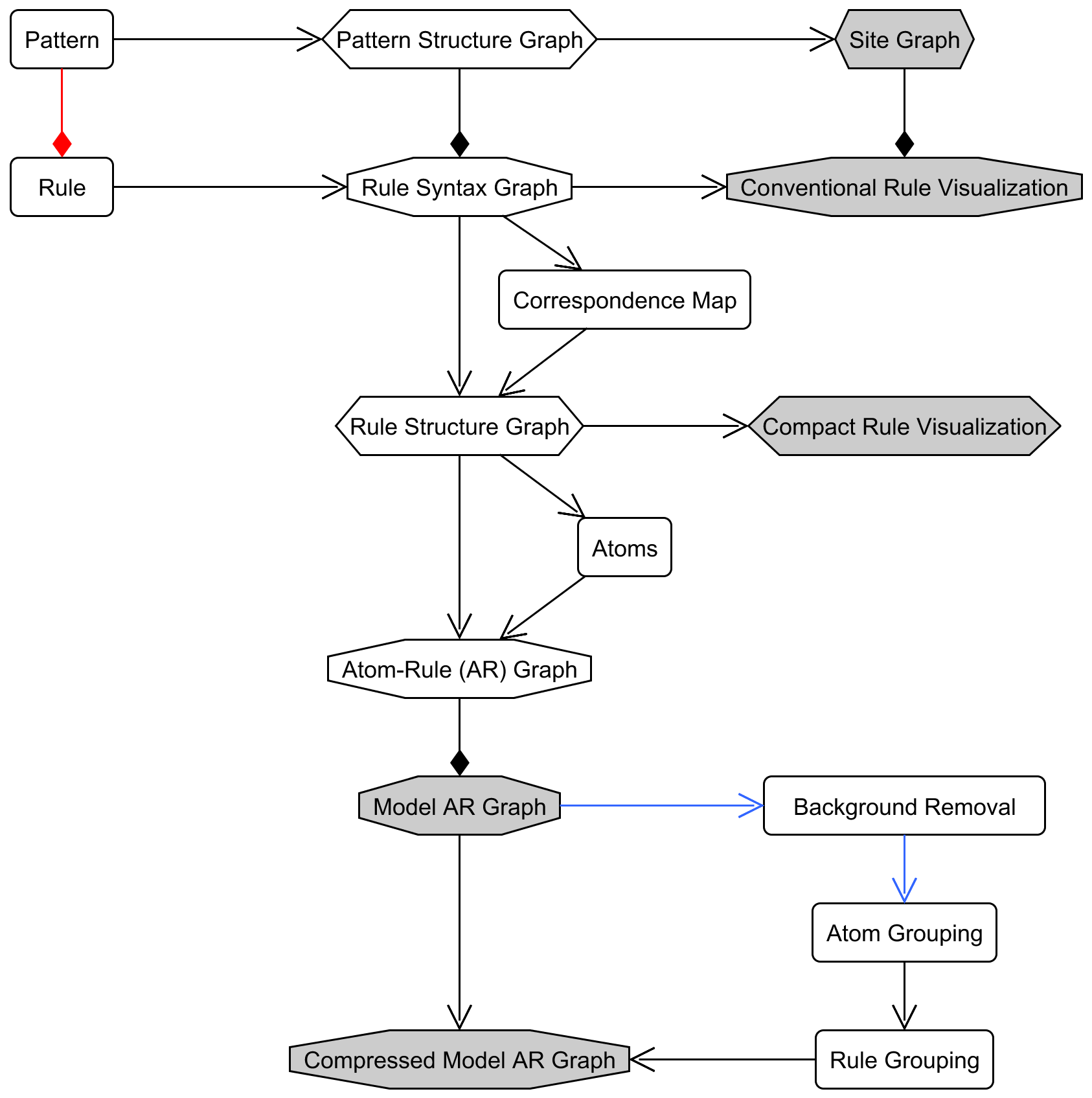
S1 Appendix: Extended Methods

# Index

S1.1: Extended methods  
 Approach  
 Intermediary Graph Types  
 Visualizing Structure  
 Visualizing Mechanism  
 Visualizing Regulation  
 Visualizing Function  
 Algorithm Time Complexity  
 Heuristics for Background Removal  
 Heuristics for Atom Grouping  
 Figures  
 F1: Summary of Methods  
 F2: Intermediary Graphs and Conventional Visualizations  
 F3: Compact Rule Visualization  
 F4: Atom-Rule Graph  
 F5: Complexity Reduction Approaches  
 F6: Rule size distributions  
S1.2: Algorithms  
 A1: Pattern Structure Graph  
 A2: Correspondence Map  
 A3: Rule Structure Graph  
 A4: Atom-Rule (AR) Graph  
 A5: Model AR Graph  
 A6: Background Removal  
 A7: Edge Signature of a Rule  
 A8: Grouping Atoms and Rules  
 A9: Collapsing Groups  
S1.3: Rendering Conventions  
 C1: Site Graph  
 C2: Compact Rule Visualization  
 C3: AR Graph

S1.1: Extended Methods

**Approach:** The goal of this work is to formally derive useful visualizations of rule-based models. The input for these methods include BioNetGen patterns and rules, whose formal definitions have been provided in previous literature (see Supplement in [1]). These are converted to intermediary graphs, which are graph data structures defined in this work. Diagrams are generated by applying rendering conventions to the nodes and edges of the intermediary graphs. Fig F1 provides a broad summary. In this section, we briefly discuss the data structures, algorithms and rendering conventions. For detailed specifications on algorithms and rendering conventions, refer to Sections S7.2 and S7.3 respectively.



**Fig F1 Summary of methods.** The graphs in this work are of two main types, structure graph (hexagon) or network graph (octagon). Objects are either transformed or generated from other objects (open arrowhead), or composed from multiple instances of other objects (diamond arrowhead). Each process is either user-specified (red), automated (black) or semi-automated (blue). In the model specification, patterns compose rules. For visualization, they are transformed to pattern structure graphs and rule syntax graphs respectively, which are visualized along conventional lines. In this work, we use the correspondence map synthesized from the rule syntax graph to create the rule structure graph, which can be visualized compactly. Also, the rule structure graph is examined to identify instances of atoms, which are then used to build the atom-rule (AR) graph. AR graphs of individual rules are aggregated into the model AR graph. Complexity reduction techniques (background removal, atom grouping, rule grouping and group collapsing) are sequentially applied to generate the compressed model AR graph.

**Intermediary Graph Types:** Intermediary graphs are of two broad types: structure graphs and network graphs (Fig F2A). **Structure graph** denotes a node-labeled undirected graph, with node attributes including but not limited to { NodeIndex, NodeLabel, NodeType }. Multiple nodes with the same NodeLabel and NodeType are allowed, but not with the same NodeIndex. Structure graphs are used to represent objects with predominantly structural information. **Network graph** denotes an edge-labeled node-labeled bipartite graph. Node attributes are { NodeLabel, NodeType } with NodeLabel being used to uniquely identify nodes. Network graphs are allowed to have only two node types with one node type referring to a structure and the other to a process. Edges are bipartite, i.e. they can only be from a structure node to a process node and not between the same node types. Edges on the network graph have one or more binary attributes that are typically mapped to visual attributes such as color and direction. Network graphs are used to represent the interaction of processes with structures. Structure graphs and network graphs have precedent in hierarchical graphs [2] and *rxncon* regulatory graphs [3] respectively.



**Fig F2. Intermediary graphs and conventional visualizations. (A)** Structure graphs ({NodeIndex, NodeLabel, NodeType} as node attributes) and network graphs ({NodeLabel, NodeType} as node attributes) are the two types of graph data structures used in this work. Edges on the network graph are bipartite and have one or more edge attributes that can take binary values. **(B)** Patterns are represented by the pattern structure graph, which has molecule, component, binding state and internal state as values for NodeType (denoted M, C, B, S respectively). To visualize a pattern as a site graph, hierarchically nest molecules (nodes 1,2), components (3,4) and internal states (8), show bonds (6) as edges between components, and ignore unbound states (7) **(C)** The rule as specified in the model is represented by the rule syntax graph, a network graph in which one node type embeds pattern structure graphs and the other is labeled with a rule name. To produce a conventional rule visualization, render embedded patterns as site graphs and use edge direction to show reactant and product relationships.

**Visualizing Structure:** The **pattern** is the BioNetGen object that is used to specify molecular structures. It is built from molecules, components of molecules, internal states of components and bonds between components [1], [4], [5]. Here, we represent the pattern as the **pattern structure graph,** in which each molecule, component, internal state and bond state (bond or unbound state) is represented as a node (Fig F2B, Algorithm A1) ([1], [2], [6] have similar graph formalisms) . To visualize the pattern as a **site graph** (Fig F2B), nest molecules, components and internal states hierarchically and show bonds as edges between components (Rendering Convention C1). The chemical species and the contact map are both special cases of the pattern structure graph and have the same rendering conventions.

**Visualizing Mechanism**: The **reaction rule** is the BioNetGen object that is used to specify kinetic processes. It contains patterns specified as reactants and products [1], [4], [5]. The **rule syntax graph** is a network graph in which one type of node embeds pattern structure graphs, the other type of node is labeled with a rule name, and edges between the nodes are typed as reactant or product (Fig F2C). This encodes the reaction rule as it is specified in the model. To achieve a **conventional rule visualization** (Fig F2C), render the embedded patterns as site graphs and use edge direction to show whether the relationship is reactant or product (pattern to rule = reactant, rule to pattern = product).

To achieve a **compact rule visualization** (Fig F3A), we first build a correspondence map between left and right sides of the rule (Algorithm A2). Then, we merge the shared structures between reactant and product sides of the rule to produce a single graph called the **rule structure graph** (Algorithm A3). The resultant nodes inherit the property of whether they originate from reactant side or product side or both, which is represented as the attribute NodeSide. This explicitly differentiates modified structures from unmodified ones. Now, render unmodified nodes on the rule structure graph using site graph conventions and use graph operation nodes to emphasize the modified structures (Rendering convention C2). Five types of graph operations are permitted in BioNetGen rules (Fig F3B): adding and removing bonds (AddBond/DeleteBond), creating and destroying whole molecules (AddMol, DeleteMol) and changing the internal state label of a component (ChangeState). A rule may have more than one graph operation occurring simultaneously. Edges on the graph operation node also indicate whether the operation consumes or produces structures.



**Fig F3. Compact rule visualization. (A)** Given a rule syntax graph, BioNetGen maps out the shared structures between reactants and products by building a correspondence map (dashed lines). We use the correspondence map to merge reactants and products into a single rule structure graph. Nodes on this graph are also labeled with a side of origin: reactant only (red node border), product only (blue) or both (gray). To visualize rules compactly, render the rule structure graph as a site graph (nodes 1-6) and use graph operation nodes (AddBond) to emphasize the modified structures (node 7). **(B)** BioNetGen has five basic graph operations: adding and removing bonds, creating and deleting molecules and changing state labels of molecular sites. A rule may have one or more instances of these graph operations. In the main text, we hide labels of operation nodes.

**Visualizing Regulation:** We introduce the notion of **atom**, a special type of pattern that specifies only a single distinct type of modifiable structure. Atoms include free binding sites and bonds (modifiable by AddBond/DeleteBond), types of molecules (modifiable by AddMol/DeleteMol) and internal states (modifiable by ChangeState). Atoms are determined by examining specific source nodes on rule structure graphs (Fig F4A, Algorithm A4). Since each node on the rule structure graph has a NodeSide attribute (reactant/product/both), this is translated as a relationship between the matched atom at that node and the rule itself (reactant/product/context). These relationships are represented as binary labels on bipartite edges between the rule and the atoms, resulting in the network graph called the **atom-rule graph** (Fig F4A, Algorithm A4), or AR graph. To visualize AR graphs (Rendering Conventions C3), we apply distinct visual attributes to the two node types (atom, rule) and the three edge types (reactant, product, context). Individual rule AR graphs are merged and processed to build the **model AR graph** (Fig F4B, Algorithm A5). The rules in a model draw from the same finite pool of structure types and binding interactions, so the model AR graph grows primarily with the number of rules in the model.



**Fig F4. Atom-Rule (AR) Graph. (A)** An atom represents a distinct type of modifiable structure: a free binding site, a bond, an internal state or a molecule type. It has a name specified using pattern syntax (brackets = containership, !{tag}=bond, ~{label}=internal state). Atoms are identified by examining source nodes on the rule structure graph (bold arrows). The atom-rule (AR) graph is a network graph with edges between atoms and a node labeled with the rule name. The NodeSide attribute of the source node (border color red=left, blue=right, gray=both) determines edge attributes on the AR graph (reactant, product and/or context). Context edges are rendered with a light color and reactant/product edges have a dark color. **(B)** Aggregating AR graphs of individual rules results in the model AR graph.

**Visualizing Function**: To build compact pathway diagrams, the elements of the model AR graph have to be organized and simplified. This involves four steps: removing background nodes, grouping atoms, grouping rules and compressing groups. Background removal consists of removing atoms and rules that are redundant for model comprehension (Fig. F5B, Algorithm A6). This greatly reduces the size of the graph. Grouping atoms involves identifying similarity between sites in the system, e.g., phospho-sites on the same molecule or bonds between the same pair of molecules (Fig F5C). Grouping rules requires computing similarity between rules based on their edge signatures, i.e., based on how each rule interacts with groups of atoms (Fig F5D, Algorithms A7 and A8). Compressing groups involves replacing groups of nodes by a representative group node and combining edges incident on individual members of the group and remapping them to the representative group node (Fig F5E, Algorithm A9). Background removal and atom grouping are performed using default heuristics whose choices can be modified with user input, then rule grouping is performed automatically based on those settings. The edge signature can be strict or permissive, i.e. it considers all three edge types or it considers only reactant and product edges and ignores context edges respectively. Using a permissive edge signature results in identifying a broader grouping of rules.

****

**Fig F5. Complexity reduction approaches. (A)** The full model AR graph from Fig F4. **(B)** Nodes redundant for human comprehension are tagged as background and removed. Here, that includes free binding sites and the unbinding rule. **(C)** Atoms are classified into groups. Here, the groups used are Enz|Sub to denote the enzyme-substrate bond and Sub\_p to denote the phosphorylated sites. Steps (B) and (C) can be performed by a heuristic that can be modified with user annotation. **(D)** Rules are classified into groups by an automated comparison of their edge signatures. Here, rules R2 and R3 have similar edge signatures: a context edge from Enz|Sub and a product edge into Sub\_p, so they are grouped under the same label RG0. **(E)** Each group of nodes is replaced by a single representative node and edges incident on individual members of the group are merged together.

**Algorithm Time Complexity**: If *p* represents the size of a pattern, building a pattern structure graph from the pattern syntax is *O(p)* (Algorithm A1). If *mL, mR* represent the size of patterns on left and right sides of a rule respectively, building a correspondence map between reactants and products before merging is *O(mL\*mR)* (Algorithm A2). Building the rule structure graph from the correspondence map (Algorithm A3) and building the atom-rule graph from the rule structure graph (Algorithm A4) are both *O(m)* where *m* represents the size of the rule. Since rules are typically finite and bounded in size (Fig F6), the contribution of rule size can be assumed to be *O(1)*. So, building the AR graph (Algorithm A5), removing background (Algorithm A6), identifying rule groups using edge signatures (Algorithms A7 and A8) and collapsing group nodes (Algorithms A9), are all *O(n)*, where *n* is the number of rules.



**C**

**BA**

**A**

**Fig F6. 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)** atom-rule graph for each rule. In all three representations, it is seen that large rules are infrequently encountered.

**Heuristics for Background Removal:** The default heuristic makes the following choices for including or excluding atoms/rules from the background

1. All free binding sites are included in background. All bonds are excluded.
2. When internal states are encountered on a component, the first such internal state is included in background. Typically, this would assign default states such as unphosphorylated state to the background.
3. All unidirectional rules are excluded from background.
4. For bidirectional rules, the forward rule is excluded from background, but the reverse rule is included.
5. Molecule atoms associated with molecule creation/deletion rules are excluded from background.

**Heuristics for Grouping Atoms:** The default heuristic for atom grouping makes the following choices:

1. Bonds are grouped based on the molecules they link, e.g., X(a!1).Y(b!1) is placed in the group X|Y.
2. Internal states are grouped based on the molecule they are present in and the states they represent, e.g., X(a~p) is placed in the group X\_p.

**References**

[1] J. S. Hogg, L. A. Harris, L. J. Stover, N. S. Nair, and J. R. Faeder, “Exact hybrid particle/population simulation of rule-based models of biochemical systems.,” *PLoS Comput. Biol.*, vol. 10, no. 4, p. e1003544, Apr. 2014.

[2] N. W. Lemons, B. Hu, and W. S. Hlavacek, “Hierarchical graphs for rule-based modeling of biochemical systems.,” *BMC Bioinformatics*, vol. 12, p. 45, 2011.

[3] C.-F. Tiger, F. Krause, G. Cedersund, R. Palmér, E. Klipp, S. Hohmann, H. Kitano, and M. Krantz, “A framework for mapping, visualisation and automatic model creation of signal-transduction networks.,” *Mol. Syst. Biol.*, vol. 8, no. 578, p. 578, Jan. 2012.

[4] J. A. Sekar and J. R. Faeder, “Rule-based modeling of signal transduction: a primer.,” *Methods Mol. Biol.*, vol. 880, pp. 139–218, 2012.

[5] J. R. Faeder, M. L. Blinov, and W. S. Hlavacek, “Rule-based modeling of biochemical systems with BioNetGen,” *Methods Mol. Biol.*, vol. 500, no. 1, pp. 113–167, 2009.

[6] M. Blinov, J. Yang, J. R. Faeder, and W. S. Hlavacek, “Graph theory for rule-based modeling of biochemical networks,” *Trans. Comput. Syst. Biol. VII*, vol. 4230, pp. 89–106, 2006.

S1.2: Algorithms

**A1 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 and a map molecule, component, internal state, bond}. Each element in has one of the following forms:

is a molecule with index of type ,

, a component with index of type in molecule ,

is an internal state with index of type on component ,

is a bond with index between components and ,

is a wildcard bond with index on component

|  |  |
| --- | --- |
| **1 2**  **3**  **4**  **5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20**  **21**  **22**  **23** | initialize empty sets and function  for each in ,  if and molecule  add node (,,molecule) to  if and component  add node (, component) to  add edg e to  False  if and internal state  add node (,,internal state) to  add edge to  if and bond  add node (,, bond state) to  add edges to  True  if  add node (,,bond state) to  add edge to  True  for each component in such that False  add node (,,bond state) to , where is a unique id  add edge to |

**Output:** Pattern structure graph .

**Complexity:**  , i.e. linear in the size of the pattern.

**Notes:** From above, . We will treat for pattern structure graphs in A2 and A3.

**A2 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 where , and { molecule, component, internal state, bond }. and are merged patterns of the left and right sides of the rule respectively.

|  |  |
| --- | --- |
| **1** | , a sequence with molecules drawn from |
| **2** | , a sequence with molecules drawn from |
| **3** | for each such that |
| **4** | , a sequence with components drawn from s.t. |
| **5** | for each such that |
| **6** | , a sequence with components drawn from s.t. |
| **7** | initialize empty sets , and function |
| **8** | for each such that and not in |
| **9** | for each such that and and not in |
| **10** | if exists preserving component type, presence of internal state, wildcard |
| **11** | add to and to |
| **12** |  |
| **13** | for each molecule map in |
| **14** | for each in such that and not in |
| **15** | for each in such that and and not in |
| **16** | add to and to |
| **17** |  |
| **18** | for each component map in such that |
| **19** | if exists such that |
| **20** | add to and to |
| **21** |  |
| **22** | if exists such that |
| **23** | add to and to |
| **24** |  |
| **25** | for each bond such that and not in |
| **26** | for each bond such that and not in |
| **27** | if exists in |
| **28** | add to and to |
| **29** |  |

**Output:** Correspondence map , where indicates that it is a partial map.

**Complexity:** . This can be considered as large rule sizes are rare (Fig F6A).

**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.

**A3 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 where are merged pattern structure graphs and is a correspondence map.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15** | initialize empty sets  for each node in , label NodeSide()=left  for each node in , label NodeSide()=right  for each map in ,  label NodeSide() = NodeSide() = both  for each edge in , add edge to  for each edge in  if NodeSide()=NodeSide()=both  next edge  if NodeSide()=both    if NodeSide()=both    add edge to |

**Output:** Rule structure graph

**Complexity:**  since for pattern structure graphs. The complexity is linear in the size of the rule . This can be considered as large rule sizes are rare (Fig F6A).

**Notes:** The rule structure graph has the same property of the pattern structure graph, i.e. .

**A4 Atom-Rule Graph**

**Goal:** The atom-rule (AR) 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 atom or rule and NodeLabel is sufficient to index nodes uniquely. Given a rule structure graph of a rule, synthesize an AR 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 A3 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 atom. 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 where is a label indexing the rule.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18** | for each node in , initialize empty set Parent()  for each edge in  if is a molecule with NodeLabel  if is a component  add label to Parent(’)  if is a component with NodeLabel and parent  if is an internal or bond state  add label to Parent(’)  initialize empty sets  add node =(,rule) to  for each node in  determine atom NodeLabel from Table below  make node (, atom),  if not in ,  add node to  add edge to with default labels (Reactant=0,Product=0,Context=0)  apply label modification of edge from Table |

**Output:** Atom-rule graph .

**Complexity:** since for the rule structure graph. In other words, the complexity is linear in the size of the rule structure graph . This can be considered as large rule structure graph sizes are rare (Fig F6B).

**Notes:** From above, for the atom-rule graph of an individual rule. Since rule sizes are bounded, atom-rule graph sizes are bounded also (Fig F6C). So we treatconstant for the individual atom-rule graphs in A5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes of node on rule structure graph of rule labeled | | | | NodeLabel for atom node | Label modification for edge |
| NodeType | NodeLabel | NodeSide | Parents |
| Molecule |  | Left | None |  | Reactant 1 |
| Right | Product 1 |
| Both | - | - |
| Internal State |  | Left |  |  | Reactant 1 |
| Right | Product 1 |
| Both | Context 1 |
| Bond State |  | Left |  |  | Reactant 1 |
| Right | Product 1 |
| Both | Context1 |
| Both |  |  | Context 1 |
|  | Left |  |  | Reactant 1 |
| Right | Product 1 |
| Both | Context 1 |

**A5 Model AR Graph**

**Goal:** Given a set of rule atom-rule graphs of individual rules, 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, atom1, atom2} and {rule2, atom2, atom3} respectively, the resultant graph will have the nodes {rule1, rule2, atom1, atom2, atom3}. Edges on the resultant graph have all the labels of the corresponding edges on the individual graphs.

**Input:** Set of atom-rule graphs , where indexes the rule.

|  |  |
| --- | --- |
| **1** |  |
| **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9** | edge label conflicts are resolved using Boolean OR.  for each wildcard in  for each edge in ,  for each bond in such that matches  add edge to with same labels as  delete edge in  delete node in |

**Output:** model AR graph .

**Complexity:**  where are nodes and edges of AR graphs of individual rules indexed by , and are number of wildcards and bonds respectively. Since constant for individual atom-rule graphs (A4 Notes) and wildcards are rarely used, the average complexity is , where is the number of rules.

**Notes:** Since there are only two node types, the model AR graph is also represented as the tuple where and partition into atoms and rules respectively. Since constant for AR graphs of individual rules, we treat for the model AR graph in A6 and A7.

**A6 Background Removal**

**Goal:** Given a model AR graph, remove background nodes.

**Example:** Given a graph {rule1, rule2, atom1, atom2}, if the set of background nodes are {rule2, atom1}, then the output graph has the nodes {rule1, atom2}. Edges between non-background nodes are transferred as is to the new graph.

**Input:** Model AR graph , background assignment

|  |  |
| --- | --- |
| **1**  **2**  **3** |  |

**Output:** model AR graph

**Complexity:** , i.e. linear in the size of the model AR graph.

**A7 Edge Signature of a Rule**

**Goals:** For a particular rule node on the model AR graph, compute a signature from adjacent nodes, edges and edge attributes.

**Example:** Given a graph {rule1, atom1, atom2, atom3} with edges { (rule1,atom1,100), (rule1,atom2,010) , (rule1,atom3,001) }, where edge label 100 indicates edge attributes Reactant=1,Product=0,Context=0, the edge signature for rule1 is (atom1):(atom2):(atom3). Suppose a partial grouping function is provided on patterns, say atom1->atomgroup1 and atom2->atomgroup2, then the edge signature is (atomgroup1):(atomgroup2):(patt3).

**input**, Rule . model AR graph , partial grouping function on atoms  
 .

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11** | initialize empty sets  for each edge in  string if not in domain of  string if in domain of  add to if Reactant=1 in  add to if Product=1 in  add to if Context=1 in  string sorted and concatenated elements of  string sorted and concatenated elements of  string sorted and concatenated elements of  string |

**Output:** Edge signature

**Complexity:** , since on the model AR graph (A5 Notes).

**Notes** A more permissive edge signature can be obtained by discounting context edges, i.e. concatenated only.

**A8: Grouping Atoms and Rules**

**Goal:** Given a model AR graph and an optional atom grouping, group rules according to edge signature.

**Example** Given a graph {rule1, rule2, atom1, atom2} with edges { (rule1,atom1,product), (rule2,atom2,product) } and atomgroup1={atom1,atom2}, assign rule1 and rule2 to the same group rulegroup1.

**Input:** Model AR graph , partial group assignment for atoms.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9** | for each node in  Edge signature of given function using S7.6  if not in domain of  increment  if  , where is a unique label  for each node in such that |

**Output:** partial group assignment for rules.

**Complexity:** , 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 with larger group sizes.

**A9: Collapsing Groups**

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

**Example:** Given a graph with nodes {atom1,atom2,atom3,rule1,rule2,rule3} in which atomgroup1 = {atom1, atom2}, rulegroup1 = {rule1, rule2}, the new graph has nodes {atomgroup1, atom3, rulegroup1, rule3}. Edges incident on either atom1 or atom2 are remapped to atomgroup1 and edges incident on rule1 or rule2 are remapped to rulegroup1.

**Input:** Model AR graph , partial group assignments on respectively.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8** | initialize empty sets and function  for each node in  make node from according to Table below  add node to  for each edge in  add edge to |

**Output:** merged AR graph .

**Complexity:** , since , i.e. linear in the size of the model AR graph .

|  |  |  |  |
| --- | --- | --- | --- |
| NodeType of | domain of | domain of | (NodeLabel,NodeType,isGroup) |
| atom | True | - | (, atom, True) |
| False | - | (, atom, False) |
| rule | - | True | (, rule, True) |
| - | False | (, rule, False) |

S1.3: Rendering Conventions

**C1 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

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14** | for each node in  if type={molecule,component,internalstate} and name=S  render as node labeled S  for each node in such that type=component  if exists in such that  type=molecule for , then nest in  type=internalstate for , then nest in  for each node in such that type=bond  ignore if name=  if name= and adjacent to two components  render as edge between renderings of  if name= and adjacent to only one component  render as node labeled  add edge to rendering of |

**Output:** Site graph

**C2 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

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20** | for each node in  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 to DeleteMol  if type=molecule and side=right  add node labeled AddMol  add directed edge from to AddMol to  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  if side=left for and side=right for  add node labeled ChangeState  nest ChangeState node within  add directed edge from to ChangeState  add directed edge from ChangeState to |

**Output:** compact rule visualization

**C3 Atom-Rule Graphs**

**Goal:** Given an AR graph draw each node and edge according to provided conventions for atom 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 {atom1, atom2, atom3, rule1, rule2, rule3} and groups atomgroup1={atom1, atom2}, rulegroup1={rule1,rule2}, draw nodes {atom1, atom2, atom3} using atom conventions, nodes {rule1, rule2, rule3} using rule node conventions, node atomgroup1 around {atom1, atom2} and node rulegroup1 around {rule1,rule2}.

**Input:** AR graph , two node rendering conventions for atom and rule respectively, three edge rendering conventions for reactant, product and context respectively, and optionally a partial group assignment .

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15** | for each label in  draw node labeled  for each node in  if type=atom and name=S  render with atom node conventions and label S  if type=rule and name=S  render with rule node conventions and label S  if  nest within node labeled  for each edge in  if type of is atom and type of is rule  if re=1, draw directed edge from to with reactant edge conventions  if pr=1, draw directed edge from to with product edge conventions  if co=1, draw directed edge from to with context edge conventions  allow multiple edges between the same pair of nodes |

**Output:** Rendered AR graph