Automated Visualization of Rule-based Models

John A. P. Sekar, Jose-Juan Tapia, James R. Faeder*
University of Pittsburgh School of Medicine
Department of Computational & Systems Biology
{johnsekar,jjtapia,faeder}@pitt.edu
*-corresponding author

Supplementary Information

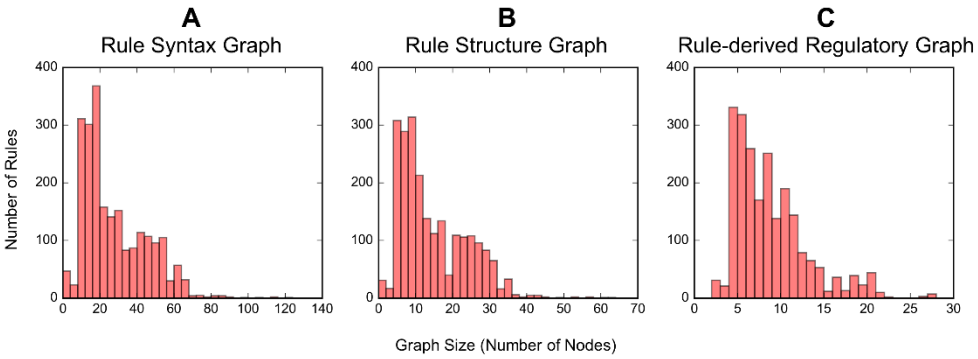

Fig S1. Rule size distributions. Distribution of rule sizes for 2239 rules from 27 rule-based models in the literature models when represented as **(A)** rule syntax graph, **(B)** rule structure graph and **(C)** rule-derived regulatory graph. In all three representations, it is seen that large rules are infrequently encountered.

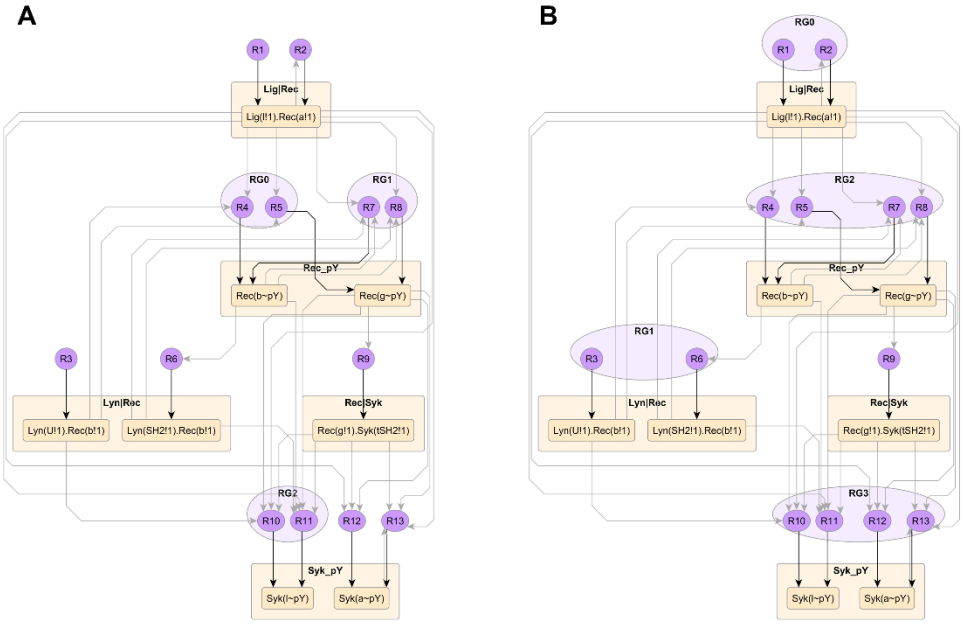

Fig S2. Group Structures on the Model Regulatory Graph. On the full model regulatory graph of Faeder et al. [13], the default grouping heuristic for atomic patterns grouped phosphorylation sites on the same molecules and binding interactions between the same pairs of molecules. Then, rules were grouped automatically by analyzing edge signatures of individual rules, i.e. how

they relate to adjacent atomic patterns and their groups. **(A)** A strict edge signature was used, which accounted for all three edge types. Rule variants that differ only in context are still considered different from each other (e.g. R12 and R13). **(B)** A permissive edge signature was used, which ignored context edges. This resulted in a broader grouping of rules that does not distinguish between contextual variants.

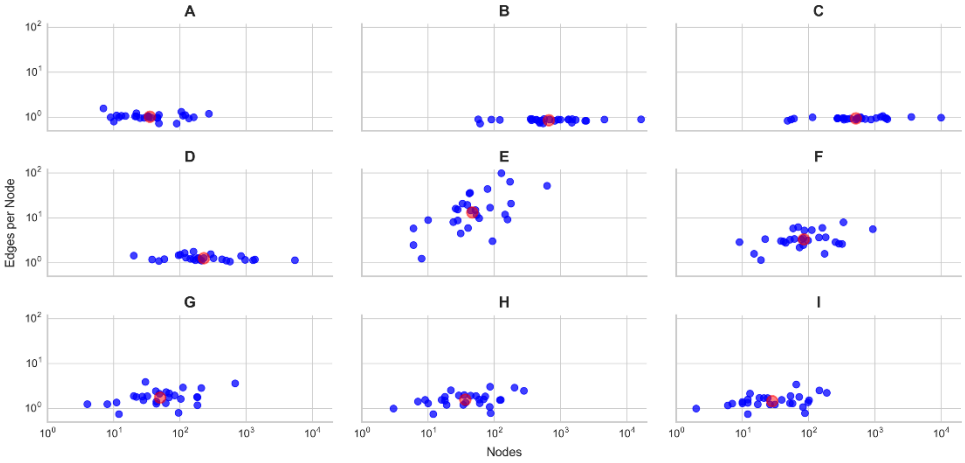

Fig S3. Distribution of graph sizes for various visualization methods. Each graph **(A-I)** has 27 data-points (blue) corresponding to 27 rule-based models from the literature, listed in S6 Appendix. The red data point marks the coordinates of the geometric mean of all 27 points. The graphs are as follows: (A) contact map, (B) conventional rule visualization, (C) compact rule visualization, (D) Simmune Network Viewer, (E) rule influence diagram, (F) full model regulatory graph, (G) model regulatory graph with background removed, (H) model regulatory graph with background removed and nodes compressed using a strict edge signature, (I) model regulatory graph with background removed and nodes compressed using a permissive edge signature. The geometric means shown are the same values plotted in Fig 13 and measure the center of each distribution in log-space.

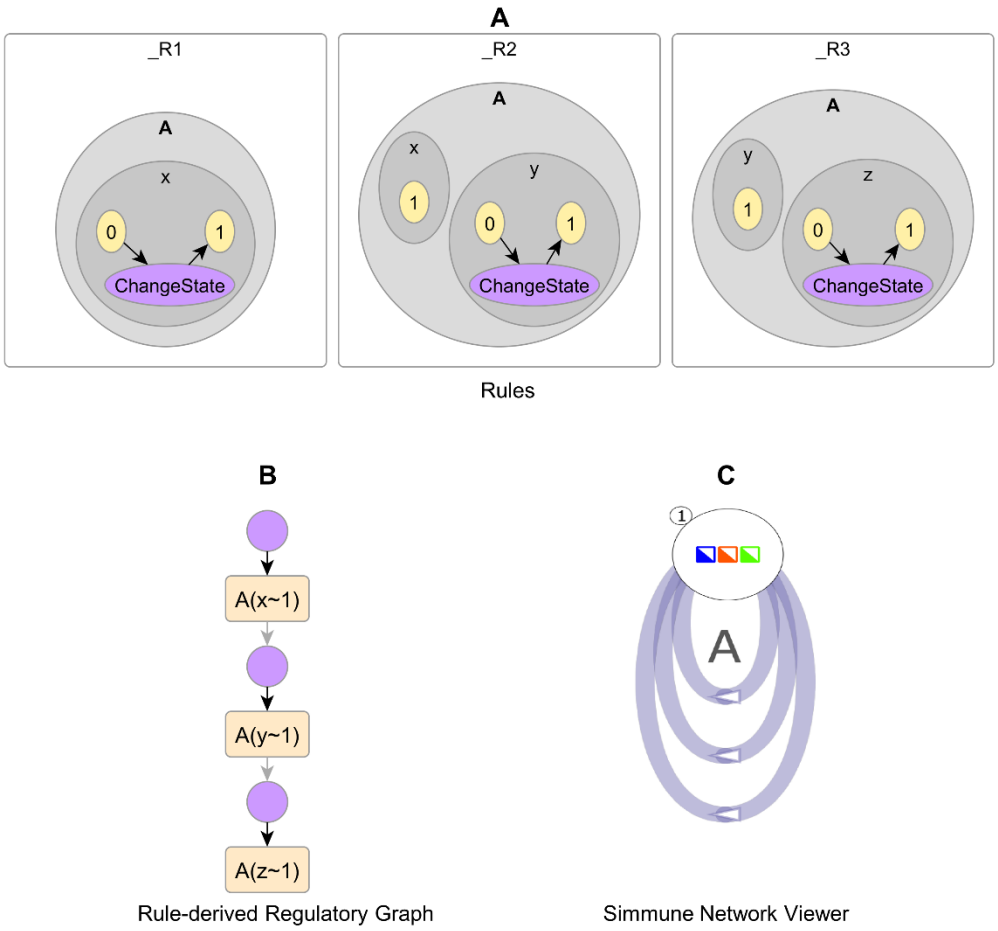

Fig S4. Comparison of rule-derived regulatory graph and Simmune Network Viewer. (A) Shown is a model in which three sites on a protein are activated in sequence. **(B)** The sequence is evident on the rule-derived regulatory graph. **(C)** The sequence cannot be seen on the Simmune Network Viewer diagram because the three patterns used have the same molecule stoichiometry {A=1} and are represented by the same node, which obscures information mediated through state changes.

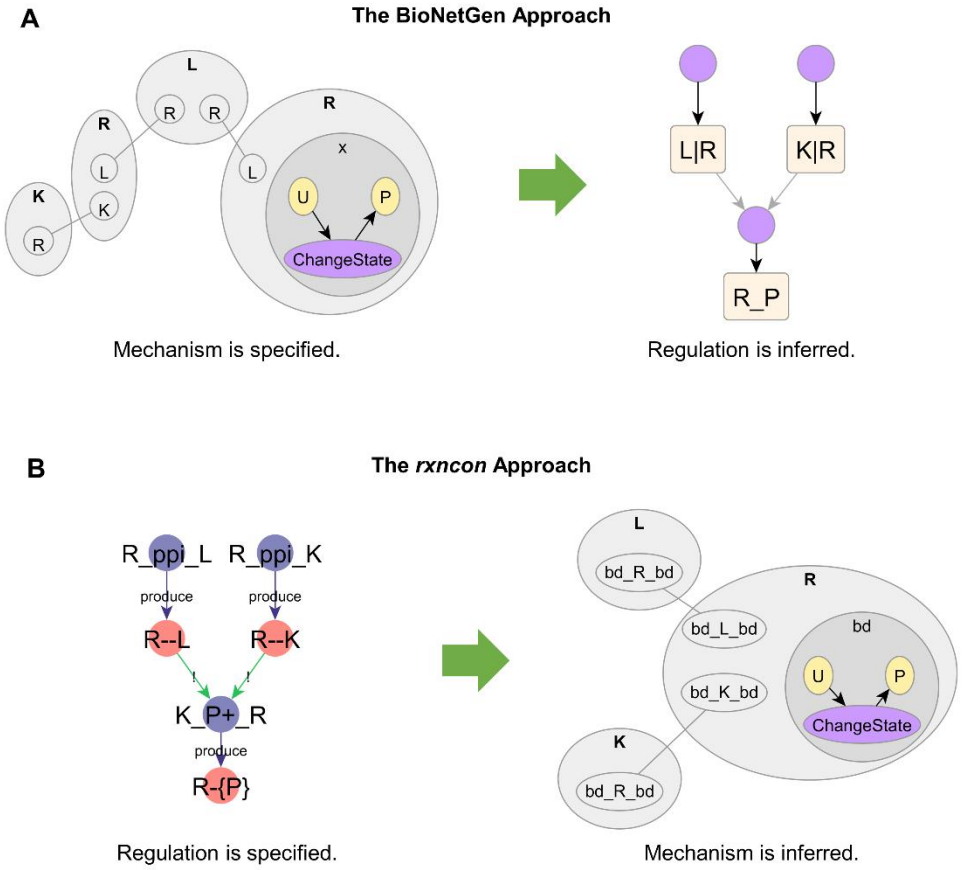

Fig S5. Comparison of BioNetGen and *rxncon* regulatory graphs. **(A)** In BioNetGen, complex reaction mechanisms are specified as reaction rules and the rule-derived regulatory graph is inferred by analyzing the specified rules. The reaction rule shown models trans-phosphorylation of receptor R in the ligand-crosslinked dimer configuration by recruited kinase K, a frequently encountered mechanism in biochemical signaling. **(B)** In *rxncon*, regulation is specified using the *rxncon* syntax and directly visualized as the regulatory graph. Reaction mechanisms are reconstructed from the specified regulatory interactions and are limited to a small set of mechanisms, e.g. the current version of *rxncon* does not natively support trans-phosphorylation reactions.

S6 Appendix: List of Rule-based Models

No.	Citation	System	Rules
1	**An et al. 2009** Math BioSci.	TLR4	59
2	**Barua et al. 2008** J Biol Chem.	PDGFRβ	43
3	**Barua et al. 2009** PLoS Comput Biol.	GH-Jak	6
4	**Barua et al. 2012** J Immunol.	BCR	44
5	**Barua et al. 2012** PLoS One	FcεRI	86
6	**Barua et al. 2013** PLoS Comput Biol.	APC	79
7	**Blinov et al. 2006** Biosystems	EGFR	39
8	**Chylek et al. 2014** Front Immunol.	FcεRI	145
9	**Chylek et al. 2014** PLoS One	TCR	158
10	**Creamer et al. 2012** BMC Syst Biol.	ErBB	625
11	**Dushek et al. 2011** Biophys J.	Multi-site phosphorylation	128
12	**Dushek et al. 2014** Biophys J.	FRET Biosensors	10
13	**Faeder et al. 2003** J Immunol.	FcεRI	24
14	**Falkenberg et al. 2013** Biophys J.	TLBR	6
15	**Hat et al. 2016** PLoS Comput Biol.	p53	94
16	**Kesseler et al. 2013** J Theor Biol.	G2 checkpoint	173
17	**Kocieniewski et al. 2012** J Theor Biol.	MAPK	28
18	**Kozer et al. 2013** Mol Biosyst.	EGFR	26
19	**Kozer et al. 2014** Biochemistry	EGFR-Grb2	28
20	**Ligon et al. 2014** PLoS One	Lipoplex transfection	33
21	**Michalski et al. 2012** Phys Biol.	CaMKII	8
22	**Mukhopadhyay et al. 2013** PLoS Comput Biol.	TCR	42
23	**Nag et al. 2009** Biophys J.	Lat-Grb2-SOS1	52
24	**Pekalski et al. 2013** PLoS One	TNFα	40
25	**Stites et al. 2015** Biophys J.	EGFR	178
26	**Szymanska et al. 2015** PLoS One	MTORC1	31
27	**Thomson et al. 2011** PNAS	Yeast GPCR	54

S7 Appendix: Algorithms

S7.1 Pattern Structure Graph

Goal: Given a BioNetGen pattern, compose a graph in which each element of the pattern is a node and each edge is a hierarchical relationship. The node has attributes (NodeIndex, NodeLabel, NodeType) where NodeLabel is a name, NodeType is one of {molecule, component, internal state, bond state} and NodeIndex is a unique index. Bond state nodes have NodeLabel + to indicate bonds and – to indicate unbound state.

Example: Given the pattern A(b!1).B(a~x!1), we get a graph with six nodes { (1,A,molecule), (2,B,molecule), (3,b,component), (4,a,component), (5,x,internal state), (6,+,bond state) } and five edges {(1,3),(2,4),(2,5),(3,6),(4,6)}.

Input: Pattern $p = (x)$ and a map $type: p \rightarrow$ {molecule, component, internal state, bond}. Each element x in p has one of the following forms:

(m, m_t), is a molecule with index m of type m_t,
(c, c_t, m), a component with index c of type c_t in molecule m,
(s, s_t, c), is an internal state with index s of type s_t on component c,
(b, c_1, c_2) is a bond with index b between components c_1 and c_2,
(b, c) is a wildcard bond with index b on component c

initialize empty sets V, E and function $bonded$
for each x in p,
    if $x = (m, m_t)$ and $type(x) =$ molecule
        add node (m,m_t,molecule) to V
    if $x = (c, c_t, m)$ and $type(x) =$ component
        add node (c, c_t, component) to V
        add edge (m, c) to E
        $bonded(c) \leftarrow$ False
    if $x = (s, s_t, c)$ and $type(x) =$ internal state
        add node (s,s_t,internal state) to V
        add edge (c, s) to E
    if $x = (b, c_1, c_2)$ and $type(x) =$ bond
        add node (b,+, bond state) to V
        add edges $(c_1, b), (c_2, b)$ to E
        $bonded(c) \leftarrow$ True
    if $x = (b, c)$
        add node (b,+,bond state) to V
        add edge (c, b) to E
        $bonded(c) \leftarrow$True
for each component x in p such that $bonded(c) =$False
    add node (u,−,bond state) to V, where u is a unique id
    add edge (c, u) to E
$G \leftarrow (V, E)$

Output: Pattern structure graph G.

Complexity: $\mathcal{O}(|p|)$, i.e. linear in the size of the pattern.

Notes: From above, $|E| \leq 2|V|$. We will treat $|E| \propto |V|$ for pattern structure graphs in S7.2, S7.3.

S7.2 Correspondence Map

Goal: Given a BioNetGen rule, synthesize a partial map between reactants and products.

Example: Given a rule A(b~x) -> A(b~y), the reactant structure graph is {(1,A,molecule), (2,b,component), (3,x,internal state), (4,-,bond state)}. Similarly, the product structure graph is {(5,A,molecule), (6,b,component), (7,y,internal state), (4,-,bond state)}. The correspondence map is {(1->5), (2->6), (4->8)}. For simplicity, we can denote this as A->A, b->b.

Input: Rule (p_l, p_r) where $p_l := (x_l)$, $p_r := (x_r)$ and $type: \{x | x \in p_l \cup p_r\} \rightarrow$ { molecule, component, internal state, bond }. p_l and p_r are merged patterns of the left and right sides of the rule respectively.

1 $M^l = (x)$, a sequence with molecules x drawn from p_l
2 $M^r = (x)$, a sequence with molecules x drawn from p_r
3 for each $x \in M^l$ such that $x = (m, m_t)$
4 $C_x = (y)$, a sequence with components y drawn from p_l s.t. $y = (c, c_t, m)$
5 for each $x \in M^r$ such that $x = (m, m_t)$
6 $C_x = (y)$, a sequence with components y drawn from p_r s.t. $y = (c, c_t, x)$
7 initialize empty sets dom, img and function $\psi: dom \rightarrow img$
8 for each $x \in M^l$ such that $x = (m^x, m_t^x)$ and x not in dom
9 for each $y \in M^r$ such that $x = (m^y, m_t^y)$ and $m_t^x = m_t^y$ and y not in img
10 if exists $C_x \rightarrow C_y$ preserving component type, presence of internal state, wildcard
11 add x to dom and y to img
12 $\psi(x) \leftarrow y$
13 for each molecule map $(x \rightarrow y)$ in ψ
14 for each a in C_x such that $a = (c^a, c_t^a, m^a)$ and a not in dom
15 for each b in C_x such that $b = (c^b, c_t^b, m^b)$ and $c_t^a = c_t^b$ and b not in img
16 add a to dom and b to img
17 $\psi(a) \leftarrow b$
18 for each component map $(a \rightarrow b)$ in ψ such that $a = (c^a, c_t^a, m^a), b = (c^b, c_t^b, m^b)$
19 if exists $(s^l, s_t^l, c^l) \in p^l, (s^r, s_t^r, c^r) \in p^r$ such that $c^l = c^a, c^r = c^b, s_t^l = s_t^r$
20 add (s^l, s_t^l, c^l) to dom and (s^r, s_t^r, c^r) to img
21 $\psi((s^l, s_t^l, c^l)) \leftarrow (s^r, s_t^r, c^r)$
22 if exists $(b^l, c^l) \in p^l, (b^r, c^r) \in p^r$ such that $c^l = c^a, c^r = c^b$
23 add (b^l, c^l) to dom and (b^r, c^r) to img
24 $\psi((b^l, c^l)) \leftarrow (b^r, c^r)$
25 for each bond $g \in p_l$ such that $g = (b^l, c_1^l, c_2^l)$ and g not in dom
26 for each bond $h \in p_r$ such that $h = (b^r, c_1^r, c_2^r)$ and h not in img
27 if exists $\{c_1^l, c_2^l\} \rightarrow \{c_1^r, c_2^r\}$ in ψ
28 add g to dom and h to img
29 $\psi(g) \leftarrow h$

Output: Correspondence map $\psi: p_l \nrightarrow p_r$, where $\nrightarrow$ indicates that it is a partial map.

Complexity: $\mathcal{O}(|p_l| * |p_r|)$. This can be considered $\mathcal{O}(1)$ since rule sizes do not get very large (Fig S1A).

Notes: Since the pattern structure graph has one node for every element in a pattern, the correspondence map can also be defined equivalently on pattern structure graphs, i.e. $\psi: V^l \nrightarrow V^r$

S7.3 Rule Structure Graph

Goal: Given a BioNetGen rule and a correspondence map between reactants and products, synthesize a graph in which the left and right sides are merged together. In addition to NodeIndex, NodeLabel, and NodeType, use the NodeSide attribute to indicate which side of the rule each node is derived from.

Example: Given the rule A(b~x) -> A(b~y) and the map A->A, b->b, it produces a graph with five nodes {(1,A,molecule,both), (2,b,component,both), (3,x,internal state,left), (4,y,internal state,right), (5,-,bond state,both)} and four edges {(1,2), (2,3), (2,4), (2,5)}. The nodes with NodeSide 'both' have been merged

from both sides of the rule whereas the nodes with NodeSide equals 'left' or 'right' come uniquely from the reactant or product sides of the rule respectively.

Input: Rule $\left(G^l, G^r, \psi\right)$ where $G^l \coloneqq \left(V^l, E^l\right), G^r \coloneqq (V^r, E^r)$ are merged pattern structure graphs and $\psi: V^l \nrightarrow V^r$ is a correspondence map.

initialize empty sets V, E
for each node v in V^l, label NodeSide(v)=left
for each node v in V^r, label NodeSide(v)=right
for each map $v \to v'$ in ψ,
    label NodeSide(v) = NodeSide(v') = both
for each edge (v, v') in E^l, add edge (v, v') to E
for each edge (v, v') in E^r
    if NodeSide(v)=NodeSide(v')=both
        next edge
    if NodeSide(v)=both
        $v \leftarrow \psi^{-1}(v)$
    if NodeSide(v')=both
        $v' \leftarrow \psi^{-1}(v')$
    add edge (v, v') to E
$G \leftarrow (V, E)$

Output: Rule structure graph G

Complexity: $\mathcal{O}\left(\left|V^l\right| + |V^r| + \left|E^l\right| + |E^r|\right) \approx \mathcal{O}\left(\left|V^l\right| + |V^r|\right)$ since $|E| \propto |V|$ for pattern structure graphs. The complexity is linear in the size of the rule $\left|V^l\right| + |V^r|$. This can be considered $\mathcal{O}(1)$ since rule sizes do not get very large (Fig S1A).

Notes: The rule structure graph has the same property of the pattern structure graph, i.e. $|E| \propto |V|$.

S7.4 Rule-derived Regulatory Graph

Goal: The rule-derived regulatory graph is a network graph in which nodes have attributes NodeType and NodeLabel and edges have attributes Reactant, Product and Context respectively which take binary values 0/1 each. NodeType can be atomic pattern or rule and NodeLabel is sufficient to index nodes uniquely. Given a rule structure graph of a rule, synthesize a rule-derived regulatory graph in which one node is labeled with the name of the rule.

Example: Given the rule labeled rule1 of the form A(b~x) -> A(b~y) and its corresponding rule structure graph (see S7.3 Example) build a network graph with four nodes having NodeLabel equals rule1, A(b), A(b~x), A(b~y) respectively. The node labeled rule1 is of NodeType rule and the other nodes are of node type atomic pattern. Draw edges { (rule1,A(b)), (rule1, A(b~x)), (rule1, A(b~y)) } respectively with edge labels 001, 100, 010 respectively, where 001 indicates that edge attributes are Reactant=0,Product=0,Context=1 respectively. Each node and edge is drawn by examining a corresponding node on the rule structure graph and its neighbors.

Input: Rule structure graph $G^s \coloneqq (V^s, E^s, r)$ where r is a label indexing the rule.

for each node v in V^s, initialize empty set Parent(v)
for each edge (v, v') in E^s
    if v is a molecule with NodeLabel m_t
        if v' is a component
            add label "$\mathrm{m_t}$" to Parent(v')
    if v is a component with NodeLabel c_t and parent "$\mathrm{m_t}$"
        if v' is an internal or bond state
        add label "$\mathrm{m_t(c_t)}$" to Parent(v')

initialize empty sets V, E
add node v_r=(r,rule) to V
for each node v^s in V^s
    determine atomic pattern NodeLabel a from Table below
    make node v_a =(a, atomicpattern),
    if v_a not in V,
        add node v_a to V
        add edge (v_a, v_r) to E with default labels (Reactant=0,Product=0,Context=0)
        apply label modification of edge (v_a, v_r) from Table
$G \leftarrow (V, E)$

Output: Rule-derived regulatory graph G.

Complexity: $\mathcal{O}(|V^s| + |E^s|) \approx \mathcal{O}(|V^s|)$ since $|E| \propto |V|$ for the rule structure graph. In other words, the complexity is linear in the size of the rule structure graph $|V^S|$. This can be considered $\mathcal{O}(1)$ since rule structure graph sizes do not get very large (Fig S1B).

Notes: From above, $|E| = |V| - 1$ for the rule regulatory graph. Since rule sizes are bounded, rule regulatory graph sizes are bounded also (Fig S1C). So we treat $|E| \approx |V| \approx$constant for the rule-derived regulatory graph in S7.5.

Attributes of node v^s on rule structure graph of rule labeled r				NodeLabel a for atomic pattern node v_a	Label modification for edge (v_a, v_r)
NodeType	NodeLabel	NodeSide	Parents		
Molecule	m_t	Left	None	$\mathrm{m_t}$	Reactant← 1
		Right			Product ← 1
		Both		-	-
Internal State	s_t	Left	$\mathrm{m_t(c_t)}$	$\mathrm{m_t(c_t{\sim}s_t)}$	Reactant ← 1
		Right			Product ← 1
		Both			Context ← 1
Bond State	$+$	Left	$\mathrm{m_t(c_t), m'_t(c'_t)}$	$\mathrm{m_t(c_t!\,1).m'_t(c'_t!\,1)}$	Reactant← 1
		Right			Product← 1
		Both			Context←1
		Both	$\mathrm{m_t(c_t)}$	$\mathrm{m_t(c_t!+)}$	Context ←1
	$-$	Left	$\mathrm{m_t(c_t)}$	$\mathrm{m_t(c_t)}$	Reactant← 1
		Right			Product ←1
		Both			Context ← 1

S7.5 Model Regulatory Graph

Goal: Given a set of rule derived regulatory graphs, merge them into a single graph. Additionally, remap wildcard bonds (e.g. A(b!+)) to matching fully specified bonds (e.g. A(b!1).B(a!1)).

Example: Given two graphs with nodes {rule1, patt1, patt2} and {rule2, patt2, patt3} respectively, the resultant graph will have the nodes {rule1, rule2, patt1, patt2, patt3}. Edges on the resultant graph have all the labels of the corresponding edges on the individual graphs.

Input: Set of rule regulatory graphs $\{G_r | G_r \coloneqq (V_r, E_r)\}$, where r indexes the rule.

$V \leftarrow \bigcup_{\forall r} V_r$, $E \leftarrow \bigcup_{\forall r} E_r$
    edge label conflicts are resolved using Boolean OR.
for each wildcard w in V

4 for each edge (w, x) in E,
5 for each bond b in V such that w matches b
6 add edge (b, x) to E with same labels as (w, x)
7 delete edge (w, x) in E
8 delete node w in V
9 $G \leftarrow (V, E)$

Output: model regulatory graph G.
Complexity: $\mathcal{O}(\Sigma_r(|V_r| + |E_r|) + |w| * |b|)$ where V_r, E_r are nodes and edges of individual rule regulatory graphs, $|w|$ and $|b|$ are number of wildcards and bonds respectively. Since $|V_r| \approx |E_r| \approx$constant for individual rule regulatory graphs (S7.4 Notes) and wildcards are rarely used, the average complexity is $\mathcal{O}(n)$, where n is the number of rules.
Notes: Since there are only two node types, the model regulatory graph is also represented as the tuple (V_A, V_R, E) where V_A and V_R partition V into atomic patterns and rules respectively. Since $|E| \approx$ constant for individual rule regulatory graphs, we treat $|E| \propto |V_R|$ for the model regulatory graph in S7.6, S7.8.

S7.6 Removing Background on the Model Regulatory Graph
Goal: Given a model regulatory graph, remove background nodes.
Example: Given a graph {rule1, rule2, patt1, patt2}, if the set of background nodes are {rule2, patt1}, then the output graph has the nodes {rule1, patt2}. Edges between non-background nodes are transferred as is to the new graph.
Input: Model regulatory graph $G \coloneqq (V, E)$, background assignment $Bkg: V \to \{0,1\}$
1 $V' \leftarrow V - \{v | Bkg(v) = 1, v \in V\}$
2 $E' \leftarrow E - \{(v, v', EdgeLabel) | Bkg(v) + Bkg(v') \geq 1, (v, v', EdgeLabel) \in E\}$
3 $G' \leftarrow (V', E')$

Output: model regulatory graph G'
Complexity: $\mathcal{O}(|V| + |E|)$, i.e. linear in the size of the model regulatory graph.

S7.7 Determining Edge Signature of a Rule
Goals: For a particular rule node on the model regulatory graph, compute a signature from adjacent nodes, edges and edge attributes.
Example: Given a graph {rule1, patt1, patt2, patt3} with edges { (rule1,patt1,100), (rule1,patt2,010) , (rule1,patt3,001) }, where edge label 100 indicates edge attributes Reactant=1,Product=0,Context=0, the edge signature for rule1 is (patt1):(patt2):(patt3). Suppose a partial grouping function is provided on patterns, say patt1->pattgroup1 and patt2->pattgroup2, then the edge signature is (pattgroup1):(pattgroup2):(patt3).
input, Rule $v_r \in V_R$. model regulatory graph $G_M \coloneqq (V_A, V_R, E_M)$, partial grouping function on atomic patterns $f: V_A \nrightarrow Labels$.
1 initialize empty sets Re, Pr, Co
2 for each edge $(v_a, v_r, EdgeLabel)$ in E_M
3 string $S \leftarrow v_a$ if v_a not in domain of f
4 string $S \leftarrow f(v_a)$ if v_a in domain of f
5 add S to Re if Reactant=1 in $EdgeLabel$
6 add S to Pr if Product=1 in $EdgeLabel$
7 add S to Co if Context=1 in $EdgeLabel$
8 string $S_1 \leftarrow$ sorted and concatenated elements of Re
9 string $S_2 \leftarrow$ sorted and concatenated elements of Pr
10 string $S_3 \leftarrow$ sorted and concatenated elements of Co
11 string $Sgn(v_r) \leftarrow S_1 : S_2 : S_3$

Output: Edge signature $Sgn(v_r)$

Complexity: $\mathcal{O}\left(\left(\frac{|E_M|}{|V_R|}\right)^2\right) \approx \mathcal{O}(1)$, since $|E_M| \propto |V_R|$ on the model regulatory graph (S7.5 Notes).

Notes A more permissive edge signature can be obtained by discounting context edges, i.e. $Sgn(v_r) \leftarrow$ concatenated $S_1 : S_2$ only.

S7.8: Grouping on the Model Regulatory Graph

Goal: Given a model regulatory graph and an optional atomic pattern grouping, group rules according to edge signature.

Example Given a graph {rule1, rule2, patt1, patt2} with edges { (rule1,patt1,product), (rule2,patt2,product) }, if pattgroup1={patt1,patt2}, then assign rule1 and rule2 to the same group rulegroup1.

Input: Model regulatory graph $G_M \coloneqq (V_A, V_R, E_M)$, partial group assignment $GrpA: V_A \nrightarrow Labels$ for atomic patterns.

1 for each node v_r in V_R
2 $Sgn(v_r) \leftarrow$ Edge signature of v_r given function $GrpA$ using S7.6
3 if $Sgn(v_r)$ not in domain of $count$
4 $count(Sgn(v_r)) \leftarrow 0$
5 increment $count(Sgn(v_r))$
6 if $count(Sgn(v_r)) = 2$
7 $ind(Sgn(v_r)) \leftarrow x$, where x is a unique label
8 for each node v_r in V_R such that $count(Sgn(v_r)) > 1$
9 $GrpR(v_R) \leftarrow ind(Sgn(v_r))$

Output: partial group assignment $GrpR: V_R \nrightarrow Labels$ for rules.

Complexity: $\mathcal{O}(|V_R|)$, i.e. linear in the number of rules.

Notes: If the edge signature is built only from reactant and product edges and does not use context edges, then the grouping is more permissive and fewer groups are obtained having larger group sizes.

S7.9: Collapsing Groups on the Model Regulatory Graph

Goal: Given a model regulatory graph with groups, replace each group of nodes by a single node labeled with the group name.

Example: Given a graph with nodes {patt1,patt2,patt3,rule1,rule2,rule3} in which pattgroup1 = {patt1, patt2}, rulegroup1 = {rule1, rule2}, the new graph has nodes {pattgroup1, patt3, rulegroup1, rule3}. Edges incident on either patt1 or patt2 are remapped to pattgroup1 and edges incident on rule1 or rule2 are remapped to rulegroup1.

Input: Model regulatory graph $G_M \coloneqq (V_A, V_R, E_M)$, partial group assignments $GrpA, GrpR$ on V_A, V_R respectively.

1 initialize empty sets V, E and function $Remap$
2 for each node v in $V_A \cup V_R$
3 make node x from v according to Table below
4 add node x to V
5 $Remap(v) \leftarrow x$
6 for each edge $(v_a, v_r, EdgeLabel)$ in E_M
7 add edge $(Remap(v_a), Remap(v_r), EdgeLabel)$ to E
8 $G \leftarrow (V, E)$

Output: collapsed regulatory graph G.

Complexity: $\mathcal{O}(|V_A| + |V_R| + |E_M|) \approx \mathcal{O}(|V_A| + |V_R|)$, since $|E_M| \propto |V_R|$, i.e. linear in the size of the model regulatory graph $|V_A| + |V_R|$.

NodeType of v	v ∈domain of $GrpA$	v ∈domain of $GrpR$	$x =$(NodeLabel,NodeType,isGroup)
atomic pattern	True	-	($GrpA(v)$, atomic pattern, True)
	False	-	(v, atomic pattern, False)
rule	-	True	($GrpR(v)$, rule, True)
	-	False	(v, rule, False)

S8 Appendix: Rendering Conventions

S8.1 Site Graph

Goal: Given a pattern structure graph, draw a site graph by nesting components within molecules, internal states within components and drawing bonds as edges between components.

Example: Given a pattern A(b!1).B(a~x!1), nest b within A, a within B, x within a and add an edge between a and b.

Input: Pattern structure graph $G \coloneqq (V, E)$

```
for each node v in V
        if type={molecule,component,internalstate} and name=S
                render as node labeled S
for each node v in V such that type=component
        if exists (v, v') in E such that
                type=molecule for v', then nest v in v'
                type=internalstate for v', then nest v' in v
for each node v in V such that type=bond
        ignore if name=−
        if name=+ and adjacent to two components c, c'
                render as edge between renderings of c, c'
        if name=+ and adjacent to only one component c
                render as node labeled +
                add edge to rendering of c
```

Output: Site graph

S8.2 Compact Rule Visualization

Goal: Given a rule structure graph, draw a site graph with the nodes labeled side=both and render nodes labeled side=left or side=right with special conventions.

Example: Given a rule A(b~x)-> A(b~y) and a rule structure graph {A,b,-,x,y} nest b within A, x within b, y within b, add ChangeState node, add directed edge from x to ChangeState, add directed edge from ChangeState to y.

Input: rule structure graph $G \coloneqq (V, E)$

```
for each node v in V
        if side=both or type=internal state, render using site graph conventions
        if type=molecule and side=left
                add node labeled DeleteMol
                add directed edge from v to DeleteMol
        if type=molecule and side=right
                add node labeled AddMol
                add directed edge from to AddMol to v
        if type=bond and name=+ and side=left
                replace with node labeled DeleteBond
                add directed edges from adjacent components to DeleteBond
        if type=bond and name=+ and side=right
                replace with node labeled AddBond
                add directed edges to adjacent components from AddBond
        if type=component and exists two adjacent states s, s'
                if side=left for s and side=right for s'
                        add node labeled ChangeState
```

18	nest ChangeState node within v
19	add directed edge from s to ChangeState
20	add directed edge from ChangeState to s'

Output: compact rule visualization

S8.3 Regulatory Graph

Goal: Given a regulatory graph draw each node and edge according to provided conventions for atomic pattern and rule node types and reactant, product and context edge types. If a grouping scheme is provided, draw groups around the respective sets of nodes.

Example: Given graph {patt1, patt2, patt3, rule1, rule2, rule3} and groups pattgroup1={patt1,patt2}, rulegroup1={rule1,rule2}, draw nodes {patt1, patt2, patt3} using atomic pattern conventions, nodes {rule1, rule2, rule3} using rule node conventions, node pattgroup1 around {patt1,patt2} and node rulegroup1 around {rule1,rule2}.

Input: Regulatory graph $G \coloneqq (V, E)$, two node rendering conventions for atomic pattern and rule respectively, three node rendering conventions for reactant, product and context respectively, and optionally a partial group assignment $Grp: V \to Labels$.

1	for each label s in $img(Grp)$
2	draw node labeled s
3	for each node v in V
4	if type=atomicpattern and name=S
5	render with atomic pattern node conventions and label S
6	if type=rule and name=S
7	render with rule node conventions and label S
8	if $Grp(v) = s$
9	nest within node labeled s
10	for each edge (v, v') in E
11	if type of v is atomicpattern and type of v' is rule
12	if re=1, draw directed edge from v to v' with reactant edge conventions
13	if pr=1, draw directed edge from v' to v with product edge conventions
14	if co=1, draw directed edge from v to v' with context edge conventions
15	allow multiple edges between the same pair of nodes

Output: Rendered regulatory graph