A Formal Semantics for the BioNetGen Language: Category-Theoretic Foundations, Stochastic Dynamics, and Causal Structure

Claude
Anthropic

January 28, 2026

Abstract

We present a comprehensive formal semantics for the BioNetGen Language (BNGL), a rule-based modeling language for biochemical reaction networks. Our treatment unifies several mathematical perspectives: (1) *category-theoretic foundations* via double-pushout and sesqui-pushout graph rewriting in adhesive categories, establishing BNGL's place within algebraic graph transformation theory; (2) *atoms* as elementary structural features with linear-time construction of atom-rule graphs encoding regulatory interactions; (3) *stochastic semantics* via rule algebras yielding continuous-time Markov chain dynamics; (4) *chemical reaction network theory* connecting rule structure to deficiency and persistence properties; (5) *bisimulation equivalences* enabling exact model reduction; (6) *causal semantics* in the sense of Pearl's interventional calculus; and (7) *temporal logic specifications* supporting statistical model checking. We provide three complementary semantic treatments—denotational semantics via network generation, small-step operational semantics for network-free simulation, and hybrid particle/population semantics with dynamic regime classification—and prove that all three produce identical stochastic dynamics under appropriate well-formedness conditions. This framework connects BNGL to the Kappa language through their shared categorical semantics while clarifying semantic differences, provides foundations for formal verification and certified code generation, and establishes the mathematical basis for compositional reasoning about rule-based models.

Contents

1 Introduction

Rule-based modeling has emerged as a powerful paradigm for representing biochemical systems exhibiting combinatorial complexity [4]. Rather than enumerating all possible molecular species and reactions explicitly, rule-based approaches specify transformation patterns that apply to classes of molecules sharing common structural features. The BioNetGen Language (BNGL) [1, 2, 3] is among the most widely-used rule-based modeling languages, with applications spanning signal transduction, immune receptor signaling, and gene regulatory networks.

Despite widespread adoption, BNGL has lacked a complete formal semantics that precisely defines the meaning of language constructs in mathematical terms. Previous work has described operational aspects of BioNetGen [2, 3] and developed visualization formalisms [7], but a unified formal treatment connecting syntax, static semantics, and multiple execution models has been absent. This gap impedes formal verification of models, certified code generation, and rigorous comparison between simulation strategies.

1.1 Contributions

This paper provides the first comprehensive formal semantics for BNGL with the following contributions:

Atoms as Semantic Primitives. We introduce *atoms*—elementary structural features including molecule existence, component states, free binding sites, and inter-molecular bonds—as the minimal semantic units of BNGL (Definition 5.1). Atoms provide a canonical decomposition of patterns that factors out syntactic redundancy and enables efficient analysis of rule interactions. We prove that every pattern admits a unique atom decomposition up to canonical ordering (Proposition 5.1).

Atom-Rule Graphs. We develop the *atom-rule (AR) graph*, a bipartite directed graph with atoms and rules as nodes and three edge types encoding consumption, production, and context requirements (Definition 5.8). Unlike rule influence diagrams requiring $O(|\mathcal{R}|^2)$ pairwise comparisons, AR graphs can be constructed in $O(|\mathcal{R}| \cdot k_{\max})$ time where $k_{\max}$ is the maximum pattern size, while encoding complete information about direct rule interactions (Theorem 5.3). We establish that paths in the AR graph correspond exactly to causal chains in the induced dynamics (Theorem 5.4).

Primitive Graph Operations. We formalize exactly five primitive operations—AddBond, DeleteBond, ChangeState, AddMol, and DeleteMol—that constitute the complete transformation language for BNGL rules (Definition 7.1). We prove that every well-formed rule admits a unique operation sequence and that operation application commutes appropriately (Lemma 7.3).

Three-Way Semantic Equivalence. We provide three complementary semantic treatments and prove their equivalence:

(i) **Denotational semantics** via network generation (Section 8): the classical approach that exhaustively enumerates all reachable species and reactions.

(ii) **Operational semantics** for network-free simulation (Section 9): an agent-based approach that applies rules on-the-fly without explicit enumeration.

(iii) **Hybrid semantics** with dynamic regime classification (Section 10): a novel formalization that partitions species between population-level and particle-level representations.

Our main technical result (Theorem 18.4) establishes that all three approaches induce identical continuous-time Markov chains under well-formedness conditions, providing rigorous justification for choosing simulation strategies based on computational considerations alone.

Full Hybrid Semantics. We provide the first complete formal treatment of hybrid particle/population simulation [8], including precise definitions for regime classification (Definition 10.2), promotion and demotion operations with correctness conditions (Definitions 10.5 and 10.6), on-the-fly network expansion (Definition 10.10), and hybrid propensity calculation (Definition 10.11).

1.2 Related Work

The Kappa language [5, 6] provides a closely related formalism with elegant categorical semantics based on graph rewriting. Our treatment differs in handling of symmetry factors (automatic computation in BNGL vs. explicit specification in Kappa), compartment support, and our introduction of atoms and AR graphs. The categorical foundations we employ draw on typed attributed graphs [12] and the single-pushout (SPO) approach to graph transformation [13].

Sekar et al. [7] introduced atoms and atom-rule graphs for visualization purposes. We extend their work with rigorous mathematical definitions, complexity analysis, and formal results connecting AR graph structure to stochastic dynamics.

Hogg et al. [8] introduced hybrid particle/population simulation. We formalize their operational descriptions and prove equivalence to other execution models.

1.3 Paper Organization

Section 2 presents abstract syntax with well-formedness judgments. Section 3 develops graph-theoretic foundations. Section 5 introduces atoms and atom-rule graphs. Section 6 formalizes pattern matching and automorphisms. Section 7 specifies graph operations. Section 8 presents network generation (denotational) semantics. Section 9 develops network-free (operational) semantics. Section 10 formalizes hybrid simulation. Section 11 treats rate laws and observables. Section 12 derives stochastic and deterministic dynamics. Section 18 proves semantic equivalence theorems. Section 19 provides worked examples. Section 20 concludes.

2 Abstract Syntax and Well-Formedness

We define the abstract syntax of BNGL as an inductively-defined grammar with associated well-formedness judgments. The formalization abstracts away concrete syntactic details (whitespace, comments, syntactic sugar) while precisely capturing the semantic content.

2.1 Syntactic Domains

Definition 2.1 (Basic Syntactic Domains)**.** The basic syntactic domains are:

$$\begin{aligned} \mathit{Name} &= \{s \in \Sigma^* \mid s \text{ matches } [A\text{-}Za\text{-}z\_][A\text{-}Za\text{-}z0\text{-}9\_]^*\} \\ \mathit{Label} &= \mathbb{N}^+ \cup \mathit{Name} \\ \mathit{StateVal} &\subseteq \mathit{Name} \\ \mathit{Num} &= \mathbb{R} \\ \mathit{Compartment} &\subseteq \mathit{Name} \end{aligned}$$

We use metavariables n, m for names, ℓ for labels, s for state values, and κ for compartments.

2.2 Molecule Types

Molecule types define the structural signature of molecules.

Definition 2.2 (Component Type)**.** A *component type* is a pair $\gamma = (c, S)$ where $c \in \mathit{Name}$ is the component name and $S \subseteq \mathit{StateVal}$ is the (possibly empty) finite set of allowed internal states. When $S = \emptyset$, the component carries no internal state.

Definition 2.3 (Molecule Type)**.** A *molecule type* is a pair $\Gamma = (M, [\gamma_1, \ldots, \gamma_k])$ where $M \in \mathit{Name}$ is the molecule type name and $[\gamma_1, \ldots, \gamma_k]$ is an ordered list of component types. The ordering is semantically significant: components with identical names are distinguished by position.

Definition 2.4 (Molecule Type Environment)**.** A *molecule type environment* $\mathcal{M}$ is a finite partial function $\mathcal{M} : \mathit{Name} \rightharpoonup \mathit{MolType}$. We write $M \in \mathcal{M}$ for $M \in \mathrm{dom}(\mathcal{M})$, $\mathcal{M}(M)$ for the associated type, and $\mathit{comps}(\mathcal{M}, M)$ for the component list of $\mathcal{M}(M)$.

2.3 Component and Molecule Instances

Definition 2.5 (Bond Specification)**.** A *bond specification* $\beta \in \mathit{BondSpec}$ is one of:

- $\ell \in \mathit{Label}$: bonded to the component sharing label ℓ
- **free**: the site is unbound
- **bound**: bound to some unspecified partner (wildcard)
- **any**: may or may not be bound (wildcard)

We define the partial order $\sqsubseteq$ on bond specifications: **any** $\sqsubseteq \beta$ for all β; **bound** $\sqsubseteq \ell$ for all $\ell \in \mathit{Label}$; **free** $\sqsubseteq$ **free**; and $\ell \sqsubseteq \ell$ for each ℓ.

Definition 2.6 (State Specification)**.** A *state specification* $\sigma \in \mathit{StateSpec}$ is one of:

- $s \in \mathit{StateVal}$: a concrete state value
- $\bot$: unspecified (matches any allowed state)
- ?: wildcard (explicitly matches any state)

We define: $\bot \sqsubseteq s$ and $? \sqsubseteq s$ for all $s \in \mathit{StateVal}$, and $s \sqsubseteq s$ for each s.

Definition 2.7 (Component Instance)**.** A *component instance* is a triple $c = (\mathit{name}, \sigma, \beta)$ where $\mathit{name} \in \mathit{Name}$, $\sigma \in \mathit{StateSpec}$, and $\beta \in \mathit{BondSpec}$.

Definition 2.8 (Molecule Instance)**.** A *molecule instance* is a triple $m = (M, [c_1, \ldots, c_k], \kappa)$ where $M \in \mathit{Name}$ is the molecule type name, $[c_1, \ldots, c_k]$ is an ordered list of component instances, and $\kappa \in \mathit{Compartment} \cup \{\bot\}$ is an optional compartment.

2.4 Species Graphs and Patterns

Definition 2.9 (Species Graph)**.** A *species graph* is a pair $G = (M, B)$ where:

- $M = [m_1, \ldots, m_n]$ is an ordered list of molecule instances
- $B \subseteq (\mathbb{N} \times \mathbb{N}) \times (\mathbb{N} \times \mathbb{N})$ is a symmetric relation encoding bonds, where $((i_1, j_1), (i_2, j_2)) \in B$ indicates a bond between component j_1 of molecule i_1 and component j_2 of molecule i_2

Definition 2.10 (Pattern)**.** A *pattern* is a species graph that may contain:

1. Unspecified states ($\sigma = \bot$) or wildcard states ($\sigma = ?$)
2. Bond wildcards ($\beta \in \{\textbf{bound}, \textbf{any}\}$)
3. Disconnected components

Definition 2.11 (Concrete Species)**.** A *concrete species* is a species graph where:

1. All states are fully specified ($\sigma \in \mathit{StateVal}$ or component has no states)
2. All bonds are concrete ($\beta \in \mathit{Label} \cup \{\textbf{free}\}$)
3. The graph is connected (viewing molecules as nodes and bonds as edges)

We write $\mathcal{S}$ for the set of all concrete species.

2.5 Well-Formedness Judgments

We define well-formedness via typing judgments of the form $\mathcal{M} \vdash G : \mathsf{ok}$.

Definition 2.12 (Well-Formed Component Instance)**.** Component instance $c = (\mathit{name}, \sigma, \beta)$ is well-formed with respect to component type $\gamma = (n, S)$, written $\gamma \vdash c : \mathsf{ok}$, iff:

1. $\mathit{name} = n$
2. If $S = \emptyset$ then $\sigma = \bot$
3. If $S \neq \emptyset$ and $\sigma \notin \{\bot, ?\}$ then $\sigma \in S$

Definition 2.13 (Well-Formed Molecule Instance)**.** Molecule instance $m = (M, [c_1, \ldots, c_k], \kappa)$ is well-formed with respect to environment $\mathcal{M}$, written $\mathcal{M} \vdash m : \mathsf{ok}$, iff:

1. $M \in \mathcal{M}$ with $\mathcal{M}(M) = (M, [\gamma_1, \ldots, \gamma_k])$
2. $\gamma_i \vdash c_i : \mathsf{ok}$ for all $i \in \{1, \ldots, k\}$

Definition 2.14 (Well-Formed Species Graph)**.** Species graph $G = (M, B)$ with $M = [m_1, \ldots, m_n]$ is well-formed with respect to $\mathcal{M}$, written $\mathcal{M} \vdash G : \mathsf{ok}$, iff:

1. $\mathcal{M} \vdash m_i : \mathsf{ok}$ for all i
2. **(Bond consistency)** For each labeled bond: if component (i_1, j_1) has $\beta = \ell$ and component (i_2, j_2) has $\beta = \ell$, then $((i_1, j_1), (i_2, j_2)) \in B$
3. **(Label uniqueness)** Each concrete label ℓ appears exactly twice
4. **(Bond multiplicity)** Each component participates in at most one bond

2.6 Reaction Rules

Definition 2.15 (Reaction Rule)**.** A *reaction rule* is a tuple $r = (L, R, k, \mathit{opts})$ where:

- $L = [P_1, \ldots, P_n]$: list of reactant patterns
- $R = [Q_1, \ldots, Q_m]$: list of product patterns
- k: rate law specification
- $\mathit{opts} \subseteq \{\texttt{DeleteMolecules}, \texttt{MoveConnected}, \texttt{TotalRate}, \texttt{MatchOnce}\}$: rule options

Definition 2.16 (Well-Formed Rule)**.** Rule $r = (L, R, k, opts)$ is well-formed with respect to $\mathcal{M}$ iff:

1. All patterns in L and R are well-formed: $\mathcal{M} \vdash P_i : \mathsf{ok}$ and $\mathcal{M} \vdash Q_j : \mathsf{ok}$
2. **(Label correspondence)** Labels in $L \cup R$ establish a consistent mapping between reactant and product elements
3. **(State definiteness)** State changes are fully specified: if $\sigma_L = \perp$ for some component, then $\sigma_R = \perp$ or the component is deleted

2.7 Observables and Complete Models

Definition 2.17 (Observable)**.** An *observable* is a triple $O = (name, type, [P_1, \ldots, P_k])$ where $type \in \{\texttt{Molecules}, \texttt{Species}\}$ determines counting semantics.

Definition 2.18 (BioNetGen Model)**.** A *BioNetGen model* is a tuple $\mathcal{B} = (\mathcal{M}, \mathcal{P}, \Sigma_0, \mathcal{R}, \mathcal{O}, \mathcal{F})$ where:

- $\mathcal{M}$: molecule type environment
- $\mathcal{P} : Name \rightharpoonup \mathbb{R}$: parameter environment
- Σ_0: initial species with concentrations
- $\mathcal{R}$: list of reaction rules
- $\mathcal{O}$: list of observables
- $\mathcal{F}$: function definitions

Model $\mathcal{B}$ is well-formed iff all components are well-formed with respect to $\mathcal{M}$.

3 Graph-Theoretic Foundations

We formalize the graph-theoretic representation of molecular structures using typed attributed graphs [12].

3.1 Typed Attributed Graphs

Definition 3.1 (Type Graph)**.** The *BNGL type graph* is $T = (T_V, T_E, s_T, t_T)$ where:

$$T_V = \{\tau_{mol}, \tau_{comp}\}$$
$$T_E = \{\epsilon_{has}, \epsilon_{bond}\}$$
$$s_T(\epsilon_{has}) = \tau_{mol}, \quad t_T(\epsilon_{has}) = \tau_{comp}$$
$$s_T(\epsilon_{bond}) = \tau_{comp}, \quad t_T(\epsilon_{bond}) = \tau_{comp}$$

Definition 3.2 (Attribute Signature)**.** The *attribute signature* assigns to each node type:

$$Attr(\tau_{mol}) = Name \times (Compartment \cup \{\perp\})$$
$$Attr(\tau_{comp}) = Name \times StateSpec \times BondSpec$$

Definition 3.3 (BNGL Graph)**.** A *BNGL graph* is a tuple $G = (V, E, \tau, \lambda, s, t)$ where:

- (V, E, s, t) is a directed multigraph with $s, t : E \to V$
- $\tau : V \to T_V$ assigns types to nodes
- $\lambda : V \to \bigcup_v Attr(\tau(v))$ assigns attributes
- For all $e \in E$: $(\tau(s(e)), \tau(t(e))) \in \{(\tau_{mol}, \tau_{comp}), (\tau_{comp}, \tau_{comp})\}$

3.2 Semantic Translation

Definition 3.4 (Graph Semantics of Species Graphs)**.** The semantic function $[\![\cdot]\!] : \mathit{SpeciesGraph} \to \mathit{BNGLGraph}$ translates species graph $G_{syn} = ([m_1, \ldots, m_n], B)$ to BNGL graph $[\![G_{syn}]\!] = (V, E, \tau, \lambda, s, t)$:

1. For each molecule $m_i = (M_i, [c_{i,1}, \ldots, c_{i,k_i}], \kappa_i)$:
   - Create node $v_i \in V$ with $\tau(v_i) = \tau_{mol}$, $\lambda(v_i) = (M_i, \kappa_i)$
   - For each $c_{i,j} = (n_{i,j}, \sigma_{i,j}, \beta_{i,j})$: create node $v_{i,j}$ with $\tau(v_{i,j}) = \tau_{comp}$, $\lambda(v_{i,j}) = (n_{i,j}, \sigma_{i,j}, \beta_{i,j})$
   - Create edge e with $s(e) = v_i$, $t(e) = v_{i,j}$
2. For each $((i_1, j_1), (i_2, j_2)) \in B$: create edges e_1, e_2 with $s(e_1) = v_{i_1,j_1}$, $t(e_1) = v_{i_2,j_2}$ and $s(e_2) = v_{i_2,j_2}$, $t(e_2) = v_{i_1,j_1}$

3.3 Graph Morphisms

Definition 3.5 (Graph Morphism)**.** A *morphism* $\phi : G_1 \to G_2$ between BNGL graphs is a pair $\phi = (\phi_V, \phi_E)$ where $\phi_V : V_1 \to V_2$ and $\phi_E : E_1 \to E_2$ such that:

1. **(Type preservation)** $\tau_2(\phi_V(v)) = \tau_1(v)$
2. **(Structure preservation)** $\phi_V(s_1(e)) = s_2(\phi_E(e))$ and $\phi_V(t_1(e)) = t_2(\phi_E(e))$
3. **(Attribute compatibility)** $\lambda_1(v) \sqsubseteq \lambda_2(\phi_V(v))$

Definition 3.6 (Embedding and Isomorphism)**.** A morphism ϕ is an *embedding*, written $\phi : G_1 \hookrightarrow G_2$, if ϕ_V is injective. It is an *isomorphism* if both ϕ_V and ϕ_E are bijective and ϕ^{-1} is also a morphism.

Proposition 3.1 (Isomorphism Decidability)**.** *For BNGL graphs with bounded component count per molecule, graph isomorphism is decidable in polynomial time via canonical labeling [11].*

4 Category-Theoretic Foundations

This section situates BNGL within the framework of algebraic graph transformation theory, connecting rule application to pushout constructions in adhesive categories. This categorical perspective provides rigorous foundations for compositional reasoning and clarifies the relationship between BNGL and the Kappa language.

4.1 The Category of Site-Graphs

Definition 4.1 (Category **SGraph**)**.** The category **SGraph** of *site-graphs* has:

- **Objects**: BNGL graphs $G = (V, E, \tau, \lambda, s, t)$
- **Morphisms**: Graph morphisms $\phi : G_1 \to G_2$
- **Composition**: Componentwise: $(\phi \circ \psi)_V = \phi_V \circ \psi_V$ and $(\phi \circ \psi)_E = \phi_E \circ \psi_E$
- **Identity**: $\mathrm{id}_G = (\mathrm{id}_{V_G}, \mathrm{id}_{E_G})$

Proposition 4.1 (Categorical Properties)**.** *The category* **SGraph** *has all finite limits and colimits. In particular:*

1. *Products exist (disjoint union with appropriate typing)*

2. *Pullbacks exist (constructed componentwise, then typed)*

3. *Pushouts exist (constructed via identification and union)*

4. *Monomorphisms are precisely injective morphisms*

Proof. Limits and colimits are constructed componentwise on the underlying sets V and E, inheriting from **Set**. The type function τ and attribute function λ are determined by the universal property. For monomorphisms, the characterization follows from **SGraph** being concrete over **Set**. □

4.2 Adhesive Categories

The theory of adhesive categories [14] identifies the abstract properties that make double-pushout rewriting well-behaved.

Definition 4.2 (Adhesive Category)**.** A category $\mathcal{C}$ is *adhesive* if:

1. $\mathcal{C}$ has pullbacks along monomorphisms

2. $\mathcal{C}$ has pushouts along monomorphisms

3. Pushouts along monomorphisms are *Van Kampen squares*: given a pushout square with $m : A \rightarrowtail B$ monic, for any commutative cube with the pushout as base and pullback squares as back faces, the front faces are pullbacks if and only if the top face is a pushout

The Van Kampen property ensures that pushouts "behave well" with respect to pullbacks—intuitively, gluing two objects together and then restricting is the same as first restricting and then gluing.

Theorem 4.2 (**SGraph** is Adhesive)**.** *The category* **SGraph** *of site-graphs is adhesive.*

Proof. The category **Graph** of directed graphs is adhesive, being a presheaf topos (functors from the schema $\bullet \rightrightarrows \bullet$ to **Set**). The category **SGraph** is obtained as a slice category $\mathbf{Graph}/T$ over the type graph, and slicing preserves adhesivity. Adding attributes via the attribute signature corresponds to taking a pullback, which composes with the Van Kampen property. □

Corollary 4.3 (Pushout Complement Uniqueness)**.** *In* **SGraph***, pushout complements (when they exist) are unique up to isomorphism.*

4.3 Double-Pushout Rewriting

Double-pushout (DPO) rewriting provides the classical categorical approach to graph transformation [12].

Definition 4.3 (DPO Production/Rule)**.** A *DPO production* (or *rule*) is a span in **SGraph**:

$$L \xhookleftarrow{l} K \xhookrightarrow{r} R$$

where L is the *left-hand side* (pattern to match), R is the *right-hand side* (replacement), and K is the *interface* (gluing graph) specifying preserved structure. The morphisms l and r are typically monomorphisms.

Definition 4.4 (DPO Direct Derivation)**.** A *DPO direct derivation* $G \Rightarrow_p H$ from graph G to graph H via rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ and match $m : L \to G$ consists of two pushout squares:

$$\begin{array}{ccccc} L & \xleftarrow{l} & K & \xrightarrow{r} & R \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle k} & & \downarrow{\scriptstyle m^*} \\ G & \xleftarrow{d} & D & \xrightarrow{d'} & H \end{array}$$

The left square is the *pushout complement*, constructing the context graph D (intuitively, G with the matched pattern removed except for the interface). The right square is the *pushout*, constructing result H (intuitively, D with the right-hand side glued in).

Definition 4.5 (Gluing Conditions)**.** The DPO derivation exists (i.e., the pushout complement exists) if and only if the match $m : L \to G$ satisfies:

1. **Dangling condition**: No edge in $G \setminus m(L)$ is incident to a node in $m(L \setminus l(K))$
2. **Identification condition**: If $m(x) = m(y)$ for $x, y \in L$, then either both $x, y \in l(K)$ or neither is

Proposition 4.4 (BNGL Rules as DPO Productions)**.** *Every well-formed BNGL rule $r = (L, R, k, opts)$ corresponds to a DPO production where the interface K consists of elements preserved by the correspondence map ϕ_r, and the five primitive operations decompose into:*

- *DeleteBond, DeleteMol: elements in $L \setminus l(K)$*
- *AddBond, AddMol: elements in $R \setminus r(K)$*
- *ChangeState: attribute differences between l and r*

4.4 Sesqui-Pushout Rewriting

Sesqui-pushout (SqPO) rewriting provides an alternative semantics that differs from DPO when gluing conditions fail.

Definition 4.6 (Final Pullback Complement)**.** Given morphisms $l : K \to L$ and $m : L \to G$, a *final pullback complement* (FPC) is a commutative square that is final among all such squares: for any other square with the same L, K, G, there is a unique mediating morphism.

Definition 4.7 (SqPO Direct Derivation)**.** A *SqPO direct derivation* uses an FPC for the left square:

$$\begin{array}{ccccc} L & \xleftarrow{l} & K & \xrightarrow{r} & R \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle k} & & \downarrow{\scriptstyle m^*} \\ G & \xleftarrow{d} & D & \xrightarrow{d'} & H \end{array}$$

where the left square is an FPC and the right square is a pushout.

The key property of FPCs is that they always exist in adhesive categories, making SqPO rewriting always applicable (unlike DPO, which requires the gluing conditions to be satisfied).

Theorem 4.5 (DPO vs. SqPO Comparison)**.** *For a match $m : L \to G$:*

1. *If m satisfies both gluing conditions, DPO and SqPO derivations coincide (isomorphic results)*
2. *If m violates the identification condition, SqPO "clones" identified elements; DPO is undefined*

3. *If m violates the dangling condition, SqPO implicitly deletes dangling edges; DPO is undefined*

Remark 4.1 (BNGL vs. Kappa Semantics). BNGL's semantics is closer to DPO rewriting: pattern matching is typically injective, and dangling bonds are explicitly handled via the `DeleteMolecules` option. Kappa uses SqPO semantics, allowing non-injective matches with implicit side effects. This semantic difference is important when translating between the languages.

4.5 Parallel Independence and the Church-Rosser Property

Definition 4.8 (Parallel Independence). Two derivations $G \Rightarrow_{p_1,m_1} H_1$ and $G \Rightarrow_{p_2,m_2} H_2$ are *parallel independent* if each rule's match doesn't use elements that the other deletes:

$$m_1(L_1) \cap m_2(L_2 \setminus l_2(K_2)) = \emptyset \quad \text{and symmetrically}$$

Theorem 4.6 (Church-Rosser Property). *In adhesive categories, parallel independent derivations satisfy Church-Rosser: there exists G' with $H_1 \Rightarrow_{p_2} G'$ and $H_2 \Rightarrow_{p_1} G'$.*

Corollary 4.7 (Order Independence). *If two BNGL rule instances have disjoint reaction centers, application order does not affect the final state.*

4.6 Sequential Composition of Rules

Definition 4.9 (Sequential Composition). Given rules $p_1 = (L_1 \leftarrow K_1 \rightarrow R_1)$ and $p_2 = (L_2 \leftarrow K_2 \rightarrow R_2)$ with a partial overlap $E : R_1 \supseteq E \subseteq L_2$, their *sequential composition* $p_1;p_2$ is computed via pushout:

$$L_1 \longleftarrow K_1 \longrightarrow R_1 \supseteq E \subseteq L_2 \longleftarrow K_2 \longrightarrow R_2$$

$$L_1 \longleftarrow K_{12} \longrightarrow R_2$$

The composed rule captures the effect of applying p_1 followed by p_2.

This composition operation is fundamental to the rule algebra semantics developed in Section 13.

5 Atoms and Atom-Rule Graphs

We introduce *atoms* as the elementary structural features of BNGL and develop the *atom-rule graph* as a semantic representation of rule interactions.

5.1 Atoms: Elementary Structural Features

Atoms represent the minimal semantic units from which patterns are composed. Each atom captures an atomic piece of structural information that rules can require, consume, or produce.

Definition 5.1 (Atom Types). Given molecule type environment $\mathcal{M}$, the set of *atom types* $\mathcal{A}_\mathcal{M}$ consists of:
1. Molecule atoms. For each $M \in \mathcal{M}$:

$$\alpha_{\mathsf{mol}}(M) \triangleq \text{“a molecule of type } M \text{ exists”}$$

2. Internal state atoms. For each $M \in \mathcal{M}$, component c of M with state set $S \neq \emptyset$, and $s \in S$:

$$\alpha_{\mathsf{state}}(M, c, s) \triangleq \text{“component } c \text{ of } M \text{ is in state } s\text{”}$$

3. Free site atoms. For each $M \in \mathcal{M}$ and component c of M:

$$\alpha_{\mathsf{free}}(M, c) \triangleq \text{“component } c \text{ of } M \text{ is unbound”}$$

4. Bond atoms. For each ordered pair of (molecule type, component) where a bond is structurally possible:

$$\alpha_{\mathsf{bond}}(M_1, c_1, M_2, c_2) \triangleq \text{“a bond exists between } c_1 \text{ of } M_1 \text{ and } c_2 \text{ of } M_2\text{”}$$

Bond atoms are symmetric: we identify $\alpha_{\mathsf{bond}}(M_1, c_1, M_2, c_2) = \alpha_{\mathsf{bond}}(M_2, c_2, M_1, c_1)$ via canonical ordering (lexicographic on (M, c) pairs).

Definition 5.2 (Atom Instance)**.** An *atom instance* in a concrete species S is a specific occurrence of an atom type. Formally, for species S with molecules $m_1, \ldots, m_n$:

- Each molecule m_i of type M contributes an instance of $\alpha_{\mathsf{mol}}(M)$
- Each component in state s contributes an instance of $\alpha_{\mathsf{state}}(M, c, s)$
- Each unbound component contributes an instance of $\alpha_{\mathsf{free}}(M, c)$
- Each bond contributes an instance of the corresponding α_{bond}

We write $\#(\alpha, S)$ for the number of instances of atom type α in S.

Definition 5.3 (Atom Extraction from Patterns)**.** For pattern P, the *atom type set* $atoms(P) \subseteq \mathcal{A}_{\mathcal{M}}$ contains:

$$\begin{aligned} atoms(P) = &\{\alpha_{\mathsf{mol}}(M) : M \text{ appears in } P\} \\ &\cup \{\alpha_{\mathsf{state}}(M, c, s) : \text{component } c \text{ of } M \text{ has } \sigma = s \text{ in } P\} \\ &\cup \{\alpha_{\mathsf{free}}(M, c) : \text{component } c \text{ of } M \text{ has } \beta = \textbf{free} \text{ in } P\} \\ &\cup \{\alpha_{\mathsf{bond}}(M_1, c_1, M_2, c_2) : \text{bond between them in } P\} \end{aligned}$$

Note: wildcards and unspecified elements do *not* contribute atoms.

Proposition 5.1 (Unique Atom Decomposition)**.** *For any concrete species S, there is a unique multiset of atom instances, and S is uniquely determined (up to isomorphism) by this multiset together with the connectivity structure among atoms.*

Proof. The atom instances encode: (1) the type and multiplicity of each molecule via α_{mol} atoms, (2) the state of each component via α_{state} atoms, (3) which components are unbound via α_{free} atoms, and (4) the bond structure via α_{bond} atoms. This information, together with the component-level connectivity implied by bonds, determines S up to molecule ordering, which is resolved by canonical labeling. □

Example 5.1. Consider species `Kin(Y~P,b!1).Sub(d!1,S~U)` representing a phosphorylated kinase bound to an unphosphorylated substrate.

The atom instances are:

- $\alpha_{\mathsf{mol}}(\mathsf{Kin})$: one instance
- $\alpha_{\mathsf{mol}}(\mathsf{Sub})$: one instance
- $\alpha_{\mathsf{state}}(\mathsf{Kin}, \mathsf{Y}, \mathsf{P})$: one instance
- $\alpha_{\mathsf{state}}(\mathsf{Sub}, \mathsf{S}, \mathsf{U})$: one instance
- $\alpha_{\mathsf{bond}}(\mathsf{Kin}, \mathsf{b}, \mathsf{Sub}, \mathsf{d})$: one instance

Note: no α_{free} atoms because both binding sites are occupied.

5.2 Atom-Rule Relationships

Rules interact with atoms in three ways: consuming them (reactant), producing them (product), or requiring them without modification (context).

Definition 5.4 (Correspondence Map)**.** For rule $r = (L, R, k, opts)$, the *correspondence map* $\phi_r : Elem(L) \rightharpoonup Elem(R)$ is a partial bijection between syntactic elements (molecules, components, bonds) of L and R, established by:

1. Explicit labels (identical labels correspond)
2. Positional correspondence (same molecule and component indices)
3. Structural constraint (bonds must connect corresponding components)

Definition 5.5 (Reaction Center and Context)**.** The *reaction center* $\text{Center}(r)$ consists of elements modified by the rule:

$$\text{Center}(r) = \{x \in L : x \notin \text{dom}(\phi_r)\} \cup \{y \in R : y \notin \text{img}(\phi_r)\} \cup \{x \in L : \phi_r(x) \neq x\}$$

The *reaction context* $\text{Context}(r)$ consists of preserved elements:

$$\text{Context}(r) = \{x \in L : x \in \text{dom}(\phi_r) \wedge \phi_r(x) = x\}$$

Definition 5.6 (Atom-Rule Relationship)**.** For rule r and atom type α, the relationship $rel(r, \alpha)$ is:

$$rel(r, \alpha) = \begin{cases} \mathsf{reactant} & \text{if } \alpha \in atoms(L|_{\text{Center}}) \setminus atoms(R) \\ \mathsf{product} & \text{if } \alpha \in atoms(R|_{\text{Center}}) \setminus atoms(L) \\ \mathsf{context} & \text{if } \alpha \in atoms(L|_{\text{Context}}) \\ \bot & \text{otherwise} \end{cases}$$

where $L|_{\text{Center}}$ denotes restriction to center elements.

Lemma 5.2 (Atom Relationship Completeness)**.** *For every atom instance that appears in a rule r, exactly one of the following holds: it is a reactant, it is a product, or it is context. An atom type may appear as both reactant and product (in different instances) when the rule transforms between states of the same structural feature.*

5.3 The Atom-Rule Graph

Definition 5.7 (Atom-Rule Graph for a Single Rule)**.** For rule r, the *atom-rule graph* $\text{AR}(r) = (V_r, E_r)$ is a bipartite directed graph:

- $V_r = \mathcal{A}_r \cup \{r\}$ where $\mathcal{A}_r = \{\alpha : rel(r, \alpha) \neq \bot\}$
- $E_r = E_{\mathsf{react}} \cup E_{\mathsf{prod}} \cup E_{\mathsf{ctx}}$ where:

$$E_{\mathsf{react}} = \{(\alpha, r) : rel(r, \alpha) = \mathsf{reactant}\}$$
$$E_{\mathsf{prod}} = \{(r, \alpha) : rel(r, \alpha) = \mathsf{product}\}$$
$$E_{\mathsf{ctx}} = \{(\alpha, r) : rel(r, \alpha) = \mathsf{context}\}$$

Edge direction encodes information flow: atoms flow *into* rules via reactant/context edges and *out of* rules via product edges.

Definition 5.8 (Model Atom-Rule Graph)**.** For model $\mathcal{B}$ with rules $\mathcal{R} = \{r_1, \ldots, r_n\}$, the *model atom-rule graph* is:

$$\mathrm{AR}(\mathcal{B}) = \left(\mathcal{A}_{\mathcal{B}} \cup \mathcal{R}, \bigcup_{r \in \mathcal{R}} E_r \right)$$

where $\mathcal{A}_{\mathcal{B}} = \bigcup_{r \in \mathcal{R}} \mathcal{A}_r$ and atoms with identical types are merged into single nodes.

Theorem 5.3 (AR Graph Construction Complexity)**.** *The model atom-rule graph* $\mathrm{AR}(\mathcal{B})$ *can be constructed in* $O(|\mathcal{R}| \cdot k_{\max})$ *time, where* $k_{\max}$ *is the maximum number of atoms in any rule.*

Proof. For each rule r: (1) extract atoms from L and R in $O(|L| + |R|)$ time, (2) classify each atom as reactant, product, or context in $O(1)$ time per atom, (3) insert atoms into a hash table for $O(1)$ lookup/merge. Total time per rule is $O(k_r)$ where $k_r = |\mathcal{A}_r|$. Summing over rules: $O(\sum_r k_r) = O(|\mathcal{R}| \cdot k_{\max})$. □

Remark 5.1. This is a significant improvement over rule influence diagrams, which require $O(|\mathcal{R}|^2)$ pairwise comparisons to determine which rules can affect which others [7].

Theorem 5.4 (AR Graph Encodes Causal Dependencies)**.** *Let* $r_1, r_2 \in \mathcal{R}$ *be distinct rules. Rule* r_1 *can directly enable* r_2 *(i.e., firing* r_1 *can make* r_2 *applicable when it was not) if and only if there exists an atom* α *such that* $(r_1, \alpha) \in E_{\mathsf{prod}}$ *and* $(\alpha, r_2) \in E_{\mathsf{react}} \cup E_{\mathsf{ctx}}$.

Proof. ($\Rightarrow$) Suppose r_1 enables r_2. Then r_1 must produce some structural feature α that r_2 requires (as reactant or context). By Definition 5.6, $(r_1, \alpha) \in E_{\mathsf{prod}}$ and $(\alpha, r_2) \in E_{\mathsf{react}} \cup E_{\mathsf{ctx}}$.

($\Leftarrow$) If such an α exists, then r_1 produces an instance of α and r_2 requires α. Consider a state lacking any instance of α: r_2 cannot fire. After r_1 fires, an instance of α exists, potentially enabling r_2. □

Corollary 5.5 (Transitive Causal Chains)**.** *A path* $r_1 \to \alpha_1 \to r_2 \to \alpha_2 \to \cdots \to r_k$ *in* $\mathrm{AR}(\mathcal{B})$ *corresponds to a potential causal chain where each rule produces features enabling subsequent rules.*

5.4 Regulatory Motif Detection

The AR graph enables efficient detection of regulatory motifs without examining rule pairs explicitly.

Definition 5.9 (Feedback Loop)**.** A *feedback loop* is a directed cycle in $\mathrm{AR}(\mathcal{B})$. The loop has *positive sign* if it contains an even number of reactant edges (atoms consumed) and *negative sign* if odd.

Definition 5.10 (Feed-Forward Motif)**.** A *feed-forward motif* consists of three paths from atom α_1 to atom α_3:

1. Direct: $\alpha_1 \to r_1 \to \alpha_3$
2. Indirect: $\alpha_1 \to r_2 \to \alpha_2 \to r_3 \to \alpha_3$

The motif is *coherent* if both paths have the same sign and *incoherent* otherwise.

Proposition 5.6 (Motif Detection Complexity)**.** *Feedback loops of length* $\leq k$ *can be detected in* $O(|\mathrm{AR}|^k)$ *time. Feed-forward motifs can be detected in* $O(|\mathrm{AR}|^3)$ *time.*

5.5 AR Graph Compression

For large models, we define a compression pipeline that preserves regulatory structure while reducing complexity.

Definition 5.11 (Background Filtering)**.** The *background* $BG \subseteq \mathcal{A} \cup \mathcal{R}$ consists of:

- Free site atoms $\alpha_{\mathsf{free}}(\cdot, \cdot)$
- Ground state atoms (designated "default" states)
- Reverse rules of designated forward reactions

The filtered graph is $\mathrm{AR}'(\mathcal{B}) = \mathrm{AR}(\mathcal{B}) \setminus BG$.

Definition 5.12 (Edge Signature)**.** The *edge signature* of rule r is:

$$sig(r) = \{(\alpha, t) : \alpha \in \mathcal{A}_r, t = rel(r, \alpha)\}$$

Rules with identical signatures are *signature-equivalent*.

Definition 5.13 (Compressed AR Graph)**.** Given equivalence relations $\sim_{\mathcal{A}}$ on atoms and $\sim_{\mathcal{R}}$ on rules, the *compressed AR graph* AR_c has:

- Nodes: equivalence classes $[\alpha]$ and $[r]$
- Edges: $([\alpha], [r])$ or $([r], [\alpha])$ iff original edge exists between representatives

Proposition 5.7 (Compression Preserves Reachability)**.** *If there is a path from α to β in $\mathrm{AR}(\mathcal{B})$, then there is a path from $[\alpha]$ to $[\beta]$ in $\mathrm{AR}_c(\mathcal{B})$.*

6 Pattern Matching and Automorphisms

Pattern matching determines when a rule can apply to a given molecular state. We formalize matching as graph embedding and develop the theory of automorphisms needed for correct rate computation.

6.1 Pattern Matching via Graph Embedding

Definition 6.1 (Match)**.** Pattern P *matches* species S, written $P \triangleright S$, iff there exists an embedding $\phi : [\![P]\!] \hookrightarrow [\![S]\!]$.

Definition 6.2 (Match Set)**.** The *match set* of pattern P in species S is:

$$Match(P, S) = \{\phi : [\![P]\!] \hookrightarrow [\![S]\!]\}$$

The *match count* is $|Match(P, S)|$.

Theorem 6.1 (Match Decidability)**.** *For fixed pattern P, determining $P \triangleright S$ is decidable in polynomial time in $|S|$. Specifically, there exists an algorithm running in $O(|S|^{|P|} \cdot |P|)$ time.*

Proof. The Ullmann algorithm [10] solves subgraph isomorphism by backtracking search with constraint propagation. The search tree has depth $|P|$ and branching factor at most $|S|$. Each consistency check takes $O(|P|)$ time. With refinement, practical complexity is much better for typical BNGL patterns. □

6.2 Automorphism Groups

Definition 6.3 (Automorphism)**.** An *automorphism* of BNGL graph G is an isomorphism $\alpha : G \to G$. The set of all automorphisms forms a group under composition:

$$\mathrm{Aut}(G) = \{\alpha : G \to G \mid \alpha \text{ is an isomorphism}\}$$

Definition 6.4 (Symmetry Factor)**.** The *symmetry factor* of pattern P is:

$$sym(P) = |\mathrm{Aut}([\![P]\!])|$$

Proposition 6.2 (Symmetry Factor Computation)**.** *For BNGL graphs,* $|\mathrm{Aut}(G)|$ *can be computed in polynomial time (for practical cases) using the nauty algorithm [11] via partition refinement and backtrack search.*

Example 6.1. The pattern `A(b!1).A(b!1)` (homodimer) has $sym = 2$ because the two A molecules can be swapped. The pattern `A(b!1).B(a!1)` (heterodimer) has $sym = 1$ because A and B are distinguishable.

6.3 Orbits and Distinct Matches

Definition 6.5 (Automorphism Action on Matches)**.** For $\alpha \in \mathrm{Aut}([\![P]\!])$ and match $\phi : [\![P]\!] \hookrightarrow [\![S]\!]$, define $\phi \cdot \alpha = \phi \circ \alpha : [\![P]\!] \hookrightarrow [\![S]\!]$. This defines a right action of $\mathrm{Aut}([\![P]\!])$ on $Match(P, S)$.

Definition 6.6 (Match Orbit)**.** The *orbit* of match ϕ is:

$$[\phi] = \{\phi \cdot \alpha : \alpha \in \mathrm{Aut}([\![P]\!])\}$$

Theorem 6.3 (Orbit-Stabilizer)**.** *For any match* ϕ, $|[\phi]| \cdot |\mathrm{Stab}(\phi)| = |\mathrm{Aut}([\![P]\!])|$, *where* $\mathrm{Stab}(\phi) = \{\alpha : \phi \cdot \alpha = \phi\}$.

Definition 6.7 (Distinct Match Count)**.** The number of *distinct matches* (orbits) of P in S is:

$$match^*(P, S) = \frac{|Match(P, S)|}{|\mathrm{Aut}([\![P]\!])|} = \frac{|Match(P, S)|}{sym(P)}$$

when all orbits have size $|\mathrm{Aut}([\![P]\!])|$ (i.e., trivial stabilizers).

Lemma 6.4 (Orbit Size for Concrete Embeddings)**.** *When* P *embeds into* S *such that the image* $\phi([\![P]\!])$ *has trivial automorphism group in* $[\![S]\!]$, *then* $|[\phi]| = |\mathrm{Aut}([\![P]\!])|$.

7 Graph Operations and Rule Application

We formalize the primitive operations that constitute BNGL's transformation language.

7.1 Primitive Graph Operations

Definition 7.1 (Graph Operations)**.** The set of *primitive graph operations* $\mathcal{O}$ consists of exactly five operations. Each operation o defines a partial function $[\![o]\!] : G \rightharpoonup G'$ with explicit preconditions and effects:

1. AddBond(c_1, c_2): Create bond between components c_1, c_2

- *Precondition*: $c_1, c_2 \in V$ are distinct component nodes, and both are currently unbound: $\lambda(c_1).\beta = \lambda(c_2).\beta = \mathbf{free}$
- *Effect*: $E' = E \cup \{(c_1, c_2), (c_2, c_1)\}$; bond specifications updated to fresh shared label ℓ: $\lambda'(c_i).\beta = \ell$

2. **DeleteBond**(c_1, c_2): Remove bond between c_1, c_2
   - *Precondition*: $(c_1, c_2) \in E$ (bond exists)
   - *Effect*: $E' = E \setminus \{(c_1, c_2), (c_2, c_1)\}$; bond specifications become free: $\lambda'(c_i).\beta = \textbf{free}$
3. **ChangeState**(c, s_{old}, s_{new}): Change internal state of component c
   - *Precondition*: $c \in V$ is a component node with $\lambda(c).\sigma = s_{old}$ and $s_{new} \in S_c$ (allowed states for c)
   - *Effect*: $\lambda'(c).\sigma = s_{new}$; all other attributes unchanged
4. **AddMol**(M, init): Add new molecule of type M with initialization
   - *Precondition*: $M \in \mathcal{M}$ (valid molecule type)
   - *Effect*: $V' = V \cup \{v_m\} \cup \{v_{c_1}, \ldots, v_{c_k}\}$ where v_m is a fresh molecule node and v_{c_i} are its components; all components initialized with $\beta = \textbf{free}$ and states from *init*; containment edges added
5. **DeleteMol**(m): Remove molecule m from the graph
   - *Precondition*: $m \in V$ is a molecule node and all its components are unbound: $\forall c \in \mathit{comps}(m) : \lambda(c).\beta = \textbf{free}$
   - *Effect*: $V' = V \setminus \{m\} \setminus \mathit{comps}(m)$; all edges incident to removed nodes are deleted

Remark 7.1 (Operation Partiality). Each operation is a *partial* function: it is undefined when its precondition is not satisfied. Well-formed rules guarantee that operation sequences always have satisfied preconditions when applied to matching reactants.

Proposition 7.1 (Operation Completeness)**.** *Every well-formed BNGL rule can be expressed as a sequence of primitive operations.*

Proof. A rule $r = (L, R, k, \mathit{opts})$ induces changes classified as: (1) bonds in R not in $L \rightsquigarrow$ AddBond, (2) bonds in L not in $R \rightsquigarrow$ DeleteBond, (3) state differences $\rightsquigarrow$ ChangeState, (4) molecules in R not in $L \rightsquigarrow$ AddMol, (5) molecules in L not in $R \rightsquigarrow$ DeleteMol. These five cases are exhaustive. □

Definition 7.2 (Rule as Operation Sequence)**.** For rule $r = (L, R, k, \mathit{opts})$, the *operation sequence* $\mathcal{O}_r = [o_1, \ldots, o_k]$ is computed by:

1. Compute correspondence map $\phi_r : L \rightharpoonup R$
2. For each bond (c_1, c_2) in R with $c_1, c_2 \notin \phi_r(L)$: append AddBond$(c_1, c_2)$
3. For each bond (c_1, c_2) in L not preserved in R: append DeleteBond(c_1, c_2)
4. For each component c with $\sigma_L \neq \sigma_R$: append ChangeState(c, σ_L, σ_R)
5. For each molecule $m \in R \setminus \mathrm{img}(\phi_r)$: append AddMol$(\mathit{type}(m), \mathit{init}(m))$
6. For each molecule $m \in L \setminus \mathrm{dom}(\phi_r)$: append DeleteMol$(m)$

Lemma 7.2 (Operation Sequence Uniqueness)**.** *For a well-formed rule r, the operation sequence $\mathcal{O}_r$ is unique up to reordering of independent operations.*

Lemma 7.3 (Operation Commutativity)**.** *Operations on disjoint graph elements commute:*

- AddBond(c_1, c_2) *commutes with* ChangeState$(c_3, \cdot, \cdot)$ *if* $c_3 \notin \{c_1, c_2\}$
- DeleteMol(m_1) *commutes with* DeleteMol(m_2) *if* $m_1 \neq m_2$
- *Similar for other pairs acting on disjoint elements*

7.2 Rule Application

Definition 7.3 (Rule Instance)**.** For rule $r = ([P_1, \ldots, P_n], R, k, opts)$ and concrete species $[S_1, \ldots, S_m]$, a *rule instance* is a tuple $\iota = (\phi_1, \ldots, \phi_n, \psi)$ where:

- Each $\phi_i : [\![P_i]\!] \hookrightarrow [\![S_{j_i}]\!]$ is a match
- ψ is an assignment of molecules to distinct species
- The matches are *compatible*: shared labels map consistently

Definition 7.4 (Rule Application)**.** Applying rule instance ι to state σ yields state σ':

$$\sigma' = apply(r, \iota, \sigma) = [\![o_k]\!] \circ \cdots \circ [\![o_1]\!](\sigma, \iota)$$

where $\mathcal{O}_r = [o_1, \ldots, o_k]$ and each operation acts on elements identified by the matches in ι.

Definition 7.5 (Product Species)**.** After rule application, the *product species* are the connected components of the resulting molecular graph. A single rule application may:

- Merge species (if AddBond joins previously separate complexes)
- Split species (if DeleteBond disconnects a complex)
- Create species (via AddMol)
- Destroy species (via DeleteMol)

8 Network Generation (Denotational Semantics)

Network generation computes the complete reaction network by exhaustively applying rules to generate all reachable species and reactions. This provides a *denotational semantics* for BNGL models.

8.1 Reaction Networks

Definition 8.1 (Reaction Network)**.** A *reaction network* is a tuple $\mathcal{N} = (\mathcal{S}, \mathcal{X}, \nu, k)$ where:

- $\mathcal{S}$ is a finite set of species (concrete species graphs up to isomorphism)
- $\mathcal{X}$ is a finite set of reactions
- $\nu : \mathcal{X} \to \mathbb{Z}^{|\mathcal{S}|}$ assigns stoichiometry vectors
- $k : \mathcal{X} \to \mathit{RateLaw}$ assigns rate laws

For reaction x with stoichiometry $\nu(x) = (\nu_1, \ldots, \nu_n)$, we write $\nu_i^-(x) = \max(0, -\nu_i)$ (reactant stoichiometry) and $\nu_i^+(x) = \max(0, \nu_i)$ (product stoichiometry).

8.2 Network Generation Algorithm

Definition 8.2 (Network Generation Semantics)**.** The *network generation function* $generate : \mathit{Model} \to \mathit{Network}$ is defined by the following fixed-point computation:

$\mathcal{S}_0 \leftarrow \{\text{species in } \Sigma_0\}$
$\mathcal{X}_0 \leftarrow \emptyset$
$i \leftarrow 0$
repeat
    $\mathcal{S}_{i+1} \leftarrow \mathcal{S}_i$

```
    X_{i+1} ← X_i
    for each rule r ∈ R do
        for each valid rule instance ι over S_i do
            (reactants, products) ← apply(r, ι)
            x ← (reactants, products, k_r)
            if x ∉ X_{i+1} then
                X_{i+1} ← X_{i+1} ∪ {x}
                S_{i+1} ← S_{i+1} ∪ products
            end if
        end for
    end for
    i ← i + 1
until S_i = S_{i−1} and X_i = X_{i−1}
return (S_i, X_i, ν, k)
```

Theorem 8.1 (Network Generation Soundness)**.** *For any reachable state σ (starting from Σ_0 and applying rules), every species in σ is in generate$(\mathcal{B}).\mathcal{S}$.*

Proof. By induction on the number of rule applications. Base case: $\Sigma_0 \subseteq \mathcal{S}$. Inductive step: if all species in σ are in $\mathcal{S}$ and rule r fires producing σ', then by construction of the generation algorithm, all products are added to $\mathcal{S}$. □

8.3 Termination

Definition 8.3 (Bounded Model)**.** Model $\mathcal{B}$ is *bounded* if there exists $N \in \mathbb{N}$ such that every reachable species has at most N molecules.

Theorem 8.2 (Network Generation Termination)**.** *For bounded models, network generation terminates.*

Proof. With bounded molecule count N and finite molecule types $|\mathcal{M}|$, there are finitely many species (up to isomorphism). Each iteration adds at least one new species or reaction, so the algorithm terminates. □

Remark 8.1. BioNetGen provides termination controls: `max_iter` (iteration limit), `max_agg` (maximum aggregate size), and `max_stoich` (per-species limits).

9 Network-Free Operational Semantics

Network-free simulation applies rules on-the-fly without explicit network enumeration. We provide a small-step operational semantics based on the NFsim implementation [9].

9.1 Agent-Based State

Definition 9.1 (Molecule Agent)**.** A *molecule agent* a is a runtime instance consisting of:

- $id(a) \in \mathbb{N}$: unique identifier
- $type(a) \in \mathcal{M}$: molecule type
- $state(a) : Comps(type(a)) \to StateVal$: component states
- $bond(a) : Sites(type(a)) \to \mathcal{G} \cup \{\perp\}$: bond partners

We write $\mathcal{G}$ for the set of all agents in a given state.

Definition 9.2 (Agent-Based State)**.** An *agent-based state* $\sigma = (\mathcal{G}, \sim)$ consists of:

- $\mathcal{G}$: finite set of molecule agents
- $\sim$: equivalence relation on $\mathcal{G}$ with $a \sim b$ iff a and b are in the same connected complex

Each equivalence class $[a]_\sim$ corresponds to a concrete species.

Definition 9.3 (State-Species Correspondence)**.** The *species multiset* of state σ is:

$$\mathit{species}(\sigma) = \{(S, n_S) : S \in \mathcal{S}, n_S = |\{[a]_\sim : [a]_\sim \cong S\}|\}$$

Two agent states are *equivalent*, $\sigma_1 \equiv \sigma_2$, iff $\mathit{species}(\sigma_1) = \mathit{species}(\sigma_2)$.

9.2 Mappings and Reactant Lists

Definition 9.4 (Mapping)**.** A *mapping* μ for pattern P is a function $\mu : V_P \to \mathcal{G}$ such that:

1. Type compatibility: $\mathit{type}(\mu(v)) = \mathit{type}_P(v)$
2. State compatibility: for each component, $\mathit{state}(\mu(v)) \sqsupseteq \sigma_P(v)$
3. Bond compatibility: bonds in P map to bonds in $\mathcal{G}$

Definition 9.5 (Reactant List)**.** For rule r with reactant pattern P_i, the *reactant list* is:

$$\mathcal{L}_{r,i}(\sigma) = \{\mu : P_i \hookrightarrow \mathcal{G} \mid \mu \text{ is a valid mapping in } \sigma\}$$

Definition 9.6 (MappingSet)**.** A *MappingSet* for rule $r = ([P_1, \ldots, P_n], R, k, \mathit{opts})$ is an n-tuple $(\mu_1, \ldots, \mu_n)$ where:

- Each $\mu_i \in \mathcal{L}_{r,i}(\sigma)$
- Mappings are *collision-free*: $\mathrm{img}(\mu_i) \cap \mathrm{img}(\mu_j) = \emptyset$ for $i \neq j$ (when patterns don't share labels)

9.3 Propensity Functions

Definition 9.7 (Rule Propensity)**.** The propensity $a_r(\sigma)$ of rule r in state σ is:
Microscopic rate (default):

$$a_r(\sigma) = k_r \cdot \prod_{i=1}^{n} |\mathcal{L}_{r,i}(\sigma)|$$

Macroscopic rate (`TotalRate` option):

$$a_r(\sigma) = \begin{cases} k_r & \text{if } \forall i : |\mathcal{L}_{r,i}(\sigma)| > 0 \\ 0 & \text{otherwise} \end{cases}$$

Definition 9.8 (Total Propensity)**.** The total propensity is $a_0(\sigma) = \sum_{r \in \mathcal{R}} a_r(\sigma)$.

9.4 Operational Semantics

Definition 9.9 (Small-Step Transition)**.** The transition relation $\sigma \xrightarrow{r,\mu,\tau} \sigma'$ holds when:

1. Rule r is selected with probability $a_r(\sigma)/a_0(\sigma)$
2. MappingSet $\mu = (\mu_1, \ldots, \mu_n)$ is selected uniformly from valid MappingSets
3. Waiting time τ is drawn from $\text{Exp}(a_0(\sigma))$
4. State σ' is obtained by applying operations $\mathcal{O}_r$ via μ

Definition 9.10 (Network-Free Execution)**.** A *network-free execution* from initial state σ_0 is a sequence:

$$\sigma_0 \xrightarrow{r_1,\mu_1,\tau_1} \sigma_1 \xrightarrow{r_2,\mu_2,\tau_2} \sigma_2 \to \cdots$$

The *trajectory* is the sequence $((t_i, \sigma_i))_{i \geq 0}$ where $t_i = \sum_{j<i} \tau_j$.

9.5 Incremental Update

Definition 9.11 (Affected Agents)**.** After applying rule r via MappingSet μ, the *affected agents* are:

$$\mathit{Affected}(r, \mu) = \bigcup_i \text{img}(\mu_i) \cup \mathit{Neighbors}(\bigcup_i \text{img}(\mu_i))$$

where $\mathit{Neighbors}(A) = \{b : \exists a \in A, c.\ \mathit{bond}(a)(c) = b\}$.

Proposition 9.1 (Incremental Propensity Update)**.** *After rule application, only propensities for rules whose patterns can match affected agents need updating:*

$$\Delta a_{r'}(\sigma) \neq 0 \implies \exists i.\ \mathit{type}(\mathit{Affected}) \cap \mathit{types}(P_i') \neq \emptyset$$

10 Hybrid Particle/Population Semantics

Hybrid simulation combines network-based and network-free approaches by partitioning species between population-level (tracked by count) and particle-level (tracked individually) representations [8]. This section provides the first complete formal treatment.

10.1 Hybrid Ensembles

Definition 10.1 (Hybrid Ensemble)**.** A *hybrid ensemble* is a triple $\mathcal{H} = (\mathcal{X}, \mathcal{S}_{\text{pop}}, \rho)$ where:

- $\mathcal{X}$ is a multiset of fully instantiated species graphs (particles)
- $\mathcal{S}_{\text{pop}} : \mathcal{S}_{\text{pop}} \to \mathbb{N}$ maps population species to copy counts
- $\rho : \mathcal{S} \to \{\text{pop}, \text{part}\}$ is the regime classification with $\mathcal{S}_{\text{pop}} = \rho^{-1}(\text{pop})$

The ensemble represents a system state where some species are "lumped" into populations (losing individual identity) while others retain full graph structure.

Definition 10.2 (Species Regime)**.** A *regime assignment* $\rho : \mathcal{S} \to \{\text{pop}, \text{part}\}$ classifies each species as:

- pop (population): tracked by integer copy number $n_S \in \mathbb{N}$
- part (particle): tracked by individual agent instances

Definition 10.3 (Hybrid State)**.** A *hybrid state* $\sigma_H = (\vec{n}, \mathcal{G}, \rho)$ consists of:

- $\vec{n} = (n_S)_{S \in \mathcal{S}_{\text{pop}}}$: copy numbers for population species
- $\mathcal{G} = \bigcup_{S \in \mathcal{S}_{\text{part}}} \mathcal{G}_S$: agents for particle species
- ρ: current regime assignment

where $\mathcal{S}_{\text{pop}} = \rho^{-1}(\text{pop})$ and $\mathcal{S}_{\text{part}} = \rho^{-1}(\text{part})$.

Definition 10.4 (Hybrid State Equivalence)**.** Hybrid state $\sigma_H = (\vec{n}, \mathcal{G}, \rho)$ is *equivalent* to agent state $\sigma = (\mathcal{G}', \sim)$, written $\sigma_H \approx \sigma$, iff:

1. For each $S \in \mathcal{S}_{\text{pop}}$: $n_S = |\{[a]_\sim \cong S : a \in \mathcal{G}'\}|$
2. For each $S \in \mathcal{S}_{\text{part}}$: agents in $\mathcal{G}_S$ biject with equivalence classes $[a]_\sim \cong S$

10.2 Promotion and Demotion

Species can transition between regimes during simulation.

Definition 10.5 (Promotion)**.** *Promotion* converts a species from particle to population representation:

$$\mathit{promote}(S, \sigma_H) = (\vec{n}[S \mapsto |\mathcal{G}_S|], \mathcal{G} \setminus \mathcal{G}_S, \rho[S \mapsto \text{pop}])$$

Definition 10.6 (Demotion)**.** *Demotion* converts a species from population to particle representation:

$$\mathit{demote}(S, \sigma_H) = (\vec{n}|_{\mathcal{S}_{\text{pop}} \setminus \{S\}}, \mathcal{G} \cup \mathit{instantiate}(S, n_S), \rho[S \mapsto \text{part}])$$

where $\mathit{instantiate}(S, n)$ creates n fresh agents of species type S.

Lemma 10.1 (Promotion/Demotion Preserves Equivalence)**.** *For any hybrid state $\sigma_H \sim \sigma$:*

1. $\mathit{promote}(S, \sigma_H) \sim \sigma$
2. $\mathit{demote}(S, \sigma_H) \sim \sigma$

Proof. Promotion replaces agents with their count, preserving the species multiset. Demotion instantiates the correct number of agents. Both maintain the bijection between species multiplicities. □

10.3 Partial Network Expansion (PNE)

Hybrid simulation maintains a partial network that expands on-the-fly. The *Partial Network Expansion* (PNE) algorithm [8] refines the original rule set to handle the partition between populations and particles.

Definition 10.7 (Partial Network)**.** A *partial network* $\mathcal{N}_\partial = (\mathcal{S}_{enum}, \mathcal{X}_{enum})$ satisfies $\mathcal{S}_{enum} \subseteq \mathcal{S}$ and $\mathcal{X}_{enum} \subseteq \mathcal{X}$ where $(\mathcal{S}, \mathcal{X}) = \mathit{generate}(\mathcal{B})$.

Definition 10.8 (Rule Restriction)**.** Let r be a reaction rule and ϕ be a partial embedding of the reactant pattern into a specific population species $S \in \mathcal{S}_{\text{pop}}$. The *restricted rule* $r|_\phi$ describes the action of r conditioned on the reactant at position ϕ being an instance of population species S.

Definition 10.9 (Population-Mapping Rules)**.** The PNE algorithm replaces the original rule set $\mathcal{R}$ with an expanded set $\mathcal{R}_{PNE}$ containing three types of rules:

1. **Population Rules** ($\mathcal{R}_{\text{pop}}$): All reactants are consumed from $\mathcal{S}_{\text{pop}}$. These rules operate purely on population counts and can be simulated using standard SSA on the partial network.

2. **Particle Rules** ($\mathcal{R}_{\text{part}}$): All reactants are selected from $\mathcal{X}$. These rules operate on individual molecular graphs using network-free semantics.

3. **Mixed Rules** ($\mathcal{R}_{mix}$): Reactants are drawn from both $\mathcal{S}_{\text{pop}}$ and $\mathcal{X}$. These require special handling to coordinate population decrement with particle selection.

This partition ensures unambiguous determination of whether a reactant should be decremented from a counter or removed from the particle multiset.

Proposition 10.2 (PNE Partition)**.** *For any rule $r \in \mathcal{R}$ and regime assignment ρ, the PNE algorithm produces a set of restricted rules such that:*

$$\mathcal{R}_{PNE} = \mathcal{R}_{\text{pop}} \uplus \mathcal{R}_{\text{part}} \uplus \mathcal{R}_{mix}$$

where $\uplus$ denotes disjoint union, and for any state σ:

$$\sum_{r' \in \mathcal{R}_{PNE} : r' \text{ derived from } r} a_{r'}(\sigma) = a_r(\sigma)$$

Definition 10.10 (On-the-Fly Expansion)**.** When a reaction produces species $S \notin \mathcal{S}_{enum}$:

1. Add S to $\mathcal{S}_{enum}$
2. For each rule $r \in \mathcal{R}$: compute reactions involving S as reactant
3. Add new reactions to $\mathcal{X}_{enum}$
4. Classify S via ρ (typically part for newly-seen species)

Proposition 10.3 (Expansion Correctness)**.** *On-the-fly expansion produces exactly the reactions that would be in generate($\mathcal{B}$) involving species seen so far.*

10.4 Hybrid Propensity Calculation

Definition 10.11 (Hybrid Propensity)**.** For a hybrid rule $r_j \in \mathcal{R}_{PNE}$ with reactant species partitioned into population reactants $react_{\text{pop}}(r_j)$ and particle reactants $react_{\text{part}}(r_j)$, the propensity is:

$$a_j(\mathcal{H}) = k_j \cdot \underbrace{\left(\prod_{S \in react_{\text{pop}}(r_j)} \binom{\mathcal{S}_{\text{pop}}(S)}{\nu_S} \right)}_{\text{population factor}} \cdot \underbrace{(count_{matches}(react_{\text{part}}(r_j), \mathcal{X}))}_{\text{particle factor}}$$

where ν_S is the stoichiometry of species S as a reactant, and $count_{matches}$ counts the number of valid MappingSets for the particle reactant patterns in the particle multiset $\mathcal{X}$.

Lemma 10.4 (Hybrid Propensity Correctness)**.** *For $\sigma_H \approx \sigma$, the hybrid propensity equals the network-free propensity: $a_j(\sigma_H) = a_j(\sigma)$.*

Proof. For population species, $\binom{n_S}{\nu}$ counts the number of ways to choose ν copies from n_S identical molecules. For particle species, $|\mathcal{L}_{r,j}|$ counts valid mappings. The product gives the total number of ways to select reactants, which equals the propensity. □

10.5 Mixed Reactions

Reactions may involve both population and particle species.

Definition 10.12 (Reaction Classification)**.** Reaction x is:

- *Population-only* if all reactant species are in $\mathcal{S}_{\text{pop}}$
- *Particle-only* if all reactant species are in $\mathcal{S}_{\text{part}}$
- *Mixed* otherwise

Definition 10.13 (Mixed Reaction Execution)**.** For mixed reaction x with population reactants $(S_1, \ldots, S_k)$ and particle reactants via mappings $(\mu_1, \ldots, \mu_\ell)$:

1. Decrement population counts: $n_{S_i} \leftarrow n_{S_i} - \nu_i$ for $i \leq k$
2. Remove particle agents: delete agents in $\bigcup_j \text{img}(\mu_j)$
3. For each product species T:
   - If $T \in \mathcal{S}_{\text{pop}}$: increment n_T
   - If $T \in \mathcal{S}_{\text{part}}$: create agent(s) for T

Definition 10.14 (Population-to-Particle Binding)**.** When a population species $S \in \mathcal{S}_{\text{pop}}$ binds to a particle species $T \in \mathcal{S}_{\text{part}}$ forming complex U:

1. Select one of the n_S copies uniformly at random
2. *Demote* the selected copy: instantiate as agent a_S
3. Apply the binding operation to a_S and the particle a_T
4. The complex U inherits the particle classification

10.6 Regime Switching Criteria

Definition 10.15 (Threshold-Based Switching)**.** Given threshold $\theta \in \mathbb{N}$, define switching rules:

- *Promote* S when $|\mathcal{G}_S| > \theta$ (too many particles)
- *Demote* S when $n_S < \theta$ and S is involved in a particle-requiring reaction

Definition 10.16 (Conservative Switching)**.** A species S *requires particle representation* if:

1. Some rule r has a pattern P that can match S where P contains wildcards or local function references
2. S is part of a complex with a particle species

Otherwise, S can safely use population representation.

Algorithm 1 Hybrid Particle/Population Simulation

Require: Model $\mathcal{B}$, initial state σ_0, end time T, threshold θ
Ensure: Trajectory $\{(t_i, \sigma_i)\}$

$\sigma_H \leftarrow \mathit{initialize}(\sigma_0)$ ▷ Initial regime assignment
$\mathcal{N}_\partial \leftarrow \mathit{initialNetwork}(\sigma_0)$
$t \leftarrow 0$
while $t < T$ **do**
  Compute propensities a_x for $x \in \mathcal{X}_{enum}$
  $a_0 \leftarrow \sum_x a_x$
  if $a_0 = 0$ **then break**
  end if
  $\tau \sim \text{Exp}(a_0)$
  Select reaction x with probability a_x/a_0
  $\sigma_H \leftarrow \mathit{execute}(x, \sigma_H)$
  $\mathcal{N}_\partial \leftarrow \mathit{expand}(\mathcal{N}_\partial, \mathit{newSpecies})$
  $\sigma_H \leftarrow \mathit{regimeSwitch}(\sigma_H, \theta)$
  $t \leftarrow t + \tau$
  Record (t, σ_H)
end while

10.7 Hybrid Simulation Algorithm

Definition 10.17 (Hybrid Transition)**.** A hybrid transition $\sigma_H \xrightarrow{x,\tau} \sigma'_H$ consists of:

1. Compute propensities $a_x(\sigma_H)$ for all reactions x
2. Sample $\tau \sim \text{Exp}(a_0)$ where $a_0 = \sum_x a_x$
3. Select x with probability a_x/a_0
4. Execute reaction:
   - If population-only: update counts directly
   - If particle-only: use network-free execution
   - If mixed: use Definition 10.13
5. Expand partial network if new species appear
6. Check and apply regime switches

11 Rate Laws and Observables

11.1 Rate Law Types

Definition 11.1 (Elementary Rate Law)**.** An *elementary* (mass-action) rate law has propensity:

$$a(x, \sigma) = k \cdot \prod_{i=1}^{n} \binom{n_{S_i}}{\nu_i^-}$$

where k is the rate constant, n_{S_i} is the count of species S_i, and ν_i^- is its stoichiometry as reactant.

Definition 11.2 (Functional Rate Law)**.** A *functional rate law* allows arbitrary expressions:

$$a(x, \sigma) = f(\mathcal{P}, \mathcal{O}(\sigma), local)$$

where f may depend on parameters $\mathcal{P}$, observable values $\mathcal{O}(\sigma)$, and local molecular context.

Definition 11.3 (Local Function)**.** A *local function* $f_{\%x}$ is evaluated in the context of a specific reactant molecule tagged with label $\%x$. The function can access:

- Component states of the tagged molecule
- Properties of the containing complex (size, composition)
- Observable counts restricted to the complex

Definition 11.4 (Distribution of Rates)**.** For rules with local functions, *distribution of rates* (DOR) computes:

$$a_r(\sigma) = \sum_{\mu \in \mathcal{L}_{r,DOR}} k_r \cdot f(\mu)$$

where the sum ranges over all valid mappings μ and $f(\mu)$ evaluates the local function in the context of mapping μ.

11.2 Observable Semantics

Definition 11.5 (Observable Evaluation)**.** For observable $O = (name, type, [P_1, \ldots, P_k])$ and state σ:

$$[\![O]\!](\sigma) = \sum_{i=1}^{k} count(P_i, \sigma, type)$$

Definition 11.6 (Counting Semantics)**.** The count function depends on type:
Molecules type:

$$count(P, \sigma, \texttt{Molecules}) = \sum_{S \in \mathcal{S}} n_S \cdot \frac{|Match(P, S)|}{|\mathrm{Aut}([\![P]\!])|}$$

This counts the total number of *distinct embeddings* across all species.
Species type:

$$count(P, \sigma, \texttt{Species}) = \sum_{S \in \mathcal{S}: P \triangleright S} n_S$$

This counts the number of *species instances* containing at least one match.

Example 11.1. Consider species `A(b!1).A(b!1)` (A-A dimer) with count 10, and pattern `A(b!+)` (A bound to something).

- `Molecules` count: $10 \cdot 2 = 20$ (each dimer contains two bound A's)
- `Species` count: 10 (each dimer is one species instance)

Definition 11.7 (Weight Vector)**.** For observable O over species list $[\mathcal{S}_1, \ldots, \mathcal{S}_n]$, the *weight vector* is:

$$\vec{w}_O = (w_1, \ldots, w_n) \quad \text{where } w_i = \begin{cases} |Match(P, S_i)|/|\mathrm{Aut}([\![P]\!])| & \text{if } type = \texttt{Molecules} \\ \mathbf{1}[P \triangleright S_i] & \text{if } type = \texttt{Species} \end{cases}$$

Then $[\![O]\!](\sigma) = \vec{w}_O \cdot \vec{n}$.

12 Stochastic and Deterministic Dynamics

12.1 Continuous-Time Markov Chain

Definition 12.1 (State Space)**.** For network $\mathcal{N}$ with n species, the state space is $\Omega = \mathbb{N}^n$. State $\vec{x} = (x_1, \ldots, x_n)$ gives copy numbers.

Definition 12.2 (Stoichiometry Matrix)**.** The stoichiometry matrix $\mathbf{N} \in \mathbb{Z}^{n \times m}$ for m reactions has:

$$N_{ij} = \nu_{ij}^{+} - \nu_{ij}^{-}$$

where ν_{ij}^{-} and ν_{ij}^{+} are the reactant and product stoichiometries of species i in reaction j.

Definition 12.3 (Propensity Vector)**.** The propensity vector $\vec{a}(\vec{x}) = (a_1(\vec{x}), \ldots, a_m(\vec{x}))$ where:

$$a_j(\vec{x}) = k_j \cdot \prod_{i=1}^{n} \binom{x_i}{\nu_{ij}^{-}}$$

for mass-action kinetics.

Definition 12.4 (Generator Matrix)**.** The infinitesimal generator $\mathbf{Q}$ of the CTMC has entries:

$$Q_{\vec{x},\vec{x}'} = \begin{cases} \sum_{j:\vec{x}+\mathbf{N}_{\cdot j}=\vec{x}'} a_j(\vec{x}) & \text{if } \vec{x} \neq \vec{x}' \\ -\sum_{\vec{y}\neq\vec{x}} Q_{\vec{x},\vec{y}} & \text{if } \vec{x} = \vec{x}' \end{cases}$$

Definition 12.5 (Chemical Master Equation)**.** The probability distribution $P(\vec{x}, t)$ evolves according to:

$$\frac{dP(\vec{x}, t)}{dt} = \sum_{\vec{x}'} Q_{\vec{x}',\vec{x}} P(\vec{x}', t)$$

12.2 Gillespie Algorithm

Theorem 12.1 (SSA Correctness)**.** *The Stochastic Simulation Algorithm (Gillespie algorithm) generates sample paths from the exact distribution of the CTMC.*

Proof. The algorithm samples: (1) waiting time $\tau \sim \text{Exp}(a_0)$ where $a_0 = \sum_j a_j$, which is the correct inter-arrival time for exponential races; (2) reaction j with probability a_j/a_0, which is the correct conditional probability given an event occurs. The memoryless property of exponential distributions ensures the Markov property is preserved. □

12.3 ODE Limit

Definition 12.6 (Deterministic Rate Equations)**.** In the thermodynamic limit (large copy numbers, with $c_i = x_i/(N_A V)$), concentrations evolve according to:

$$\frac{d\vec{c}}{dt} = \mathbf{N} \cdot \vec{v}(\vec{c})$$

where $v_j(\vec{c}) = k_j \prod_i c_i^{\nu_{ij}^{-}}$ is the reaction rate.

Proposition 12.2 (Kurtz Theorem (informal))**.** *As the system size (volume) increases with fixed initial concentrations, the stochastic process converges in probability to the deterministic ODE trajectory.*

13 Rule Algebra Semantics

Following Behr and Krivine [15], we present rule algebra semantics that provides an algebraic perspective on stochastic rewriting systems.

13.1 The Rule Algebra

Definition 13.1 (Rule Algebra)**.** The *rule algebra* $\mathfrak{R}(\mathcal{C})$ over a category $\mathcal{C}$ with DPO rewriting is the free $\mathbb{R}$-vector space on the set of DPO rules (spans $L \leftarrow K \rightarrow R$), equipped with a composition operation $\diamond : \mathfrak{R} \times \mathfrak{R} \to \mathfrak{R}$ derived from sequential rule composition.

The composition $p_1 \diamond p_2$ encodes all ways that rule p_2 can follow rule p_1, weighted by the number of overlaps between R_1 and L_2.

Definition 13.2 (Stochastic Mechanics)**.** A *stochastic mechanics* is a linear functional $\omega : \mathfrak{R}(\mathcal{C}) \to \mathbb{R}$ that encodes the probability of rule application. For mass-action kinetics:

$$\omega(p) = k_p \cdot \frac{|Match(L_p, G)|}{|\mathrm{Aut}(L_p)|}$$

where G is the current state.

Theorem 13.1 (Universal CTMC Semantics)**.** *The rule algebra equipped with stochastic mechanics yields a continuous-time Markov chain with:*

- *State space:* $\mathcal{C}/\cong$ *(isomorphism classes of objects)*
- *Generator:* $Q_{G,H} = \sum_{p:G \Rightarrow H} \omega(p)$ *for* $G \neq H$

Proposition 13.2 (Equivalence to Standard Semantics)**.** *The CTMC induced by the rule algebra with mass-action stochastic mechanics coincides with the standard BNGL stochastic semantics of Section 12.*

13.2 Observable Algebra

Definition 13.3 (Observable Algebra)**.** The *observable algebra* $\mathfrak{O}$ is the commutative algebra generated by pattern-counting observables O_P for patterns P, with product given by independent counting.

Theorem 13.3 (Moment Closure)**.** *For finite rule sets, the time evolution of expected observables* $\mathbb{E}[O_P(t)]$ *satisfies a system of ODEs that closes at finite order when patterns have bounded size.*

14 Chemical Reaction Network Theory Connections

We establish connections between BNGL rule structure and chemical reaction network theory (CRNT) [16].

14.1 Deficiency Theory

Definition 14.1 (CRN Structure)**.** For a generated reaction network $\mathcal{N}$, define:

- n: number of complexes (distinct left/right-hand sides)
- ℓ: number of linkage classes (connected components of the reaction graph)
- $s = \mathrm{rank}(\mathbf{N})$: dimension of the stoichiometric subspace

Definition 14.2 (Deficiency)**.** The *deficiency* of a chemical reaction network is:

$$\delta = n - \ell - s$$

This non-negative integer measures a form of linear dependency among reaction vectors within linkage classes.

Theorem 14.1 (Deficiency Zero Theorem (Feinberg))**.** *If a reaction network is weakly reversible and has $\delta = 0$, then:*

1. *There exists exactly one positive steady state in each stoichiometric compatibility class*
2. *Each positive steady state is asymptotically stable*
3. *These conclusions are independent of rate constants*

Proposition 14.2 (Reversible Rules and Weak Reversibility)**.** *If every BNGL rule is reversible (bidirectional), then the generated reaction network is weakly reversible.*

Proof. For each reaction $x : C_i \to C_j$ in the network generated by rule r, the reverse rule generates $x' : C_j \to C_i$. Since the reaction graph is symmetric, every linkage class is strongly connected, hence the network is weakly reversible. □

14.2 Conservation Laws

Definition 14.3 (Conservation Law)**.** A *conservation law* is a vector $\vec{w} \in \mathbb{R}^{|\mathcal{S}|}$ satisfying $\vec{w}^T \mathbf{N} = \vec{0}$. The quantity $\vec{w} \cdot \vec{c}(t)$ is conserved along all trajectories.

Proposition 14.3 (Atom Conservation)**.** *The total count of molecule type M (summed over all species containing M) is conserved if and only if no rule creates or destroys molecules of type M (i.e., $\alpha_{\mathsf{mol}}(M)$ appears in neither E_{react} nor E_{prod} of the AR graph).*

14.3 Persistence and Boundedness

Definition 14.4 (Persistence)**.** A reaction network is *persistent* if no species that is initially present can go extinct: $c_i(0) > 0 \Rightarrow c_i(t) > 0$ for all $t > 0$.

Proposition 14.4 (Structural Persistence Conditions)**.** *If the reaction network has a* siphon *(a set of species W such that all reactions producing any species in W also consume some species in W), then the network may fail to be persistent.*

15 Bisimulation and Model Equivalence

We develop bisimulation equivalences for rule-based models, extending the theory of exact lumpability for CTMCs.

15.1 Forward and Backward Bisimulation

Definition 15.1 (Forward Bisimulation)**.** A partition $\mathcal{H} = \{H_1, \ldots, H_m\}$ of species $\mathcal{S}$ is a *forward bisimulation* if for all pairs of species $S, S' \in H_i$ (same block) and all blocks H_j:

$$\sum_{T \in H_j} Q_{S,T} = \sum_{T \in H_j} Q_{S',T}$$

That is, the aggregate transition rate out of any state to a given block depends only on which block the source state belongs to.

Definition 15.2 (Backward Bisimulation)**.** A partition $\mathcal{H}$ is a *backward bisimulation* if for all $S, S' \in H_i$ and all blocks H_j:

$$\sum_{T \in H_j} Q_{T,S} = \sum_{T \in H_j} Q_{T,S'}$$

That is, the aggregate transition rate into any state from a given block depends only on which block the target state belongs to.

Theorem 15.1 (Exact Lumpability)**.** *Forward bisimulation is equivalent to* exact lumpability*: the lumped process $\tilde{X}_t = [X_t]_{\mathcal{H}}$ (mapping states to their blocks) is itself a CTMC with generator:*

$$\tilde{Q}_{H_i,H_j} = \sum_{T \in H_j} Q_{S,T} \quad \textit{for any } S \in H_i$$

15.2 Rule-Level Bisimulation

Definition 15.3 (Rule Bisimulation)**.** Two rule sets $\mathcal{R}_1$ and $\mathcal{R}_2$ over the same molecule types are *rule bisimilar* if there exists a relation $\mathcal{R}$ on patterns such that:

1. For each rule $r_1 \in \mathcal{R}_1$, there exists $r_2 \in \mathcal{R}_2$ with $(L_1, L_2) \in \mathcal{R}$ and $(R_1, R_2) \in \mathcal{R}$
2. The relation is preserved under all matches: if $(P, Q) \in \mathcal{R}$ and $P \triangleright S$, then $Q \triangleright S'$ for some $S' \cong S$
3. Rate laws satisfy $k_{r_1} \cdot \mathit{matches}(r_1, \sigma) = k_{r_2} \cdot \mathit{matches}(r_2, \sigma)$ for all states σ

Proposition 15.2 (Rule Bisimulation Implies CRN Bisimulation)**.** *If $\mathcal{R}_1 \sim \mathcal{R}_2$ at the rule level, then the generated networks $\mathcal{N}_1$ and $\mathcal{N}_2$ are forward bisimilar as CTMCs.*

15.3 Categorical Equivalence

Definition 15.4 (Category of Rule-Based Models)**.** Define the category **RBM** with:

- Objects: BNGL models $(\mathcal{M}, \mathcal{R}, \Sigma_0)$
- Morphisms: pairs $(\phi_{\mathcal{M}}, \phi_{\mathcal{R}})$ of type-preserving maps that commute with rule application

Definition 15.5 (Semantic vs. Observational Equivalence)**.** Models $\mathcal{B}_1$ and $\mathcal{B}_2$ are:

- *Semantically equivalent* if $[\![\mathcal{B}_1]\!] \cong [\![\mathcal{B}_2]\!]$ as CTMCs
- *Observationally equivalent* if $\mathbb{E}[O(X_t^1)] = \mathbb{E}[O(X_t^2)]$ for all observables O

Proposition 15.3 (Equivalence Hierarchy)**.** *For BNGL models: isomorphism $\Rightarrow$ rule bisimulation $\Rightarrow$ semantic equivalence $\Rightarrow$ observational equivalence.*

16 Causal Semantics

We formalize how atom-rule graphs encode causal structure in the sense of Pearl's interventional calculus [17].

16.1 Pearl's Causal Hierarchy

Definition 16.1 (Causal Hierarchy)**.** Pearl's causal hierarchy distinguishes three levels of causal reasoning:

1. **Association** (Level 1): $P(Y|X = x)$ – observational conditioning
2. **Intervention** (Level 2): $P(Y|\mathrm{do}(X = x))$ – effect of external manipulation
3. **Counterfactual** (Level 3): $P(Y_x|X = x', Y = y')$ – what would have happened under different circumstances

16.2 Causal DAG from AR Graph

Definition 16.2 (Induced Causal DAG)**.** The *causal DAG* $\mathcal{G}_{causal}$ induced by atom-rule graph $\mathrm{AR}(\mathcal{B})$ has:

- Nodes: atom types $\mathcal{A}$
- Directed edge $\alpha_1 \to \alpha_2$ if there exists rule r with $\alpha_1 \in atoms(L_r)$ and $\alpha_2 \in atoms(R_r) \setminus atoms(L_r)$

This captures direct causal influence: α_1's presence enables a rule that produces α_2.

16.3 Interventions

Definition 16.3 (Intervention Operator)**.** The *intervention* $\mathrm{do}(\alpha = v)$ on atom α with value $v \in \{0, 1\}$ produces a modified model $\mathcal{B}|_{\mathrm{do}(\alpha=v)}$:

1. Remove all rules r where $\alpha \in E_{\mathsf{prod}}(r)$ if $v = 0$, or $\alpha \in E_{\mathsf{react}}(r)$ if $v = 1$
2. Add constraints ensuring α-atoms have the specified value
3. Compute modified dynamics on the constrained state space

Definition 16.4 (Knockout Semantics)**.** A *gene knockout* of molecule type M corresponds to the intervention $\mathrm{do}(\alpha_{\mathsf{mol}}(M) = 0)$:

1. Remove all rules producing molecules of type M
2. Remove initial molecules of type M from Σ_0
3. The resulting dynamics represent the knockout phenotype

16.4 Causal Completeness

Theorem 16.1 (Causal Completeness)**.** *Rule-based models are causally complete for levels 1–2 of Pearl's hierarchy:*

1. ***Level 1**: Associational queries $P(Y|X)$ are answered by standard simulation/analysis*
2. ***Level 2**: Interventional queries $P(Y|\mathrm{do}(X))$ are answered by modifying the rule set per Definition 16.3 and simulating*

Level 3 (counterfactual) queries require additional structural assumptions about exogenous variables.

Proof. Level 1 follows from the stochastic semantics. For Level 2, the intervention $\mathrm{do}(X = x)$ corresponds to surgery on the model that removes incoming causal arrows to X and fixes its value. The AR graph encodes exactly these causal dependencies, so the modified model correctly represents the post-intervention distribution. □

17 Temporal Logic Specifications

We present a temporal logic for specifying properties of BNGL models, supporting statistical model checking.

17.1 Probabilistic Bounded Linear Temporal Logic

Definition 17.1 (PBLTL Syntax)**.** The syntax of Probabilistic Bounded Linear Temporal Logic (PBLTL) is:

$$\phi ::= \mathit{true} \mid O \bowtie c \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \bigcirc^{\leq t}\phi \mid \phi_1 \mathcal{U}^{\leq t} \phi_2$$
$$\Phi ::= \mathbb{P}_{\bowtie p}[\phi]$$

where O is an observable, $\bowtie \in \{<, \leq, =, \geq, >\}$, $c \in \mathbb{R}$, $t \in \mathbb{R}_{\geq 0}$ is a time bound, and $p \in [0,1]$ is a probability threshold.

Definition 17.2 (PBLTL Semantics)**.** For a trajectory $\omega = (X_t)_{t \geq 0}$ and time τ:

$$\begin{aligned}
(\omega, \tau) &\models O \bowtie c && \Leftrightarrow O(X_\tau) \bowtie c \\
(\omega, \tau) &\models \bigcirc^{\leq t}\phi && \Leftrightarrow \exists \tau' \in (\tau, \tau + t] : (\omega, \tau') \models \phi \\
(\omega, \tau) &\models \phi_1 \mathcal{U}^{\leq t} \phi_2 && \Leftrightarrow \exists \tau' \in [\tau, \tau + t] : (\omega, \tau') \models \phi_2 \\
& && \quad \wedge \forall \tau'' \in [\tau, \tau') : (\omega, \tau'') \models \phi_1
\end{aligned}$$

Definition 17.3 (Derived Operators)**.** Standard derived operators:

$$\begin{aligned}
\Diamond^{\leq t}\phi &\equiv \mathit{true}\mathcal{U}^{\leq t}\phi && \text{(eventually within time } t\text{)} \\
\Box^{\leq t}\phi &\equiv \neg\Diamond^{\leq t}\neg\phi && \text{(always within time } t\text{)}
\end{aligned}$$

17.2 Statistical Model Checking

Definition 17.4 (Statistical Model Checking Problem)**.** Given model $\mathcal{B}$ and specification $\Phi = \mathbb{P}_{\bowtie p}[\phi]$, determine whether $\mathcal{B} \models \Phi$ using simulation-based hypothesis testing.

Definition 17.5 (Sequential Probability Ratio Test)**.** The SPRT with error bounds α, β and indifference region $[p - \delta, p + \delta]$ accepts $H_0 : \theta \leq p - \delta$ or $H_1 : \theta \geq p + \delta$ based on the likelihood ratio:

$$\Lambda_n = \prod_{i=1}^{n} \frac{P(X_i | H_1)}{P(X_i | H_0)} = \frac{(p+\delta)^{n_+}(1-p-\delta)^{n_-}}{(p-\delta)^{n_+}(1-p+\delta)^{n_-}}$$

where $n_+ = |\{i : \omega_i \models \phi\}|$ and $n_- = n - n_+$.

Theorem 17.1 (SPRT Correctness)**.** *The SPRT terminates with probability 1 and satisfies:*

$$P(\mathit{accept}\ H_1 | H_0) \leq \alpha$$
$$P(\mathit{accept}\ H_0 | H_1) \leq \beta$$

17.3 Example Specifications

Example 17.1 (Receptor Activation). "Receptor activation exceeds 100 within 10 minutes with probability > 0.9":

$$\mathbb{P}_{>0.9}[\Diamond^{\leq 600}(\mathit{Active\_Receptor} > 100)]$$

Example 17.2 (Steady State). "System reaches steady state (changes $< 1\%$) by time 1000":

$$\mathbb{P}_{>0.95}[\Diamond^{\leq 1000}\Box^{\leq 100}(|O' - O|/O < 0.01)]$$

Example 17.3 (Stimulus Response). "If ligand is added, activation occurs within 5 minutes":

$$\mathbb{P}_{>0.99}[\Box^{\leq T}(\mathit{Ligand} > 0 \Rightarrow \Diamond^{\leq 300}\mathit{Active} > 50)]$$

18 Semantic Equivalence Theorems

We now prove our main technical results establishing equivalence between the three simulation approaches.

18.1 State Correspondence

Definition 18.1 (State Abstraction)**.** Define abstraction functions:

- $\alpha_{NF} : \mathit{AgentState} \to \mathbb{N}^{|\mathcal{S}|}$: $\alpha_{NF}(\sigma) = (|\{[a] : [a] \cong S\}|)_{S \in \mathcal{S}}$
- $\alpha_H : \mathit{HybridState} \to \mathbb{N}^{|\mathcal{S}|}$: $\alpha_H(\sigma_H) = (n_S + |\mathcal{G}_S|)_{S \in \mathcal{S}}$ where we abuse notation for population vs. particle counts

Lemma 18.1 (Abstraction Commutativity)**.** *For equivalent states $\sigma_H \sim \sigma$:*

$$\alpha_H(\sigma_H) = \alpha_{NF}(\sigma)$$

18.2 Propensity Correspondence

Lemma 18.2 (Propensity Equivalence)**.** *For network reaction x corresponding to rule r:*

1. *Network propensity (in terms of species counts) equals network-free propensity (in terms of mapping counts) after accounting for symmetry*
2. *Hybrid propensity equals network-free propensity*

Formally, if $\sigma_H \sim \sigma$ and $\alpha(\sigma) = \vec{n}$:

$$a_x^{net}(\vec{n}) = a_r^{NF}(\sigma) = a_x^H(\sigma_H)$$

Proof. For network propensity with mass-action kinetics:

$$a_x^{net} = k_x \prod_i \binom{n_{S_i}}{\nu_i}$$

For network-free, the number of MappingSets is:

$$\prod_i |\mathcal{L}_{r,i}| = \prod_i \left(n_{S_i} \cdot \frac{|\mathit{Match}(P_i, S_i)|}{|\mathrm{Aut}([\![P_i]\!])|} \right)$$

When rules are generated from the network with correct symmetry factor handling:

$$k_r = k_x \cdot \prod_i |\mathrm{Aut}([\![P_i]\!])| / |\mathit{Match}(P_i, S_i)|$$

Substituting and simplifying yields equality.
For hybrid propensity, Lemma 10.4 establishes $a_x^H = a_r^{NF}$. □

18.3 Transition Correspondence

Lemma 18.3 (Transition Equivalence)**.** *For corresponding states and reactions:*

1. *Network transition $\vec{n} \to \vec{n}'$ via reaction x corresponds to the set of NF transitions $\sigma \to \sigma'$ via rule instances yielding $\alpha(\sigma') = \vec{n}'$*
2. *The aggregate probability over NF transitions equals the network transition probability*

Proof. Each network reaction x corresponds to one or more rule instances. The instances producing the same species transformation have probabilities that sum to a_x/a_0. After the transition, species counts change by the stoichiometry vector $\mathbf{N}_{\cdot j}$, matching the network dynamics. □

18.4 Main Equivalence Theorem

Theorem 18.4 (Three-Way Semantic Equivalence)**.** *Let $\mathcal{B}$ be a well-formed BioNetGen model where network generation terminates, producing network $\mathcal{N} = \text{generate}(\mathcal{B})$. Consider initial state σ_0 with species counts $\vec{n}_0 = \alpha(\sigma_0)$.*

The following three stochastic processes are equivalent in distribution:

1. ***Network-based SSA*** *on $\mathcal{N}$ starting from $\vec{n}_0$*
2. ***Network-free simulation*** *on $\mathcal{B}$ starting from σ_0*
3. ***Hybrid simulation*** *on $\mathcal{B}$ starting from $\sigma_H^0 \sim \sigma_0$*

More precisely, let X_t^{net}, X_t^{NF}, X_t^H denote the respective processes (with agent states abstracted to species counts). Then:

$$X_t^{net} \overset{d}{=} \alpha(X_t^{NF}) \overset{d}{=} \alpha_H(X_t^H)$$

for all $t \geq 0$.

Proof. We establish that all three processes define the same CTMC by showing they have:

1. The same state space: $\mathbb{N}^{|\mathcal{S}|}$ (after abstraction)
2. The same transition rates: by Lemma 18.2
3. The same transition targets: by Lemma 18.3

State correspondence: Network states are species count vectors. Agent states abstract to count vectors via α_{NF}. Hybrid states abstract via α_H. By Lemma 18.1, equivalent inputs map to equal abstractions.

Propensity correspondence: By Lemma 18.2, for any network reaction x, the propensity computed network-based, network-free, or hybrid are equal when evaluated on corresponding states. For the hybrid case, this relies on Proposition 10.2: the PNE algorithm partitions $\mathcal{R}$ into $\mathcal{R}_{PNE} = \mathcal{R}_{\text{pop}} \uplus \mathcal{R}_{\text{part}} \uplus \mathcal{R}_{mix}$ such that total propensity is preserved.

Transition correspondence: By Lemma 18.3, transitions in the three formulations produce corresponding state changes. The correctness of rule application relies on the unique decomposition of patterns into atoms (Proposition 5.1) and the commutativity of primitive graph operations on disjoint elements (Lemma 7.3).

Since all three define CTMCs with the same generator matrix (up to abstraction), they are distributionally equivalent. □

Corollary 18.5 (Observable Equivalence)**.** *For any observable O and time t:*

$$\mathbb{E}[[\![O]\!](X_t^{net})] = \mathbb{E}[[\![O]\!](X_t^{NF})] = \mathbb{E}[[\![O]\!](X_t^H)]$$

18.5 Complexity Tradeoffs

Proposition 18.6 (Space-Time Tradeoff)**.** *The three simulation approaches have different complexity characteristics:*

Approach	***Space***	***Time per step***
Network-based	$O(\lvert\mathcal{S}\rvert + \lvert\mathcal{X}\rvert)$	$O(\lvert\mathcal{X}\rvert)$
Network-free	$O(\lvert\mathcal{G}\rvert)$	$O(\lvert\mathcal{R}\rvert \cdot k_{\max}^2)$
Hybrid	$O(\lvert\mathcal{S}_{\text{pop}}\rvert + \lvert\mathcal{G}_{\text{part}}\rvert)$	$O(\lvert\mathcal{X}_{\text{pop}}\rvert + \lvert\mathcal{R}_{\text{part}}\rvert \cdot k^2)$

where $k_{\max}$ is the maximum pattern size.

The hybrid approach can achieve the best of both: population-level efficiency for high-copy species and particle-level representation for rare species requiring exact tracking.

19 Worked Examples

19.1 Example 1: Ligand-Receptor Binding

Consider the canonical binding model:

```
begin model
begin molecule types
    L(r)       # Ligand with receptor-binding site
    R(l,Y~U~P) # Receptor with ligand site and phosphorylation site
end molecule types

begin seed species
    L(r) 1000
    R(l,Y~U) 300
end seed species

begin reaction rules
    # Binding/unbinding
    L(r) + R(l) <-> L(r!1).R(l!1) kf, kr
    # Phosphorylation (requires bound ligand)
    L(r!1).R(l!1,Y~U) -> L(r!1).R(l!1,Y~P) kp
end reaction rules
end model
```

Atom Analysis:

$$\begin{aligned}\mathcal{A} = \{&\alpha_{\mathsf{mol}}(\mathsf{L}), \alpha_{\mathsf{mol}}(\mathsf{R}),\\ &\alpha_{\mathsf{free}}(\mathsf{L},\mathsf{r}), \alpha_{\mathsf{free}}(\mathsf{R},\mathsf{l}),\\ &\alpha_{\mathsf{state}}(\mathsf{R},\mathsf{Y},\mathsf{U}), \alpha_{\mathsf{state}}(\mathsf{R},\mathsf{Y},\mathsf{P}),\\ &\alpha_{\mathsf{bond}}(\mathsf{L},\mathsf{r},\mathsf{R},\mathsf{l})\}\end{aligned}$$

AR Graph for Rule 1 (binding):

- Reactant atoms: $\alpha_{\mathsf{free}}(\mathsf{L},\mathsf{r})$, $\alpha_{\mathsf{free}}(\mathsf{R},\mathsf{l})$
- Product atoms: $\alpha_{\mathsf{bond}}(\mathsf{L},\mathsf{r},\mathsf{R},\mathsf{l})$
- Context atoms: $\alpha_{\mathsf{mol}}(\mathsf{L})$, $\alpha_{\mathsf{mol}}(\mathsf{R})$

AR Graph for Rule 2 (phosphorylation):

- Reactant atoms: $\alpha_{\mathsf{state}}(\mathsf{R},\mathsf{Y},\mathsf{U})$
- Product atoms: $\alpha_{\mathsf{state}}(\mathsf{R},\mathsf{Y},\mathsf{P})$
- Context atoms: $\alpha_{\mathsf{bond}}(\mathsf{L},\mathsf{r},\mathsf{R},\mathsf{l})$

Causal Path: Rule 1 produces α_{bond} which enables Rule 2. This is visible as path $r_1 \to \alpha_{\mathsf{bond}} \to r_2$ in the AR graph.

19.2 Example 2: Symmetric Dimerization

```
begin molecule types
    A(d)  # Molecule with dimerization site
end molecule types

begin reaction rules
    A(d) + A(d) <-> A(d!1).A(d!1) kf, kr
end reaction rules
```

Symmetry Analysis:
The product pattern `A(d!1).A(d!1)` has $sym = 2$ (the two A molecules are interchangeable). For the forward reaction with n monomers:

- Number of unordered pairs: $\binom{n}{2} = \frac{n(n-1)}{2}$
- Propensity: $a_f = k_f \cdot \frac{n(n-1)}{2}$

For the reverse reaction with m dimers:

- Each dimer can dissociate: m ways
- But symmetry factor in product is 2, so: $a_r = k_r \cdot m$

Operation Sequence:

1. AddBond(A_1.d, A_2.d)

This single operation transforms two monomers into one dimer.

19.3 Example 3: Hybrid Simulation Scenario

Consider a signaling cascade with:

- Abundant scaffold protein (10^4 copies)
- Rare activated receptor (10 copies)
- Cascade of modifications

Regime Assignment:

- Scaffold (unbound): pop (high copy, no need for individual tracking)
- Receptor complex: part (low copy, stochastic effects important)
- Scaffold-receptor complex: part (inherits from receptor)

Mixed Reaction: When scaffold binds to receptor:

1. Population count of scaffold decremented
2. One scaffold instantiated as agent
3. Bond created to receptor particle
4. Complex tracked at particle level

This demonstrates how hybrid simulation efficiently handles systems spanning multiple scales.

20 Conclusion

We have presented a comprehensive formal semantics for the BioNetGen Language that unifies seven mathematical perspectives:

Category-Theoretic Foundations. We situated BNGL within the framework of adhesive categories, showing that rule application corresponds to DPO pushout constructions. This clarifies the relationship between BNGL (DPO-like semantics with explicit dangling bond handling) and Kappa (SqPO semantics with implicit side effects), and provides the mathematical foundation for the Church-Rosser property ensuring order independence for parallel rule applications.

Atoms and Atom-Rule Graphs. The atom formalism provides a canonical decomposition of patterns into minimal semantic units, enabling efficient analysis of rule interactions. The AR graph encodes complete causal dependency information with $O(|\mathcal{R}| \cdot k)$ construction complexity, compared to $O(|\mathcal{R}|^2)$ for rule influence diagrams.

Primitive Operations. The five graph operations (AddBond, DeleteBond, ChangeState, AddMol, DeleteMol) provide a complete and minimal transformation language for BNGL rules.

Rule Algebra Semantics. Connection to the Behr-Krivine framework provides an algebraic perspective on stochastic rewriting, with composition operations on rules and moment closure for observables.

Chemical Reaction Network Theory. We established connections between rule structure and CRNT properties including deficiency, weak reversibility, and persistence, enabling structural analysis of generated networks.

Bisimulation Equivalences. Forward and backward bisimulation provide exact lumpability, and rule-level bisimulation offers a compositional notion of model equivalence with a categorical characterization.

Causal and Temporal Semantics. The AR graph encodes causal structure supporting interventional queries (knockouts, perturbations), while PBLTL specifications enable statistical model checking of behavioral properties.

Three-Way Semantic Equivalence. We proved that network-based, network-free, and hybrid simulation produce identical stochastic dynamics under well-formedness conditions, providing rigorous justification for simulation strategy selection based purely on computational considerations.

20.1 Applications

This formal semantics enables:

- **Certified compilation**: Formally verified translation between BNGL and simulator implementations
- **Model equivalence checking**: Rigorous comparison of models with different syntactic representations via bisimulation
- **Static analysis**: Type checking, reachability analysis, deficiency computation, and invariant verification at the rule level
- **Optimization**: Formally justified transformations for improved simulation efficiency
- **Causal reasoning**: Support for knockout experiments and interventional queries via AR graph analysis
- **Verification**: Statistical model checking of temporal properties against simulation traces

20.2 Future Work

Several extensions merit further development:

- **Compartmental models**: Formal treatment of volumes, transport, and surface-volume reactions within the categorical framework
- **Energy-based semantics**: Formalization of detailed balance constraints and energy patterns with CRNT connections
- **Model reduction**: Formal criteria connecting rule-level bisimulation to aggregation that preserves essential dynamics
- **Counterfactual reasoning**: Extensions to Level 3 of Pearl's hierarchy with appropriate structural assumptions
- **Verified implementation**: Extraction of certified simulators from the formal semantics using proof assistants

References

[1] M.L. Blinov, J.R. Faeder, B. Goldstein, and W.S. Hlavacek. BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291, 2004.

[2] J.R. Faeder, M.L. Blinov, and W.S. Hlavacek. Rule-based modeling of biochemical systems with BioNetGen. In *Systems Biology*, pages 113–167. Humana Press, 2009.

[3] L.A. Harris, J.S. Hogg, J.J. Tapia, J.A.P. Sekar, S. Gupta, I. Korsunsky, A. Arber, D. Barua, R.P. Sheehan, and J.R. Faeder. BioNetGen 2.2: advances in rule-based modeling. *Bioinformatics*, 32(21):3366–3368, 2016.

[4] W.S. Hlavacek, J.R. Faeder, M.L. Blinov, R.G. Posner, M. Hucka, and W. Fontana. Rules for modeling signal-transduction systems. *Science STKE*, 2006(344):re6, 2006.

[5] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[6] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling of cellular signalling. In *CONCUR 2007*, LNCS 4703, pages 17–41. Springer, 2007.

[7] J.A.P. Sekar, J.J. Tapia, and J.R. Faeder. Automated visualization of rule-based models. *PLoS Computational Biology*, 13(11):e1005857, 2017.

[8] 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 Computational Biology*, 10(4):e1003544, 2014.

[9] M.W. Sneddon, J.R. Faeder, and T. Emonet. Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nature Methods*, 8(2):177–183, 2011.

[10] J.R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.

[11] B.D. McKay and A. Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014.

[12] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Springer, 2006.

[13] M. Löwe. Algebraic approach to single-pushout graph transformation. *Theoretical Computer Science*, 109(1-2):181–224, 1993.

[14] S. Lack and P. Sobociński. Adhesive and quasiadhesive categories. *RAIRO - Theoretical Informatics and Applications*, 39(3):511–545, 2005.

[15] N. Behr and J. Krivine. Stochastic mechanics of graph rewriting. In *LICS 2020*, pages 46–59. ACM, 2020.

[16] M. Feinberg. *Foundations of Chemical Reaction Network Theory*. Springer, 2019.

[17] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009.

[18] E.M. Clarke, T.A. Henzinger, H. Veith, and R. Bloem. *Handbook of Model Checking*. Springer, 2018.

[19] H.L.S. Younes and R.G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation*, 204(9):1368–1409, 2006.

[20] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Symbolic computation of differential equivalences. *Theoretical Computer Science*, 777:132–154, 2019.

A Grammar Summary

```
<model>       ::= 'begin model' <block>* 'end model'
<block>       ::= <param_block> | <moltype_block> | <species_block>
                | <obs_block> | <rule_block> | <func_block>

<param_block> ::= 'begin parameters' <param>* 'end parameters'
<param>       ::= <name> <expr>

<moltype_block> ::= 'begin molecule types' <moltype>* 'end molecule types'
<moltype>     ::= <name> '(' <comptype_list>? ')'
<comptype>    ::= <name> ('~' <state>)*

<species_block> ::= 'begin seed species' <species>* 'end seed species'
<species>     ::= <species_graph> <concentration>

<species_graph> ::= <molecule> ('.' <molecule>)*
<molecule>    ::= <name> '(' <component_list>? ')' <compartment>?
<component>   ::= <name> ('~' <state>)? <bonds>?
<bonds>       ::= ('!' (<label> | '+' | '?'))*

<rule_block>  ::= 'begin reaction rules' <rule>* 'end reaction rules'
<rule>        ::= <reactants> ('->' | '<->') <products> <ratelaw>
<reactants>   ::= <pattern> ('+' <pattern>)*
<products>    ::= <pattern> ('+' <pattern>)* | '0'
<pattern>     ::= <species_graph>

<ratelaw>     ::= <name> | <number> | <functional_rate>
```

```
<obs_block>   ::= 'begin observables' <obs>* 'end observables'
<obs>         ::= ('Molecules' | 'Species') <name> <pattern_list>
```

B Complexity Analysis

Operation	Complexity	Notes
Pattern matching	$O(\|S\|^{\|P\|})$	Ullmann algorithm
Canonical labeling	$O(\text{poly})$ typical	nauty; worst case exp.
Automorphism group	$O(\text{poly})$ typical	Via canonical labeling
AR graph construction	$O(\|\mathcal{R}\| \cdot k_{\max})$	Linear in model size
Network generation	$O(\|\mathcal{S}\|^2 \cdot \|\mathcal{R}\|)$	Per iteration
NF simulation step	$O(\|\mathcal{R}\| \cdot k^2)$	Per reaction
Hybrid step	$O(\|\mathcal{X}_{\text{pop}}\| + \|\mathcal{R}_{\text{part}}\| \cdot k^2)$	Mixed

C Proof Details

C.1 Proof of Lemma 6.4

Proof. By the orbit-stabilizer theorem, $|[\phi]| \cdot |\text{Stab}(\phi)| = |\text{Aut}(\llbracket P \rrbracket)|$. If $\phi(\llbracket P \rrbracket)$ has trivial automorphism group in $\llbracket S \rrbracket$, then any $\alpha \in \text{Aut}(\llbracket P \rrbracket)$ with $\phi \circ \alpha = \phi$ must satisfy $\alpha = \text{id}$. Thus $\text{Stab}(\phi) = \{\text{id}\}$ and $|[\phi]| = |\text{Aut}(\llbracket P \rrbracket)|$. □

C.2 Proof of Lemma 7.3

Proof. Operations on disjoint graph elements modify disjoint sets of nodes, edges, and attributes. Since the operations are pure functions on disjoint state components, their composition is independent of order:

$$\llbracket o_1 \rrbracket \circ \llbracket o_2 \rrbracket(G) = \llbracket o_2 \rrbracket \circ \llbracket o_1 \rrbracket(G)$$

when $\mathit{affected}(o_1) \cap \mathit{affected}(o_2) = \emptyset$. □

C.3 Explicit combinatorial derivation of rate constants

We make the mapping between network (species-based) rate constants k_x and rule-based (microscopic) rate constants k_r fully explicit. Let a rule pattern P have automorphism group $\text{Aut}(P)$ with symmetry factor $\text{sym}(P) = |\text{Aut}(P)|$. Assume the network-free semantics counts *ordered* matches of P into a mixture M. Let $N(P, M)$ be the number of ordered embeddings of P into M. Then the network-free propensity is

$$a_{\text{NF}}(M) = k_r \, N(P, M). \tag{1}$$

For the corresponding network reaction X with mass-action rate k_x, the network propensity is

$$a_{\text{net}}(M) = k_x \prod_i \binom{n_i}{\nu_i}, \tag{2}$$

where n_i is the population of species i and ν_i is its stoichiometric coefficient in X. For each pattern P, the relationship between ordered and unordered matches is

$$N(P, M) = \text{sym}(P) \prod_i \binom{n_i}{\nu_i}. \tag{3}$$

Equating propensities yields the exact conversion

$$k_r = \frac{k_x}{\text{sym}(P)}. \tag{4}$$

Example (homodimerization). For $A + A \to A_2$, we have $\mathrm{sym}(P) = 2$. The network propensity is $a_{\mathrm{net}} = k_x\binom{n_A}{2} = k_x n_A(n_A-1)/2$. The network-free semantics counts ordered matches $N(P,M) = n_A(n_A-1)$, so $a_{\mathrm{NF}} = k_r n_A(n_A-1)$. Thus $k_r = k_x/2$, in agreement with the general formula above.

C.4 Canonical labeling and complexity notes

All results that rely on canonical representatives of molecular graphs assume the existence of a canonical labeling algorithm. In practice, this is implemented using graph isomorphism software (e.g., nauty or bliss). While average-case performance is efficient, worst-case complexity of canonical labeling is not known to be polynomial. This does not affect semantic correctness, but it bounds implementation complexity rather than mathematical validity.

Explicit inverse construction (Prop. 5.1). Given a multiset of atoms encoding component states, free sites, and bonds, one reconstructs a unique (up to isomorphism) species graph by: (i) instantiating one node per molecule atom, (ii) assigning component states and site states, (iii) connecting sites according to bond atoms, and (iv) applying canonical labeling to fix a unique representative. This makes the bijection between abstract states and species explicit.

A Categorical formulation of pattern matching

We formalize pattern matching as a functor between categories of graphs. Let **Patt** be the category whose objects are pattern graphs and whose morphisms are graph isomorphisms. Let **Mix** be the category of mixture graphs with injective homomorphisms. A match of pattern P in mixture M is an injective graph homomorphism $f : P \to M$ preserving labels, states, and bonds.

Define the matching functor

$$\mathcal{M} : \mathbf{Patt}^{op} \times \mathbf{Mix} \to \mathbf{Set}$$

by $\mathcal{M}(P,M) = \{f \mid f : P \to M \text{ is an injective homomorphism}\}$. Automorphisms of P act on $\mathcal{M}(P,M)$ by precomposition, yielding the orbit set

$$\mathcal{M}(P,M)/\mathrm{Aut}(P).$$

Unordered (network) matches correspond to these orbits, while ordered (network-free) matches correspond to elements of $\mathcal{M}(P,M)$ itself. Thus,

$$|\mathcal{M}(P,M)| = |\mathrm{Aut}(P)| \cdot |\mathcal{M}(P,M)/\mathrm{Aut}(P)|,$$

recovering the symmetry factor relation used in Section 18.

B Formal proof sketch of Lemma 18.2

Let P be a rule pattern and X its induced network reaction with stoichiometry ν_i for species i. For a mixture M with populations n_i, the number of unordered embeddings of P is $\prod_i \binom{n_i}{\nu_i}$. Each unordered embedding corresponds to $\mathrm{sym}(P)$ ordered embeddings. Thus $N(P,M) = \mathrm{sym}(P)\prod_i \binom{n_i}{\nu_i}$. Then

$$a_{\mathrm{NF}}(M) = k_r N(P,M) = k_r \mathrm{sym}(P) \prod_i \binom{n_i}{\nu_i}.$$

Equating with $a_{\mathrm{net}}(M) = k_x \prod_i \binom{n_i}{\nu_i}$ gives $k_r = k_x/\mathrm{sym}(P)$. □