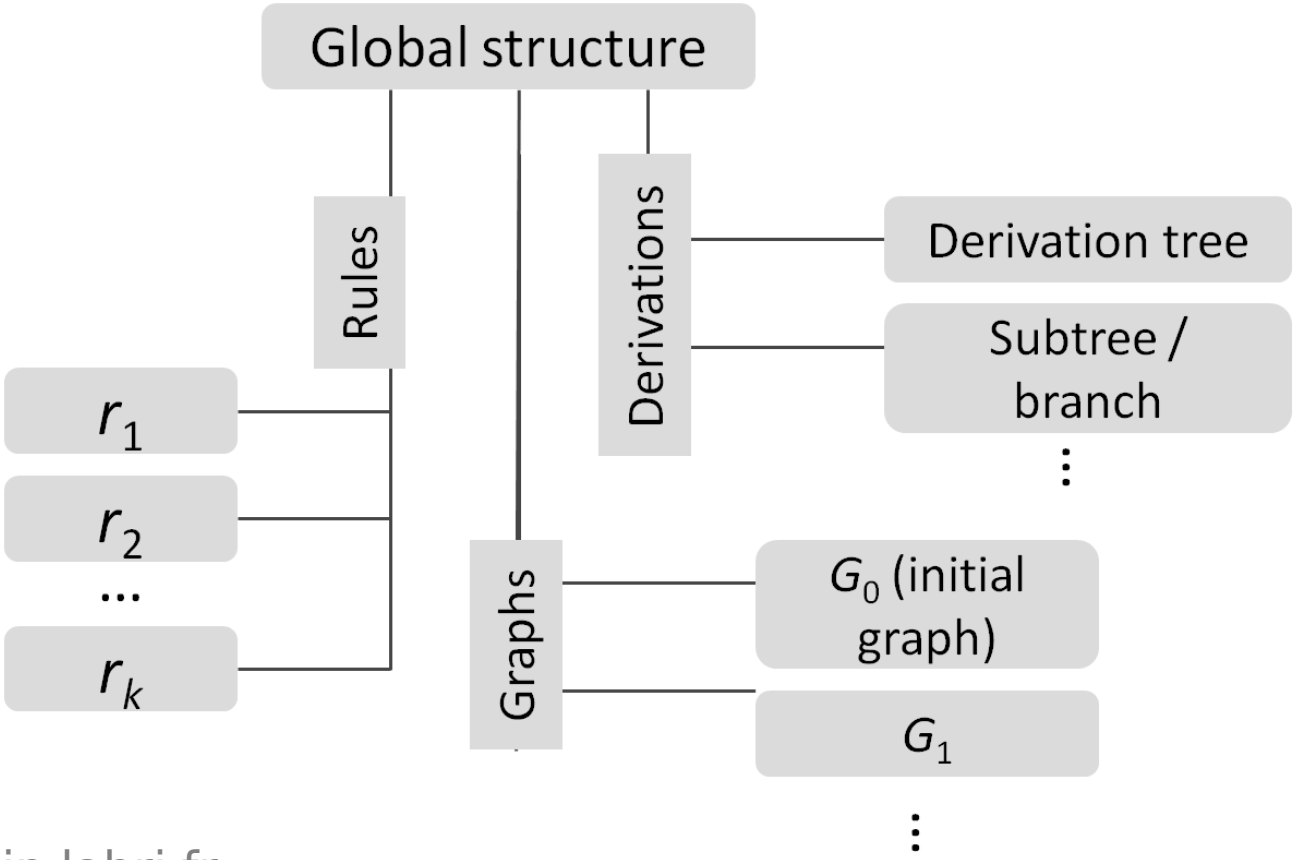
- Ease entities manipulation between views (drag & drop)
  - Rules / strategies dropped on graphs
- Tooltips to have a closer look at entites (rules, graphs) without having to instantiate views
  - For graphs
  - For rules
- Selection of entities *across* all views

Data structure

- Implementing the necessary underlying data structures is far from obvious
  - All graphs resulting from rule applications share a common pool of nodes and edges
  - Derivation tree: nodes contain graphs
  - Folded graph
  - Nodes in scatterplots *are* graphs – allows direct selection from/to derivation tree

Data structure

- Implementing the necessary underlying data structures is far from obvious



Discussion

- PORGY is quite unique in offering simulation steering using a derivation tree, as far as GRS are concerned
  - Compares to World Lines [Waser et al. 2010] in that respect
  - EGRF model was post-validated using combined GRS and parameter tracking
  - Visualization & Interaction vs text-based approaches
- The design of PORGY relies on long term user experience, based on Munzner's formal approach to viz design
  - Potential impact on both GRS and domain application communities

Future Work

- Layout stability issues
  - Difficult because incremental change take place over a hierarchy
  - The drawing of rules often relies on implicit assumptions: no universal layout for rules
- Extend model tracking to multiple parameters
- Graph folding
  - Scalability issues with subgraph isomorphism

Thanks

- Download the tool http://tulip.labri.fr
  (search for Projects/PORGY)

- Ask for a live demo

- {Bruno.Pinaud, Guy.Melancon}@labri.fr