MPRI

Reachability Analysis of Rule-based Models

[ICCMSE'07,VMCAI'08]

Jérôme Feret
DI - ÉNS

Wednesday, the 16th of October, 2019

In this talk...

We illustrate the following concepts:

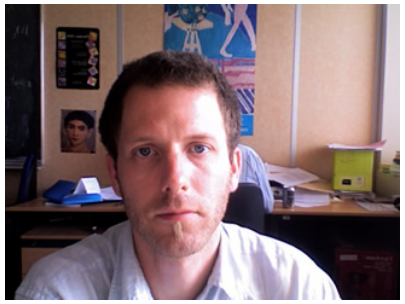
- Galois connections:
  - the upper closure operator $\gamma \circ \alpha$,
  - the lower closure operator $\alpha \circ \gamma$;
- soundness:
  - the abstraction forgets no behavior;
- completeness:
  - sufficient conditions that ensure the absence of false positive;

on an abstraction of the reachable connected components in a site-graph rewriting language.

Joint-work with...

Walter Fontana
Harvard Medical School

Vincent Danos
ÉNS



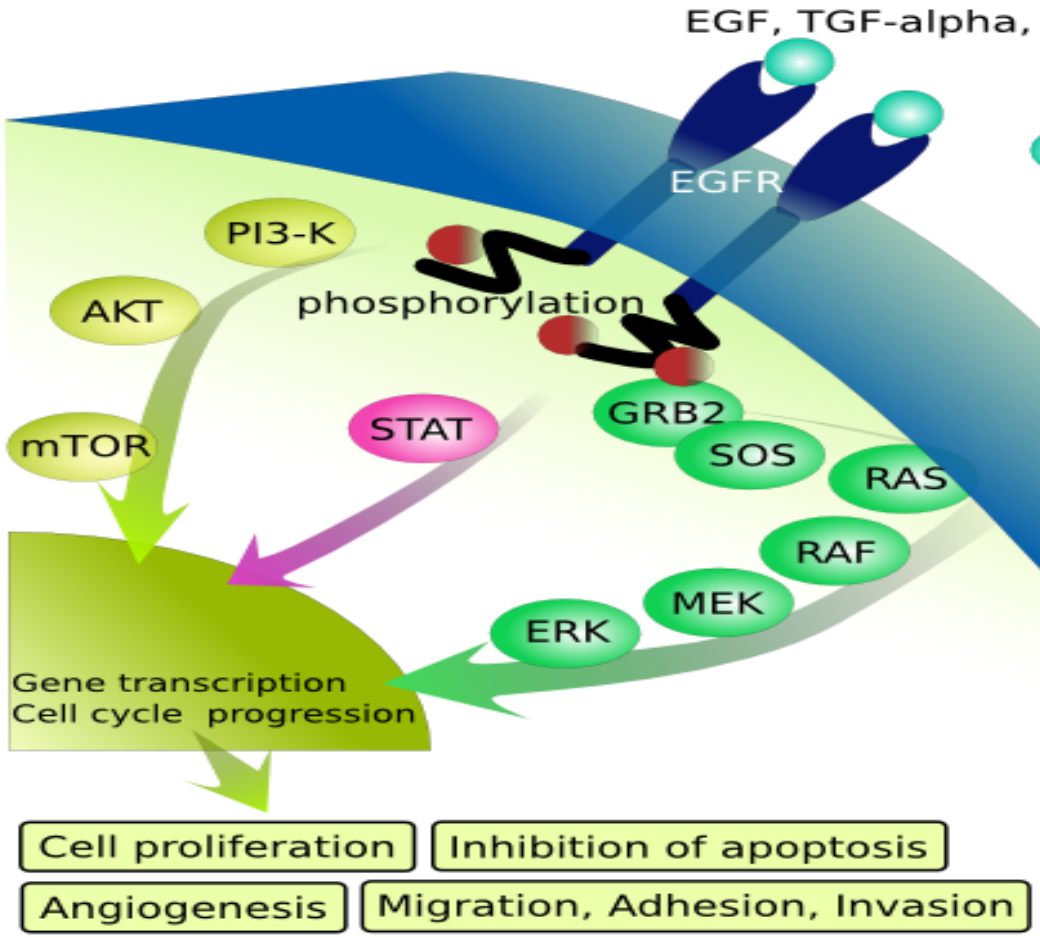
Russ Harmer
ÉNS Lyon

Jean Krivine
Paris VII

Overview

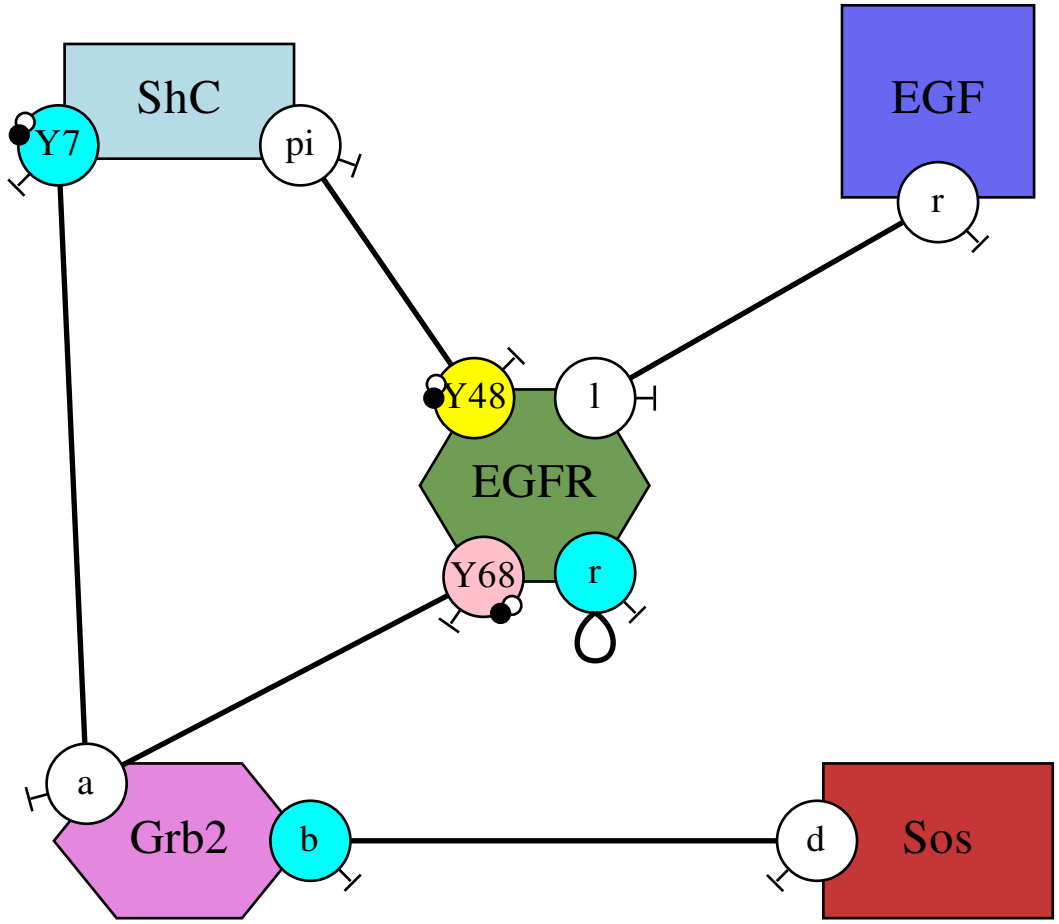
1. Introduction
2. Language: Kappa
3. Abstraction: Local views
4. Completeness: false positives?
5. Local fragment of Kappa
6. Decontextualization
7. Conclusion

Signaling Pathways

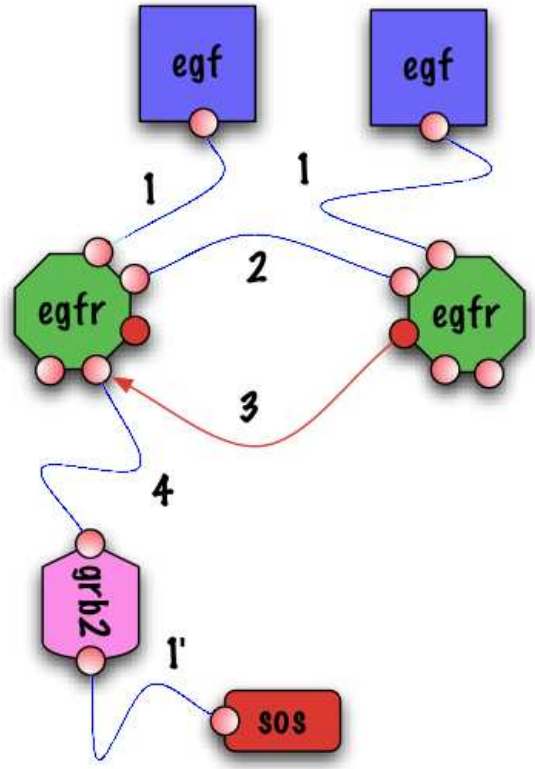


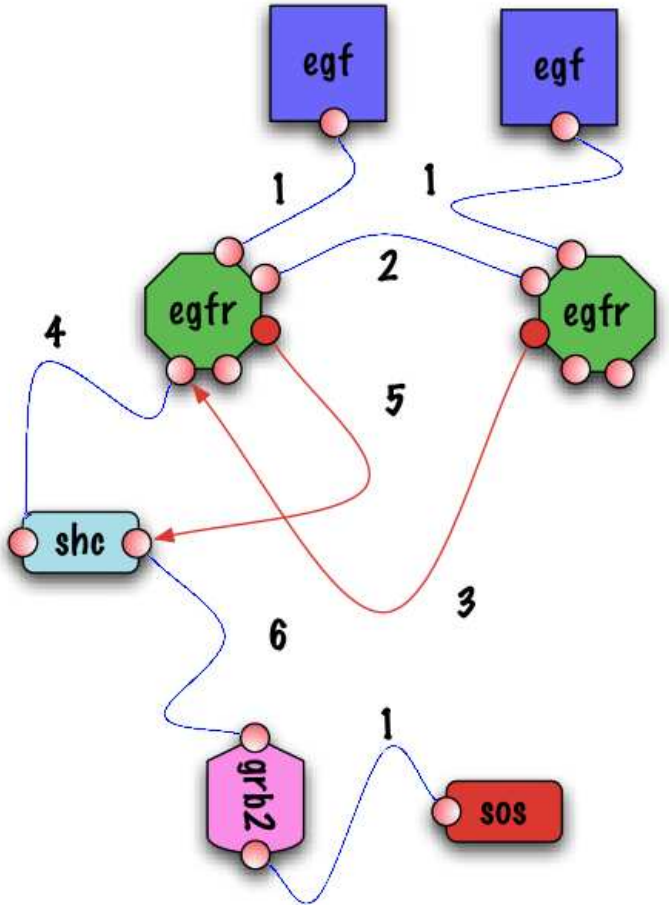
Eikuch, 2007

Contact map

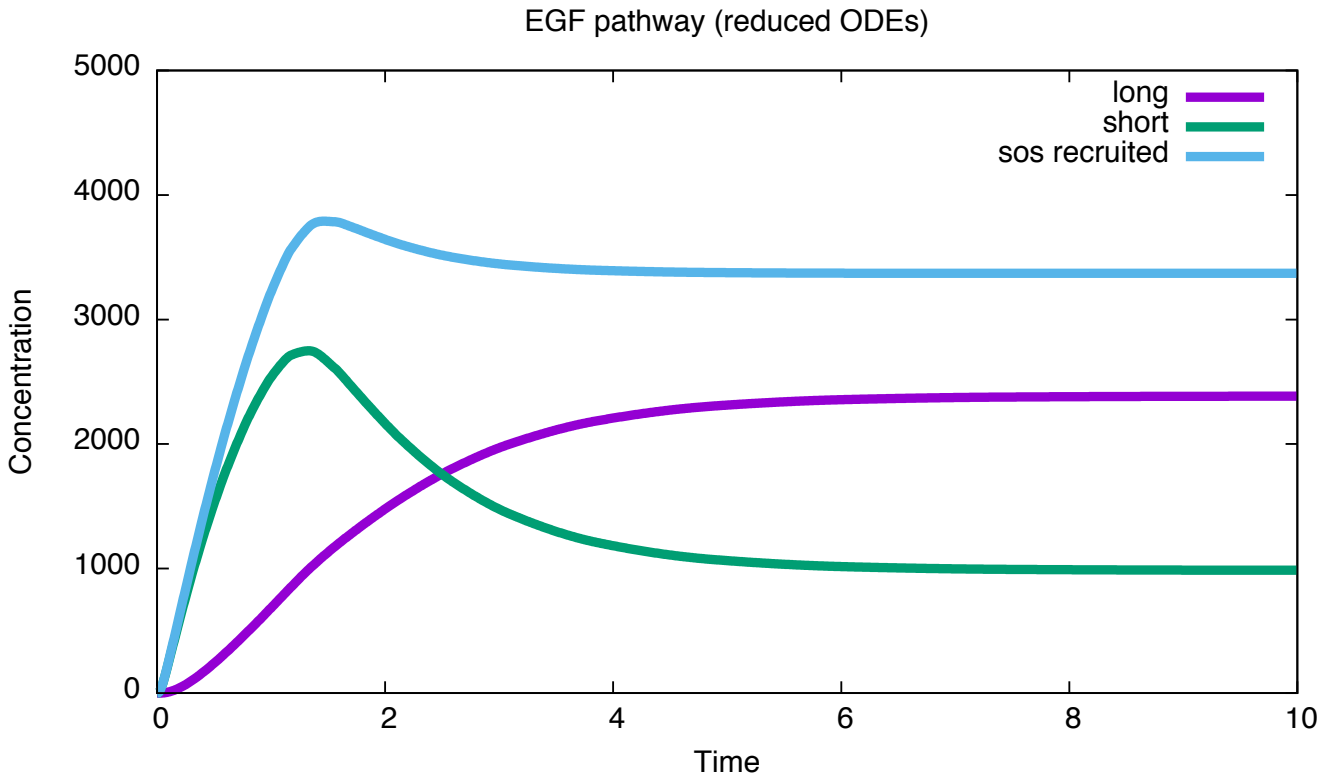


Causal traces

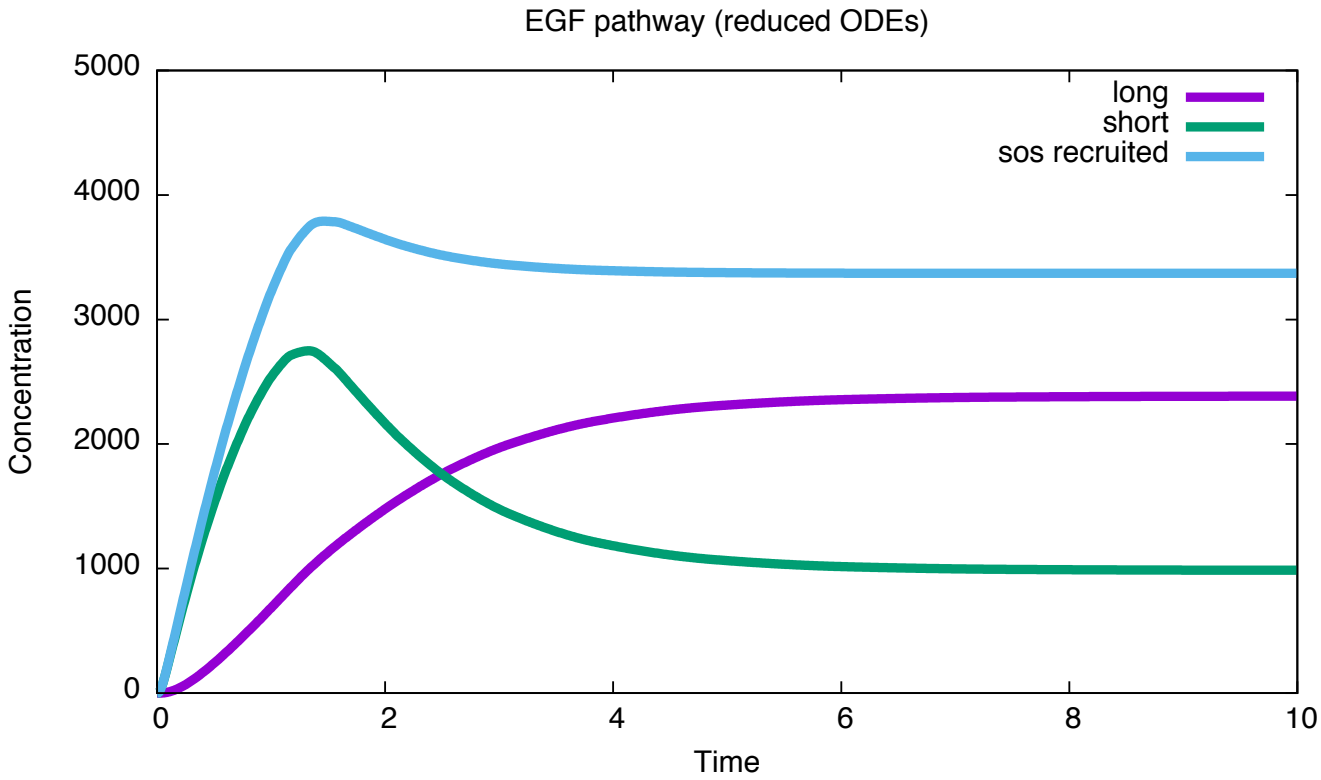




ODE semantics

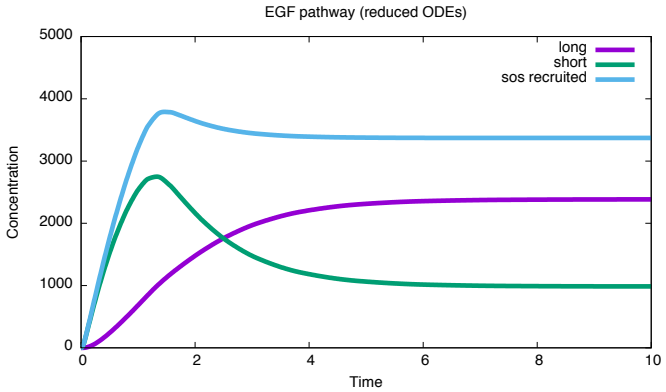


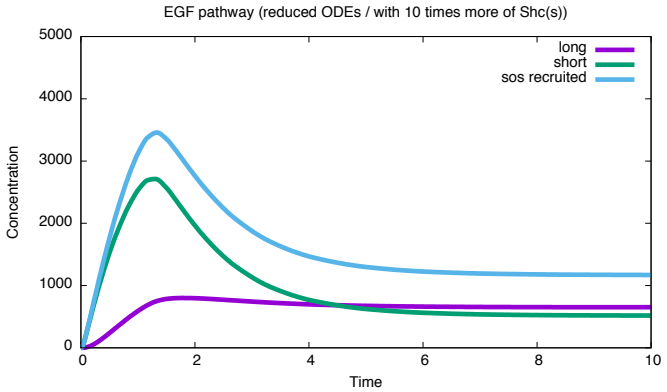
ODE semantics



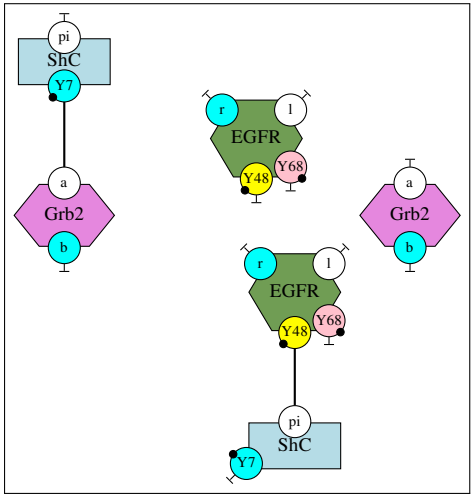
What will happen if more Shc(s) is put in the system?

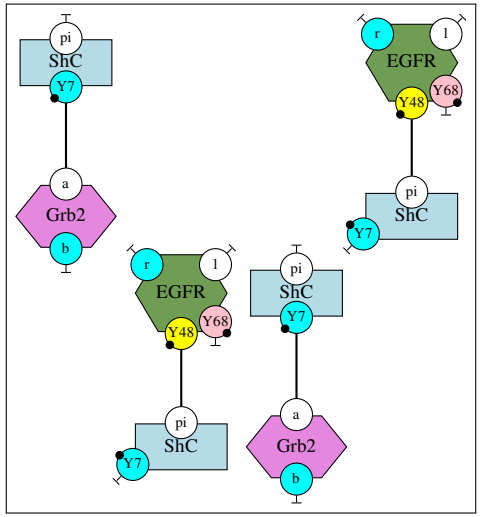
ODE semantics

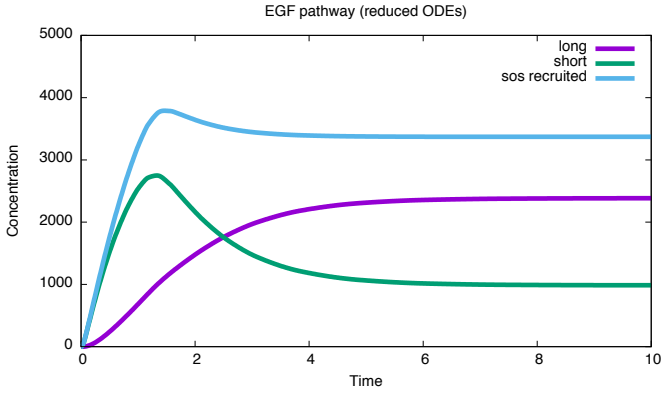


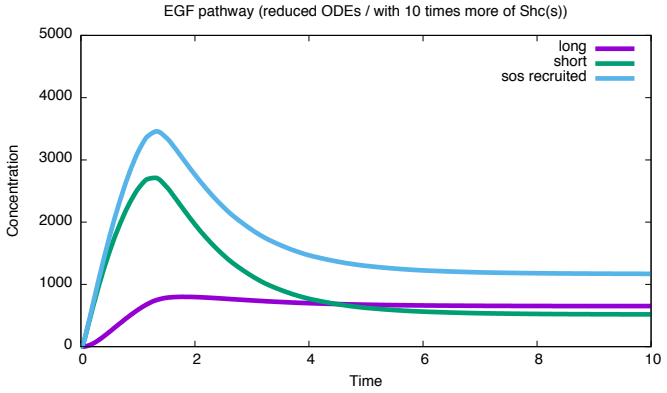


Crowding effect





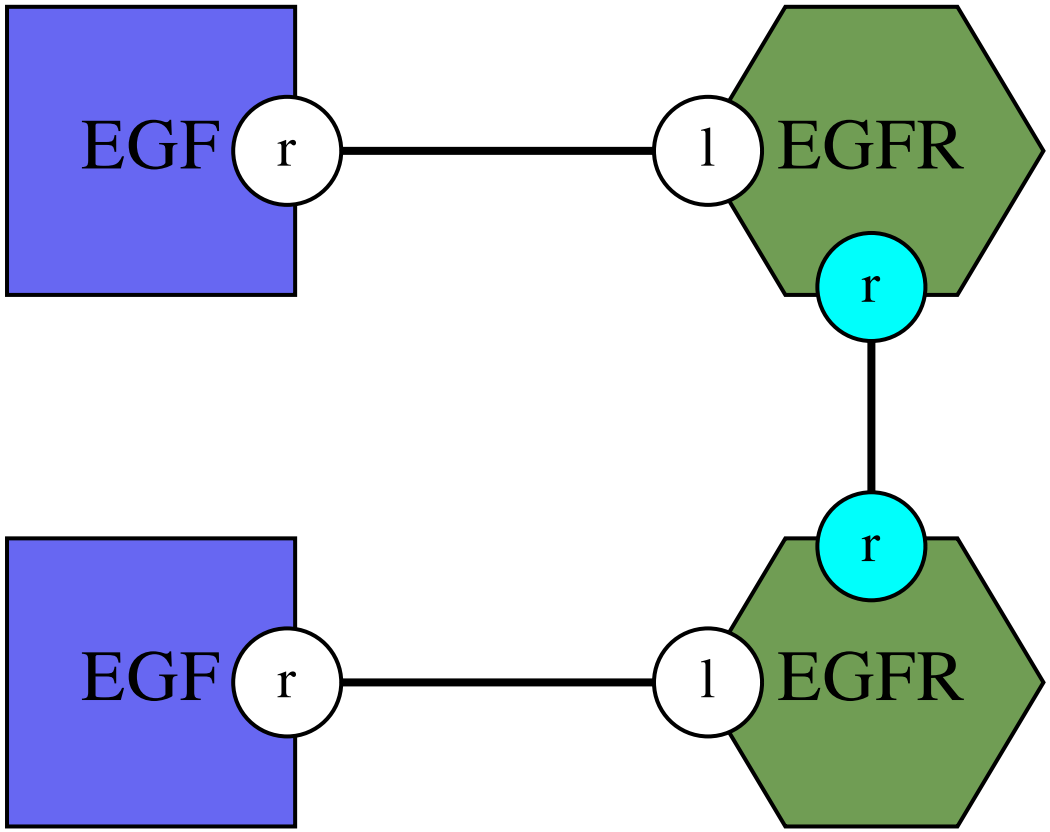




Overview

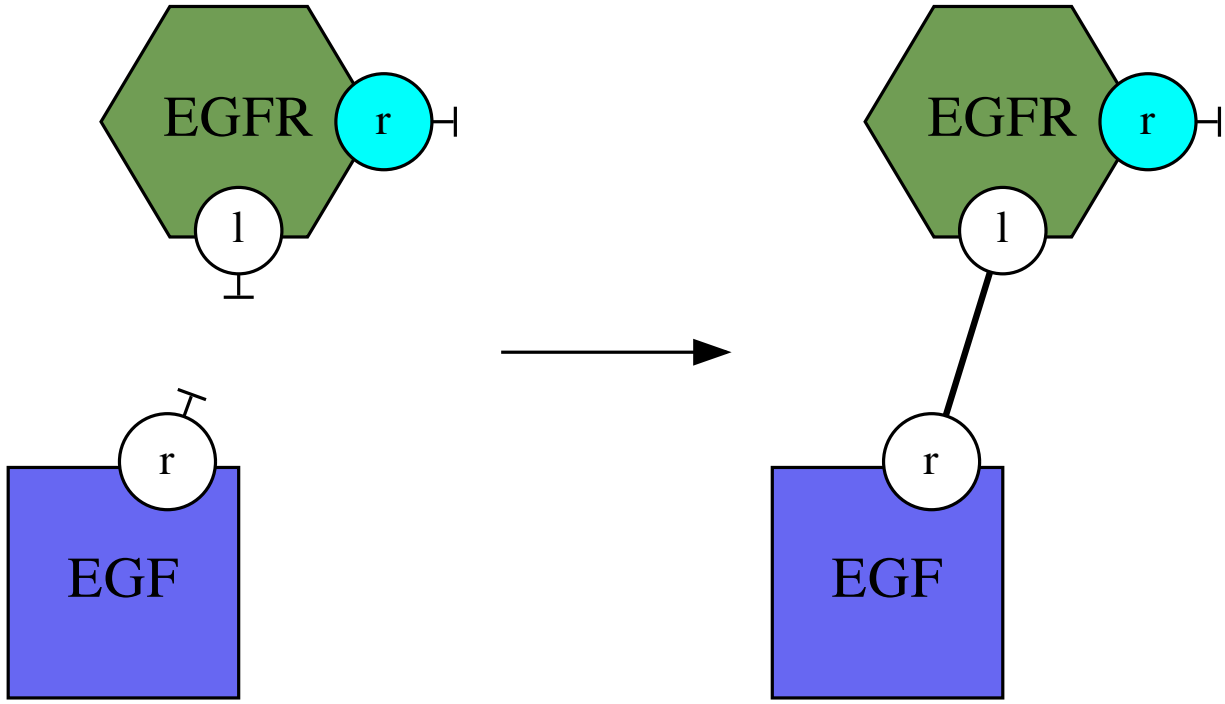
1. Introduction
2. Language: Kappa
3. Abstraction: Local views
4. Completeness: false positives?
5. Local fragment of Kappa
6. Decontextualization
7. Conclusion

A chemical species



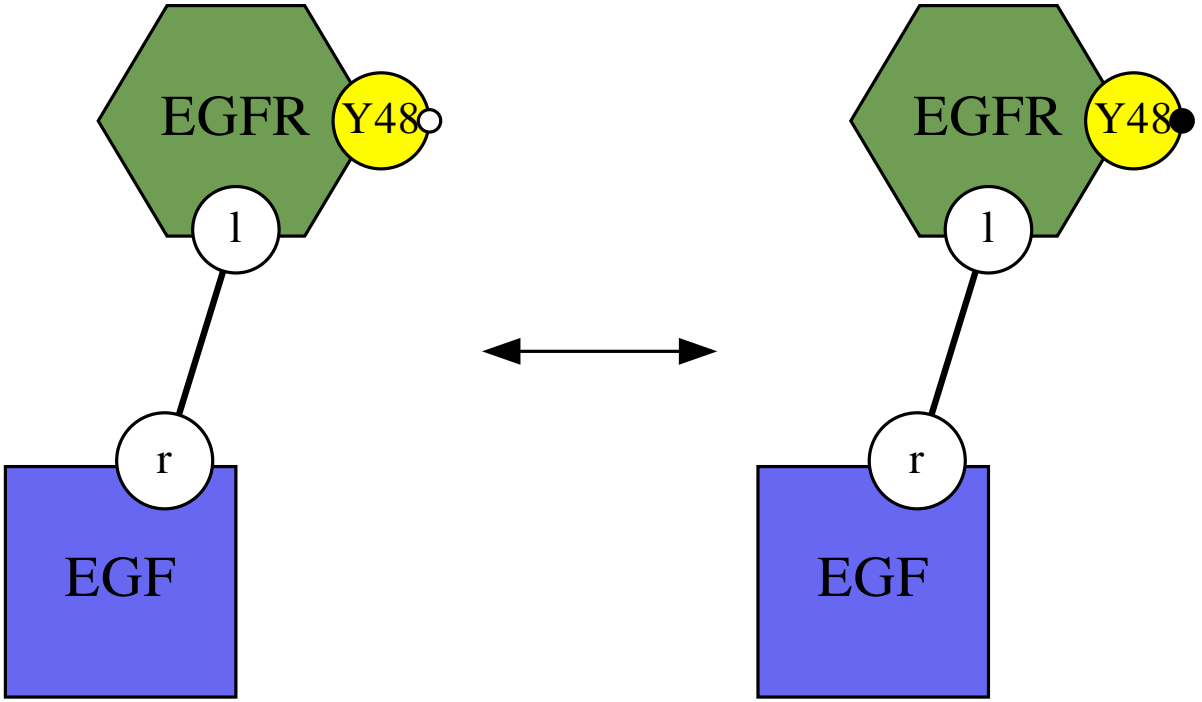
EGF(r[1]), EGFR(l[1],r[2]), EGFR(r[2],l[3]), EGF(r[3])

A Unbinding/Binding Rule



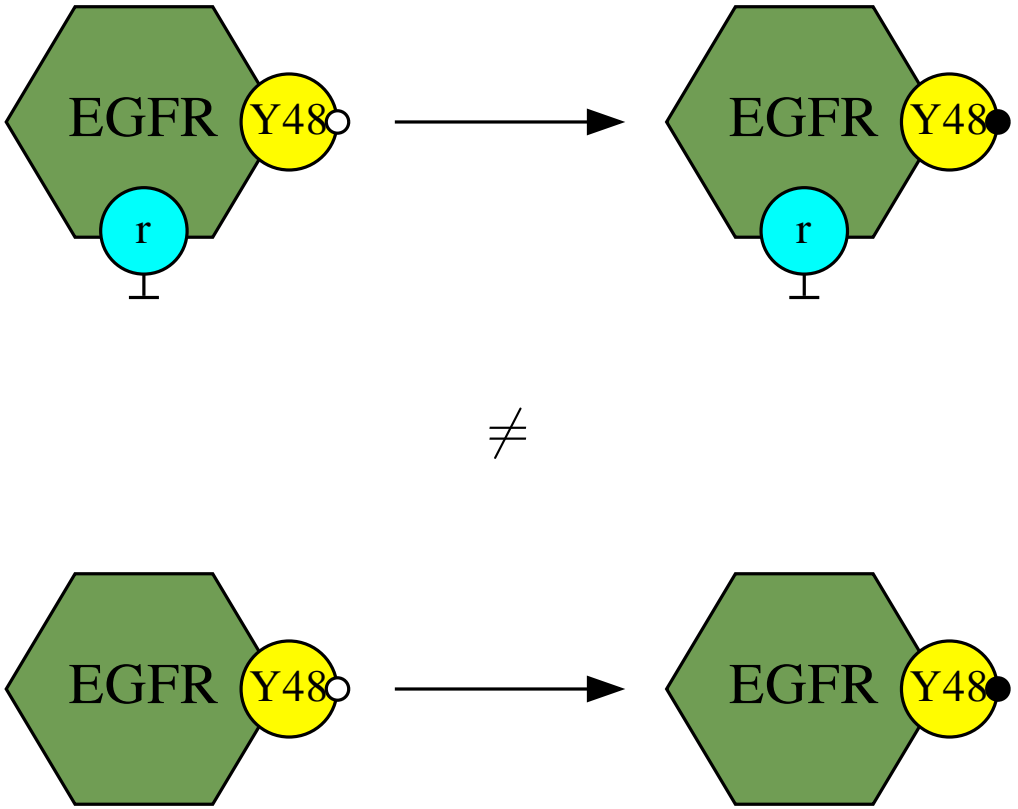
EGF(r[.]), EGFR(l[.],r[.]) $\longleftrightarrow$ EGF(r[1]), EGFR(l[1],r[.])

Internal state

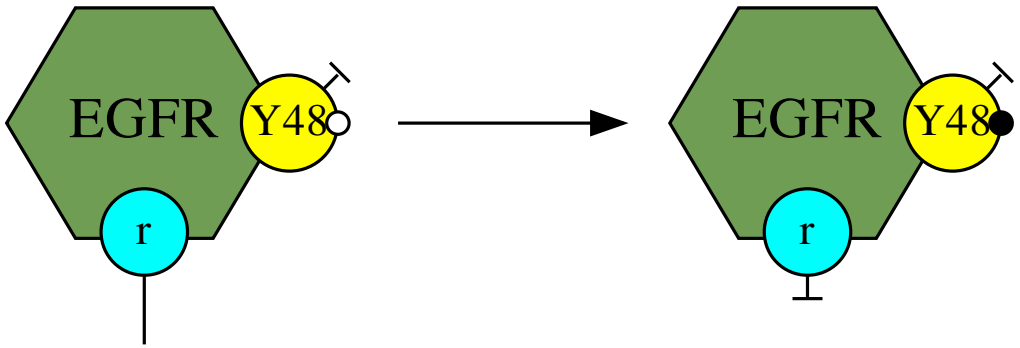


EGFR(Y48{u},l[1]), EGF(r[1]) $\longleftrightarrow$ EGFR(Y48{p},l[1]), EGF(r[1])

Don’t care, Don’t write

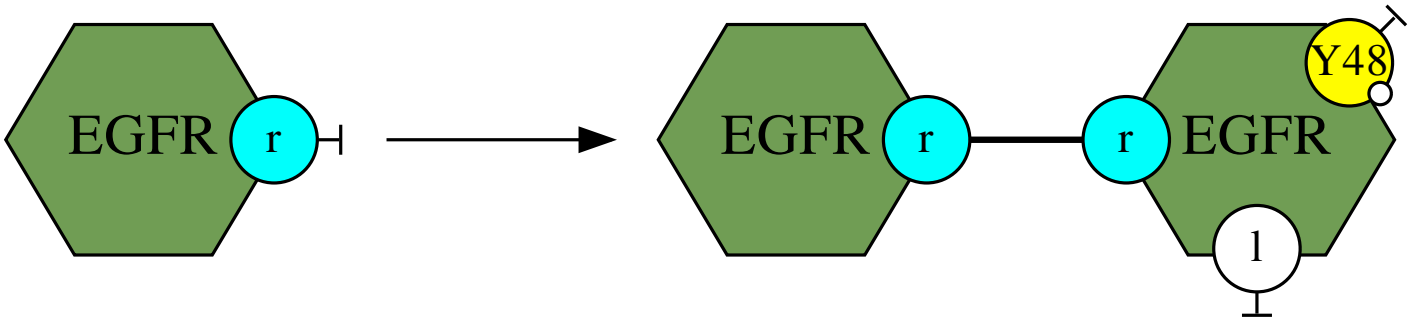


A contextual rule

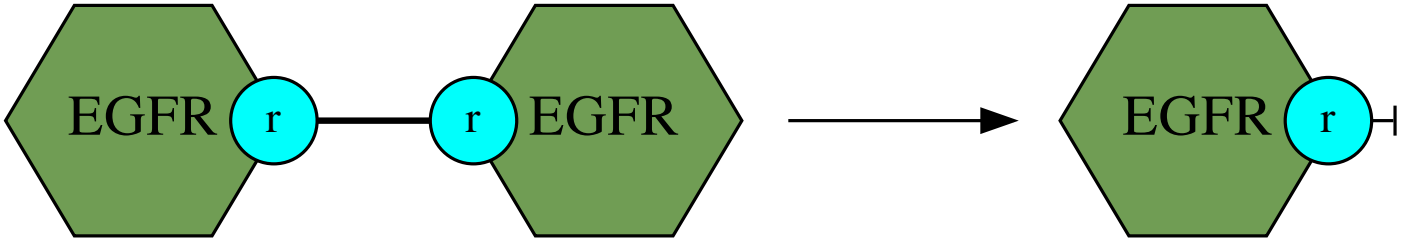


$$\mathrm{EGFR(Y48\{u\}[.],r[\_])} \rightarrow \mathrm{EGFR(Y48\{p\}[.],r[.])}$$

Creation/Suppression



$R(r[.]) \rightarrow R(r[1]), R(r[1],l[.],Y48\{u\}[.])$



$R(r[1]), R(r[1]) \rightarrow R(r[.])$

Early EGF example

egf rules 1

protein shorthands: E:=egf, R:=egfr, So:=Sos,Sh:=Sh,G:=grb2
site abbreviations & fusions: Y68:=Y1068, Y48:=Y1148/73, Y7:=Y317, π:=PTB/SH2

- Ligand-receptor binding, receptor dimerisation, rtk x-phosph, & de-phosph
  - 01: R(l,r), E(r) <-> R(l^1,r), E(r^1)
  - 02: R(l^1,r), R(l^2,r) <-> R(l^1,r^3), R(l^2,r^3)
  - 03: R(r^1,Y68) -> R(r^1,Y68^p)
  - R(Y68^p) -> R(Y68)
  - 04: R(r^1,Y48) -> R(r^1,Y48^p)
  - R(Y48^p) -> R(Y48)

receptor type: R(l,r,Y68,Y48)

- Sh x-phosph & de-phosph
  - 14: R(r^2,Y48^{p1}), Sh(π^1,Y7) -> R(r^2,Y48^{p1}), Sh(π^1,Y7^p)
  - ??: Sh(π^1,Y7^p) -> Sh(π^1,Y7)
  - 16: Sh(π,Y7^p) -> Sh(π,Y7)

refined from Sh(Y7^p)-> Sh(Y7)

- Y68-G binding
  - 09: R(Y68^p), G(a,b) <-> R(Y68^{p1})+G(a^1,b)
  - 11: R(Y68^p), G(a,b^2) <-> R(Y68^{p1})+G(a^1,b^2)

refined from R(Y68^p)+G(a)<->R(Y68^{p1})+G(a^1)

Early EGF example

egf rules 2

refined from
So(d)+G(b)<->So(d^1)+G(b^1)

interface note: highlight the interacting parts

- G-So binding
  - 10: R(Y68^{p1}), G(a^1,b), So(d) <-> R(Y68^{p1}), G(a^1,b^2), So(d^2)
  - 12: G(a,b), So(d) <-> G(a,b^1), So(d^1)
  - 22: Sh(π,Y7^{p2}), G(a^2,b), So(d) <-> Sh(π,Y7^{p2}), G(a^2,b^1), S(d^1)
  - 19: Sh(π^1,Y7^{p2}), G(a^2,b), So(d) <-> Sh(π^1,Y7^{p2}), G(a^2,b^1), S(d^1)
- Y48-Sh binding
  - 13: R(Y48^{p}), Sh(π,Y7) <-> R(Y48^{p1}), Sh(π^1,Y7)
  - 15: R(Y48^{p}), Sh(π,Y7^{p}) <-> R(Y48^{p1}), Sh(π^1,Y7^{p})
  - 18: R(Y48^{p}), Sh(π,Y7^{p1}), G(a^1,b) <-> R(Y48^{p2}), Sh(π^2,Y7^{p1}), G(a^1,b)
  - 20: R(Y48^{p}), Sh(π,Y7^{p1}), G(a^1,b^3), S(d^3) <-> R(Y48^{p2}), Sh(π^2,Y7^{p1}), G(a^1,b^3), S(d^3)

refined from
R(Y48^{p})+Sh(π)<->R(Y48^{p1})+Sh(π^1)

why not simply G(b^3)??

- Sh-G binding
  - 17: R(Y48^{p1}), Sh(π^1,Y7^{p}), G(a,b) <-> R(Y48^{p1}), Sh(π^1,Y7^{p2}), G(a^2,b)
  - 21: Sh(π,Y7^{p}), G(a,b) <-> Sh(π,Y7^{p1}), G(a^1,b)
  - 23: Sh(π,Y7^{p}), G(a,b^2) <-> Sh(π,Y7^{p1}), G(a^1,b^2)
  - 24: R(Y48^{p1}), Sh(π^1,Y7^{p}), G(a,b^3), S(d^3) <-> R(Y48^{p1}), Sh(π^1,Y7^{p2}), G(a^2, b^3), S(d^3)

refined from
Sh(π), G(a)<->Sh(π^1), G(a^1)

Properties of interest

1. Show the absence of modeling errors:
   - detect dead rules;
   - detect overlapping rules;
   - detect non exhaustive interactions;
   - detect rules with ambiguous molecularity.
2. Get idiomatic description of the networks:
   - capture causality;
   - capture potential interactions;
   - capture relationships between site states;
   - simplify rules.
3. Allow fast simulation:
   - capture accurate approximation of the wake-up relation.

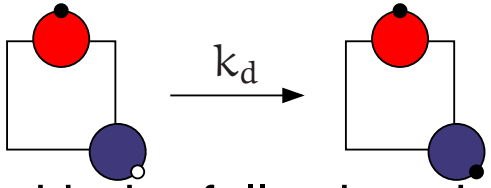
Overview

1. Introduction
2. Language: Kappa
3. Abstraction: Local views
4. Completeness: false positives?
5. Local fragment of Kappa
6. Decontextualization
7. Conclusion

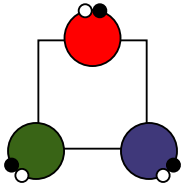
Concrete semantics

A rule is a symbolic representation of a multi-set of reactions.

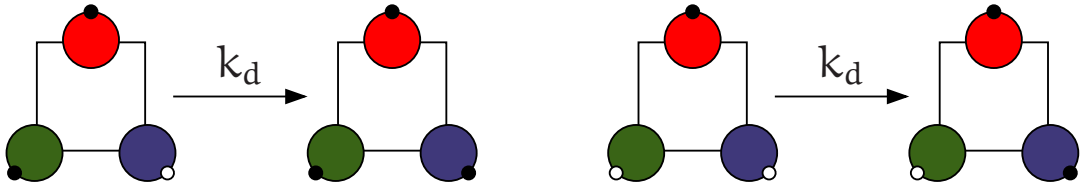
For instance, the rule:



within a model with the following signature:



denotes the following two reactions:



Set of reachable chemical species

Let $\mathcal{R} = \{R_i\}$ be a set of rules.
Let *Species* be the set of all chemical species ($C, c_1, c'_1, \ldots, c_k, c'_k, \ldots \in$ *Species*).
Let $Species_0$ be the set of initial .

We are interested in $Species_\omega$ the set of all chemical species that can be constructed in one or several applications of the reactions induced by the rules in $\mathcal{R}$, starting from the set $Species_0$ of initial chemical species.

(We do not care about the number of occurrences of each chemical species).

Inductive definition

We define the mapping $\mathbb{F}$ as follows:

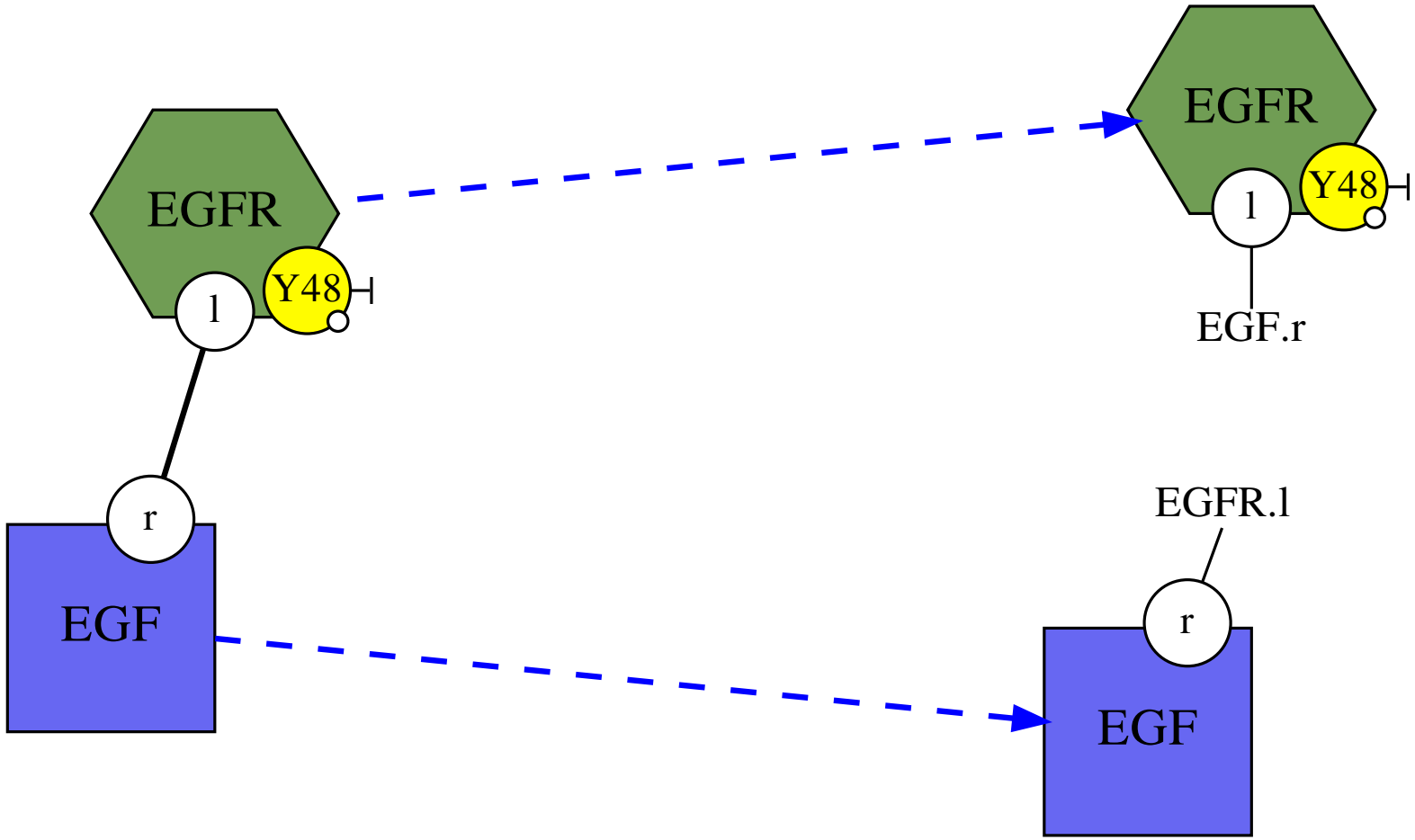
$$\mathbb{F} : \begin{cases} \wp(\mathit{Species}) & \to \wp(\mathit{Species}) \\ X & \mapsto X \cup \left\{ c'_j \;\middle|\; \begin{array}{l} \exists R_k \in \mathcal{R}, c_1, \ldots, c_m \in X, \\ c_1, \ldots, c_m \to_{R_k} c'_1, \ldots, c'_n \end{array} \right\}. \end{cases}$$

The set $\wp(\mathit{Species})$ is a complete lattice.
The mapping $\mathbb{F}$ is an extensive $\cup$-complete morphism.

We define the set of reachable chemical species as follows:

$$\mathit{Species}_\omega = \bigcup \left\{ \mathbb{F}^n(\mathit{Species}_0) \;\middle|\; n \in \mathbb{N} \right\}.$$

Local views



$\alpha(\{R(Y48\{u\}[.],l[1]),\ E(r[1])\}) = \{R(Y48\{u\}[.],l[r.E]);\ E(r[l.R])\}.$

Galois connection

Let $\mathit{Local\_view}$ be the set of all local views.

Let $\alpha \in \wp(\mathit{Species}) \rightarrow \wp(\mathit{Local\_view})$ be the function that maps any set of chemical species into the set of their local views.

The set $\wp(\mathit{Local\_view})$ is a complete lattice.
The function α is a $\cup$-complete morphism.

Thus, it defines a Galois connection:

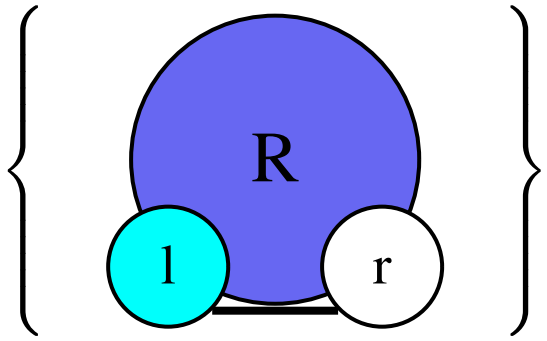
$$\wp(\mathit{Species}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \wp(\mathit{Local\_view}).$$

(The function γ maps a set of local views into the set of complexes that can be built with these local views).

$\gamma \circ \alpha$

$\gamma \circ \alpha$ is an upper closure operator: it abstracts away some information.

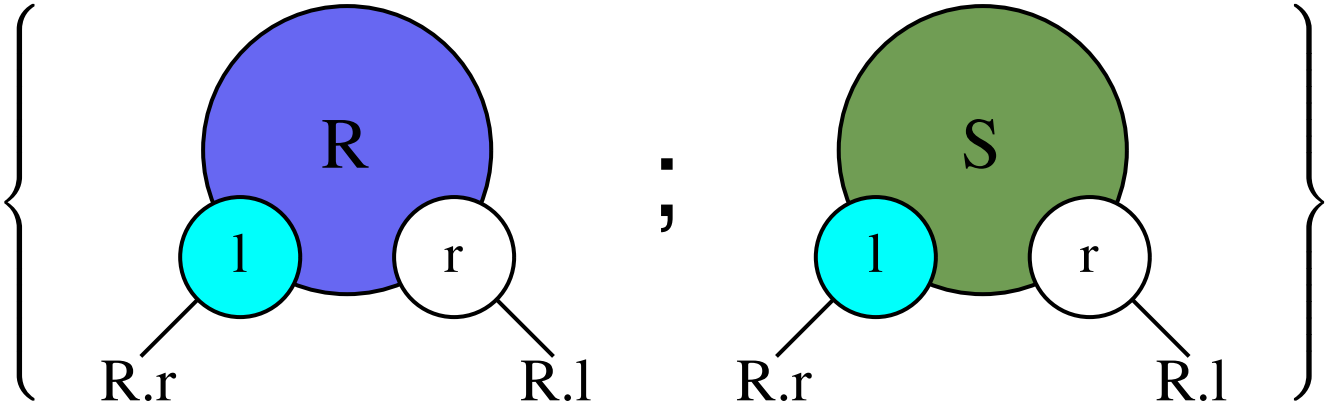
Guess the image of the following set of chemical species ?



$\alpha \circ \gamma$

$\alpha \circ \gamma$ is a lower closure operator: it simplifies (or reduces) constraints.

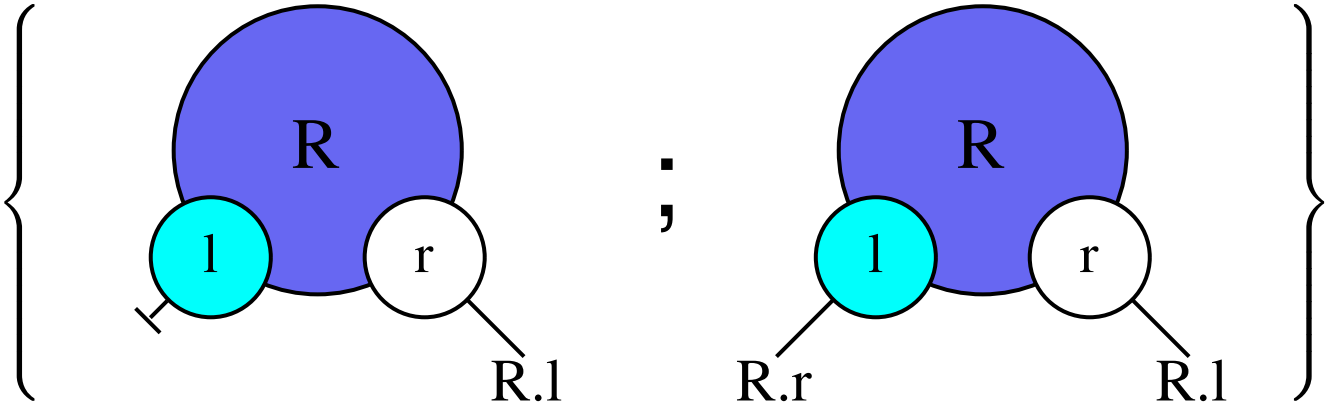
Guess the image of the following set of local views ?



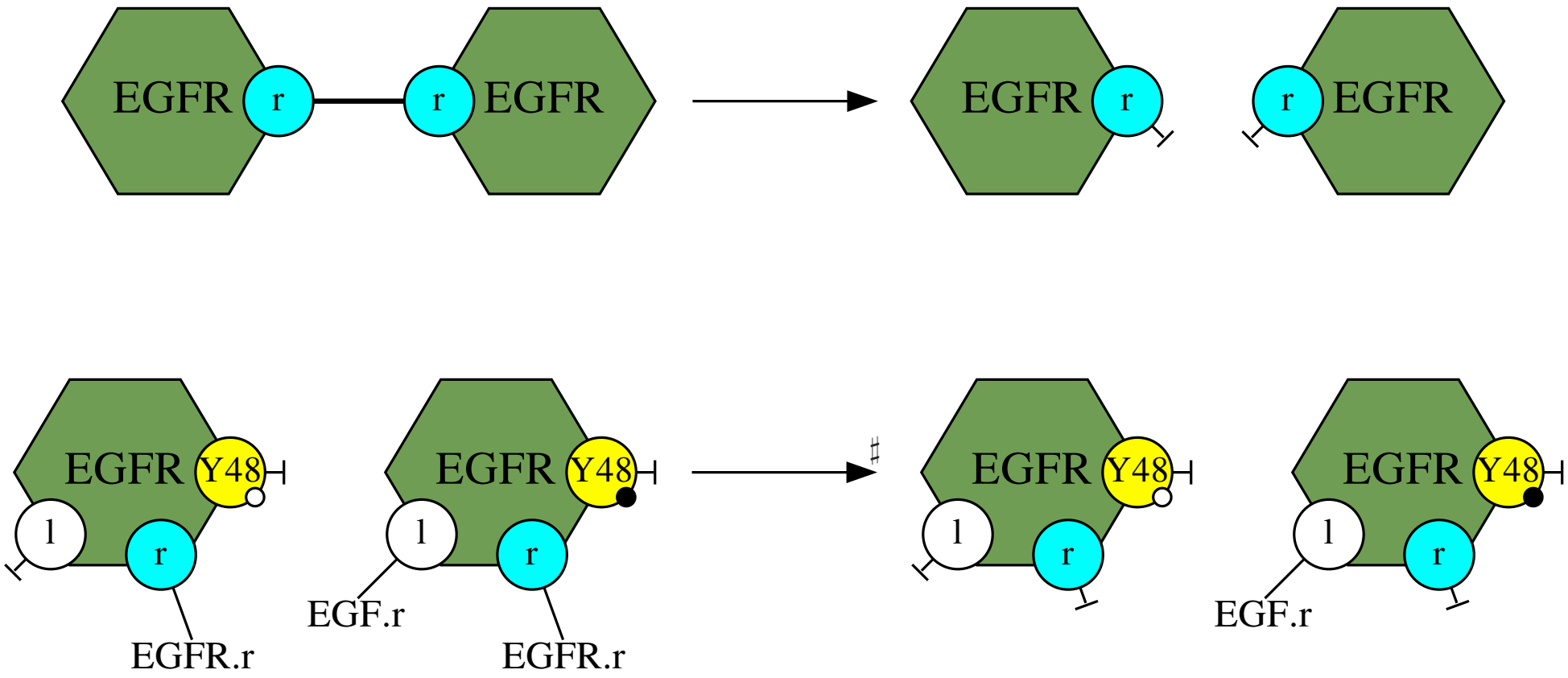
One more question

$\alpha \circ \gamma$ is a lower closure operator: it simplifies (or reduces) constraints.

Guess the image of the following set of local views ?



Abstract reactions



Abstract counterpart to $\mathbb{F}$

We define $\mathbb{F}^\sharp$ as:

$$\mathbb{F}^\sharp : \begin{cases} \wp(\mathit{Local\_view}) & \to \wp(\mathit{Local\_view}) \\ Y & \mapsto Y \cup \left\{ \mathit{lv}'_j \;\middle|\; \begin{array}{l} \exists R_k \in \mathcal{R}, \mathit{lv}_1, \ldots, \mathit{lv}_m \in Y, \\ \mathit{lv}_1, \ldots, \mathit{lv}_m \to^\sharp_{R_k} \mathit{lv}'_1, \ldots, \mathit{lv}'_n \end{array} \right\}. \end{cases}$$

We have:

- $\mathbb{F}^\sharp$ is extensive;
- $\mathbb{F}^\sharp$ is monotonic;
- $\mathbb{F} \circ \gamma \mathrel{\dot\subseteq} \gamma \circ \mathbb{F}^\sharp$;
- $\mathbb{F}^\sharp \circ \alpha = \alpha \circ \mathbb{F} \circ \gamma \circ \alpha$ (we will see later why).

Soundness

Theorem 1 Let:

1. $(D, \subseteq, \cup)$ and $(D^\sharp, \sqsubseteq, \sqcup)$ be chain-complete partial orders;
2. $D \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} D^\sharp$ be a Galois connection;
3. $\mathbb{F} \in D \to D$ and $\mathbb{F}^\sharp \in D^\sharp \to D^\sharp$ be monotonic mappings such that: $\mathbb{F} \circ \gamma \dot{\subseteq} \gamma \circ \mathbb{F}^\sharp$;
4. $X_0 \in D$ be an element such that: $X_0 \subseteq \mathbb{F}(X_0)$;

Then:

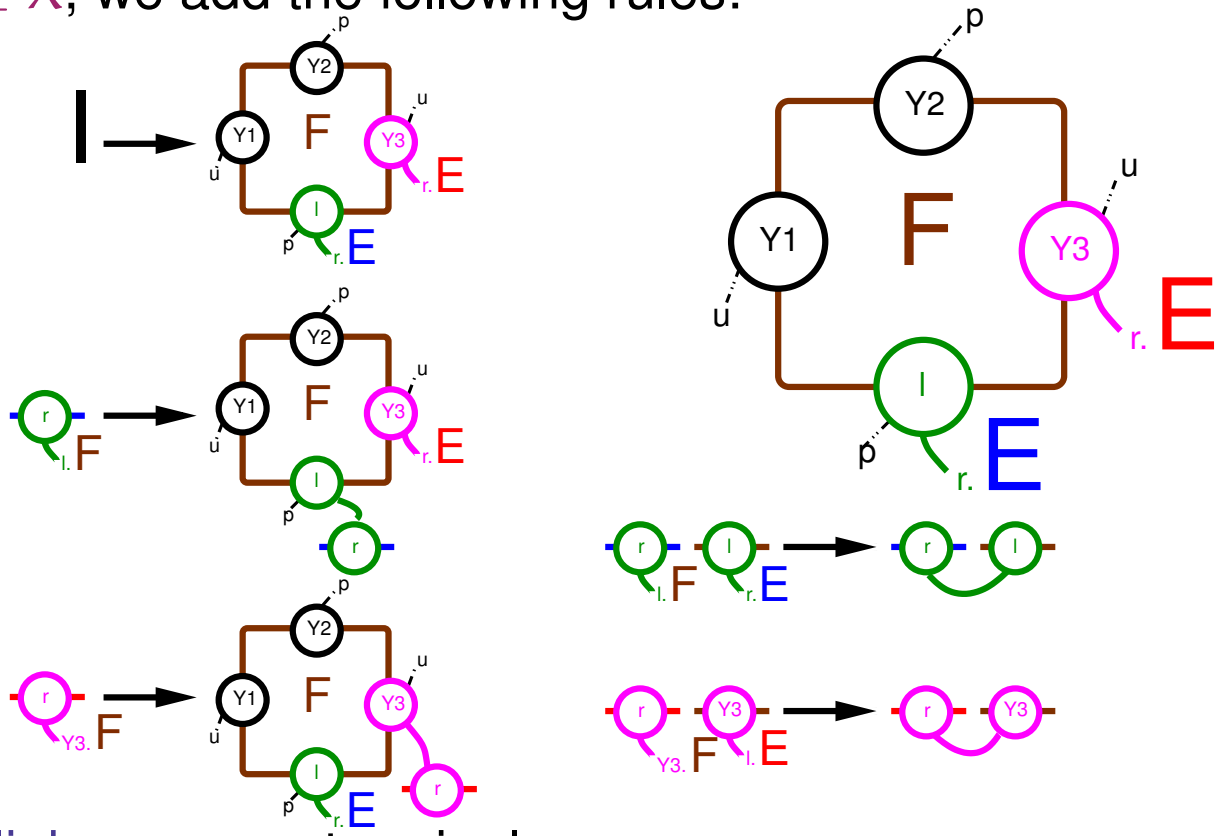
1. both $\mathit{lfp}_{X_0}\mathbb{F}$ and $\mathit{lfp}_{\alpha(X_0)}\mathbb{F}^\sharp$ exist,
2. $\mathit{lfp}_{X_0}\mathbb{F} \subseteq \gamma(\mathit{lfp}_{\alpha(X_0)}\mathbb{F}^\sharp)$.

Overview

1. Introduction
2. Language: Kappa
3. Abstraction: Local views
4. Completeness: false positives?
5. Local fragment of Kappa
6. Decontextualization
7. Conclusion

From views to species

For any $X \in \wp(Local\_view)$, $\gamma(X)$ is given by a rewrite system:
For any $lv \in X$, we add the following rules:



I and semi-links are non-terminal.
I is the initial symbol.

Pumping lemma

- We use this rewrite system to enumerate the chemical species of $\gamma(X)$.
- There are two cases:
  1. either there is a finite number of rewrite sequences;
  2. or we encounter cyclic derivations
     i.e. an open chemical species with a cycle of the following form:

     R.l-r.E ... R.l-r.E

     can be built.
- We only enumerate chemical species that are reached through an acyclic rewriting computation.
- It turns out that: if $X \in \alpha(\wp(\mathit{Species}))$ then each rewrite sequence is the prefix of a terminating rewrite sequence.

  (So there is an unbounded number of species if, and only if, there is an unbounded number of rewrite sequences.)

Examples

1. Make the demo for egf
2. Make the demo for fgf
3. Make the demo for Global invariants

Counting chemical species

Given a set of local views X, we can easily count the number of species in $\gamma(X)$ by using the following lemmas:

Lemma 1 (rigidity) An embedding between two connected components is fully characterized by the image of one agent.

Lemma 2 (automorphism) If $\gamma(X)$ is finite, then for any $C \in \gamma(X)$:

- C has at most two automorphisms;
- if C has two automorphisms, then C has a bond of the form $R.r - r.R$. Moreover one automorphism swaps the two R of this bond.

Lemma 3 (Euler) If a chemical species has no cycle, then it has an agent with only one site.

sketch the algorithm

Which information is abstracted away ?

Our analysis is exact (no false positive):

- for EGF cascade (356 chemical species);
- for FGF cascade (79080 chemical species);
- for SBF cascade (around 10^{19} chemical species).

We know how to build systems with false positives. . .
. . .but they seem to be biologically meaningless.

This raises the following issues:

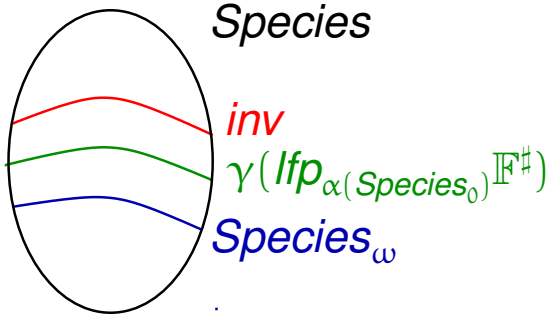
- Can we characterize which information is abstracted away ?
- Which is the form of the systems, for which we have no false positive ?
- Do we learn something about the biological systems that we describe ?

Which information is abstracted away ?

Theorem 2 We suppose that:

1. $(D, \subseteq)$ be a partial order;
2. $(D^\sharp, \sqsubseteq, \sqcup)$ be chain-complete partial order;
3. $D \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} D^\sharp$ be a Galois connection;
4. $\mathbb{F} \in D \to D$ and $\mathbb{F}^\sharp \in D^\sharp \to D^\sharp$ are monotonic;
5. $\mathbb{F} \circ \gamma \dot{\subseteq} \gamma \circ \mathbb{F}^\sharp$;
6. $X_0, \mathit{inv} \in D$ such that:
   - $X_0 \subseteq \mathbb{F}(X_0) \subseteq \mathbb{F}(\mathit{inv}) \subseteq \mathit{inv}$,
   - $\mathit{inv} = \gamma(\alpha(\mathit{inv}))$,
   - and $\alpha(\mathbb{F}(\mathit{inv})) = \mathbb{F}^\sharp(\alpha(\mathit{inv}))$;

Then, $\mathit{lfp}_{\alpha(X_0)}\mathbb{F}^\sharp$ exists and $\gamma(\mathit{lfp}_{\alpha(X_0)}\mathbb{F}^\sharp) \subseteq \mathit{inv}$.



Proof I/III

We have already seen (previous lectures) that:

1. $\mathit{lfp}_{\alpha(X_0)}\mathbb{F}^\sharp$ exists;
2. there exists an ordinal δ such that $\mathit{lfp}_{\alpha(X_0)}\mathbb{F}^\sharp = \mathbb{F}^{\sharp\delta}(\alpha(X_0))$.

Proof II/III

Let us show that $\gamma(\mathit{lfp}_{\alpha(X_0)}\mathbb{F}^\sharp) \subseteq \mathit{inv}$.

Let us prove instead by induction over δ that $\mathbb{F}^{\sharp\delta}(\alpha(X_0)) \sqsubseteq \alpha(\mathit{inv})$.

- If $Y \in D^\sharp$ is an element such that $Y \sqsubseteq \alpha(\mathit{inv})$,
  $\mathbb{F}^\sharp(Y) \sqsubseteq \mathbb{F}^\sharp(\alpha(\mathit{inv}))$ ($\mathbb{F}^\sharp$ is mon)
  $\mathbb{F}^\sharp(\alpha(\mathit{inv})) = \alpha(\mathbb{F}(\mathit{inv}))$ (assumption)
  $\alpha(\mathbb{F}(\mathit{inv})) \sqsubseteq \alpha(\mathit{inv})$. ($\alpha$ is mon and inv is a post)

  Thus: $\mathbb{F}^\sharp(Y) \sqsubseteq \alpha(\mathit{inv})$
- If $Y_i \in D^{\sharp I}$ is a chain of elements such that $Y_i \sqsubseteq \alpha(\mathit{inv})$ for any $i \in I$, then, $\sqcup Y_i \sqsubseteq \alpha(\mathit{inv})$ (lub).

So: $\mathbb{F}^{\sharp\delta}(\alpha(X_0)) \sqsubseteq \alpha(\mathit{inv})$.

Proof III/III

We have:

$$\mathbb{F}^{\sharp\delta}(\alpha(X_0)) \sqsubseteq \alpha(\mathit{inv}).$$

Since γ is monotonic:

$$\gamma(\mathbb{F}^{\sharp\delta}(\alpha(X_0))) \subseteq \gamma(\alpha(\mathit{inv})).$$

But, by assumption, $\gamma(\alpha(\mathit{inv})) = \mathit{inv}$.
Thus,

$$\gamma(\mathbb{F}^{\sharp\delta}(\alpha(X_0))) \subseteq \mathit{inv}.$$

When is there no false positive ?

Theorem 3 We suppose that:

1. $(\mathrm{D}, \subseteq, \cup)$ and $(\mathrm{D}^\sharp, \sqsubseteq, \sqcup)$ are chain-complete partial orders;
2. $(\mathrm{D}, \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathrm{D}^\sharp, \sqsubseteq)$ is a Galois connection;
3. $\mathbb{F} : \mathrm{D} \to \mathrm{D}$ is a monotonic map;
4. X_0 is a concrete element such that $\mathrm{X}_0 \subseteq \mathbb{F}(\mathrm{X}_0)$;
5. $\mathbb{F} \circ \gamma \dot{\subseteq} \gamma \circ \mathbb{F}^\sharp$;
6. $\mathbb{F}^\sharp \circ \alpha = \alpha \circ \mathbb{F} \circ \gamma \circ \alpha$.

Then:

- $\mathit{lfp}_{\mathrm{X}_0}\mathbb{F}$ and $\mathit{lfp}_{\alpha(\mathrm{X}_0)}\mathbb{F}^\sharp$ exist;
- $\mathit{lfp}_{\mathrm{X}_0}\mathbb{F} = \gamma(\alpha(\mathit{lfp}_{\mathrm{X}_0}\mathbb{F})) \iff \mathit{lfp}_{\mathrm{X}_0}\mathbb{F} = \gamma(\mathit{lfp}_{\alpha(\mathrm{X}_0)}\mathbb{F}^\sharp)$.

We need to understand under which assumptions $\mathit{lfp}_{\mathrm{X}_0}\mathbb{F} = \gamma(\alpha(\mathit{lfp}_{\mathrm{X}_0}\mathbb{F}))$.

Local set of chemical species

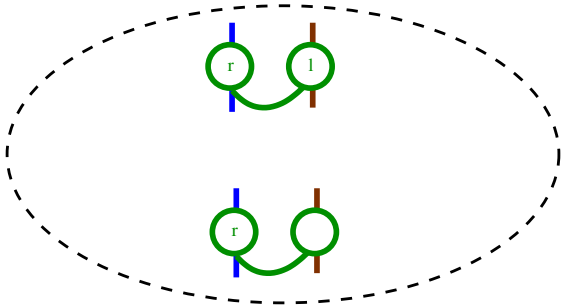
Definition 1 We say that a set $X \in \wp(Species)$ of chemical species is local if and only if $X \in \gamma(\wp(Local\_view))$.

(ie. a set X is local if and only if X is exactly the set of all the species that are generated by a given set of local views.)

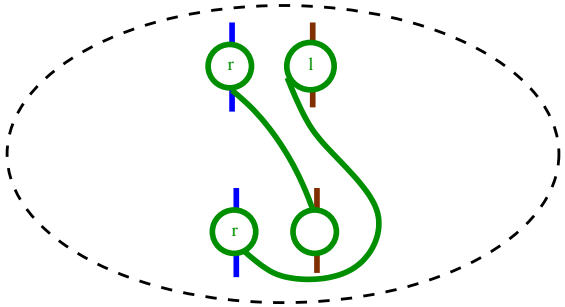
Swapping relation

We define the binary relation $\overset{\text{SWAP}}{\sim}$ among tuples $\mathit{Species}^*$ of chemical species.
We say that $(C_1, \ldots, C_m) \overset{\text{SWAP}}{\sim} (D_1, \ldots, D_n)$ if and only if:

$(C_1, \ldots, C_m)$ matches with



while $(D_1, \ldots, D_n)$ matches with



Swapping closure

Theorem 4 Let $X \in \wp(\mathit{Species})$ be a set of chemical species.

The two following assertions are equivalent:

1. $X = \gamma(\alpha(X))$;
2. for any tuples $(C_i), (D_j) \in \mathit{Species}^*$ such that:
   - $(C_i) \in X^*$,
   - and $(C_i) \overset{\text{SWAP}}{\sim} (D_j)$;

   we have $(D_j) \in X^*$.

Proof (easier implication way)

If:

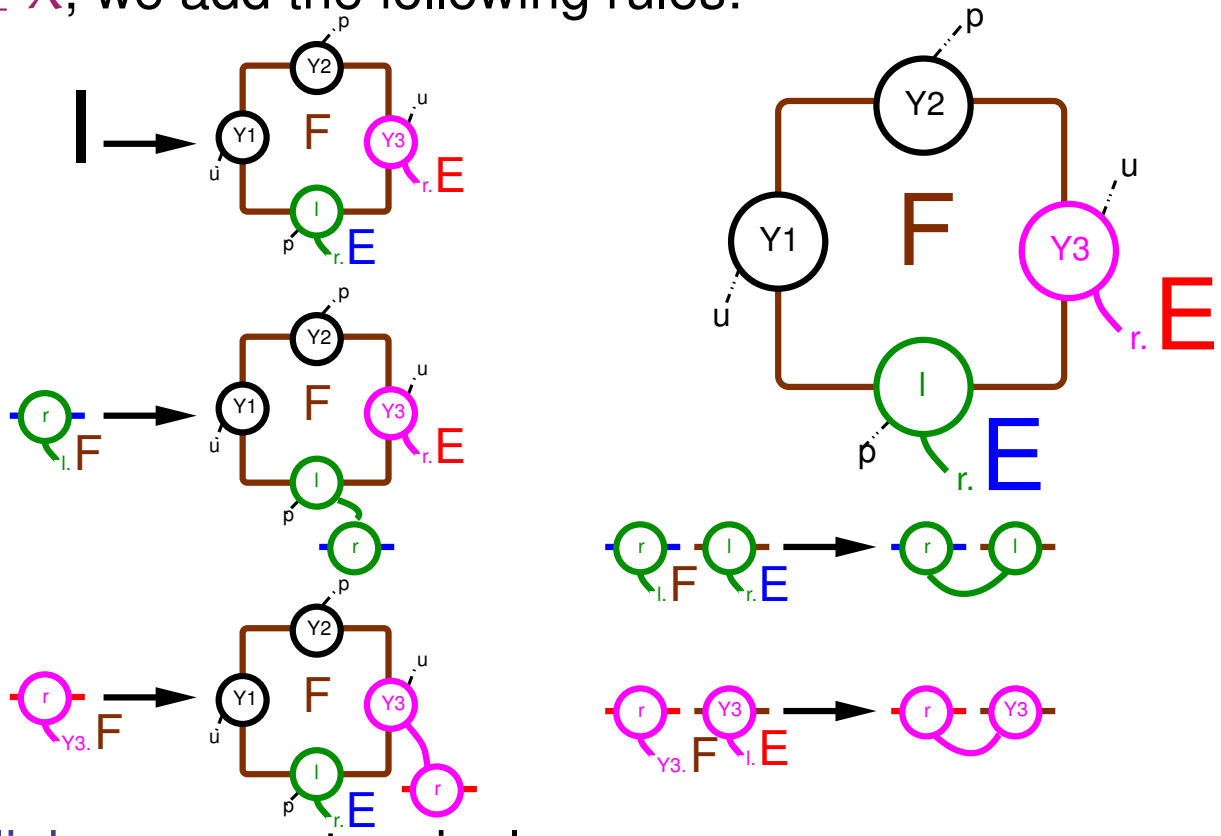
- $X = \gamma(\alpha(X))$,
- $(C_i)_{i \in I} \in X^*$,
- and $(C_i)_{i \in I} \overset{\text{SWAP}}{\sim} (D_j)_{j \in J}$;

Then:

we have $\alpha(\{C_i \mid i \in I\}) = \alpha(\{D_j \mid j \in J\})$ (because $(C_i) \overset{\text{SWAP}}{\sim} (D_j)$)
and $\alpha(\{C_i \mid i \in I\}) \subseteq \alpha(X)$ (because $(C_i) \in X^*$ and α mon);
so $\alpha(\{D_j \mid j \in J\}) \subseteq \alpha(X)$;
so $\{D_j \mid j \in J\} \subseteq \gamma(\alpha(X))$ (by def. of Galois connections);
so $\{D_j \mid j \in J\} \subseteq X$ (since $X = \gamma(\alpha(X))$);
so $(D_j)_{j \in J} \in X^*$.

Proof: more difficult implication way

For any $X \in \wp(Local\_view)$, $\gamma(X)$ is given by a rewrite system:
For any $lv \in X$, we add the following rules:



I and semi-links are non-terminal.
I is the initial symbol.

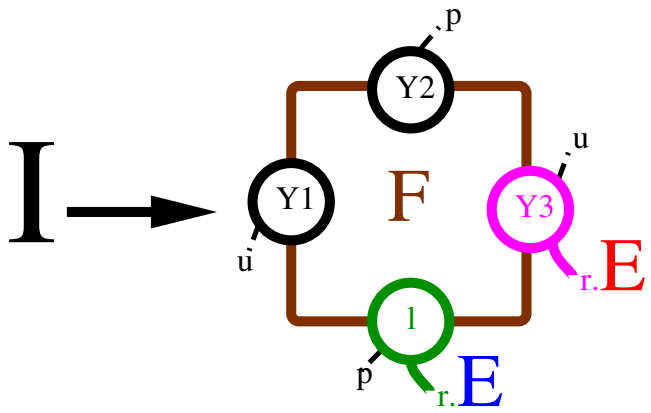
Proof (more difficult implication way)

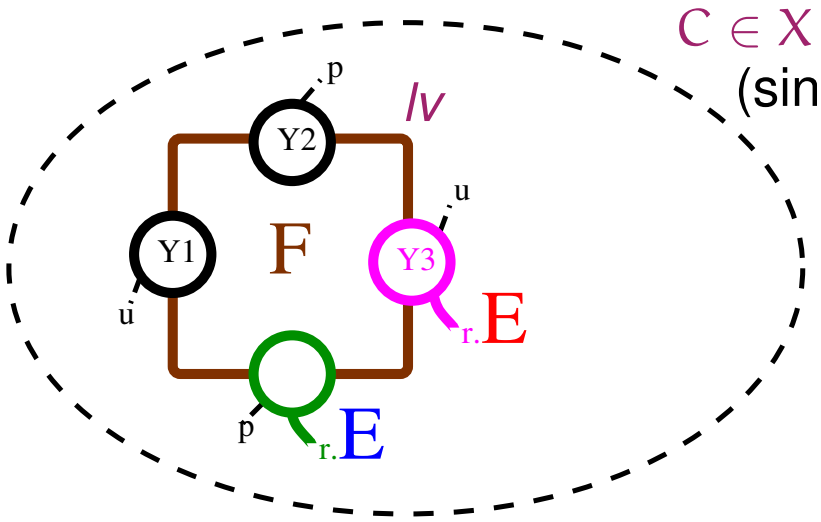
We suppose that X is close with respect to $\overset{\text{SWAP}}{\sim}$.
We want to prove that $\gamma(\alpha(X)) \subseteq X$.

We prove, by induction, that any open complex that can be built by gathering the views of $\alpha(X)$, can be embedded in a complex in X:

- By def. of α, this is satisfied for any local view in $\alpha(X)$;
- This remains satisfied after unfolding a semi-link with a local view;
- This remains satisfied after binding two semi-links.

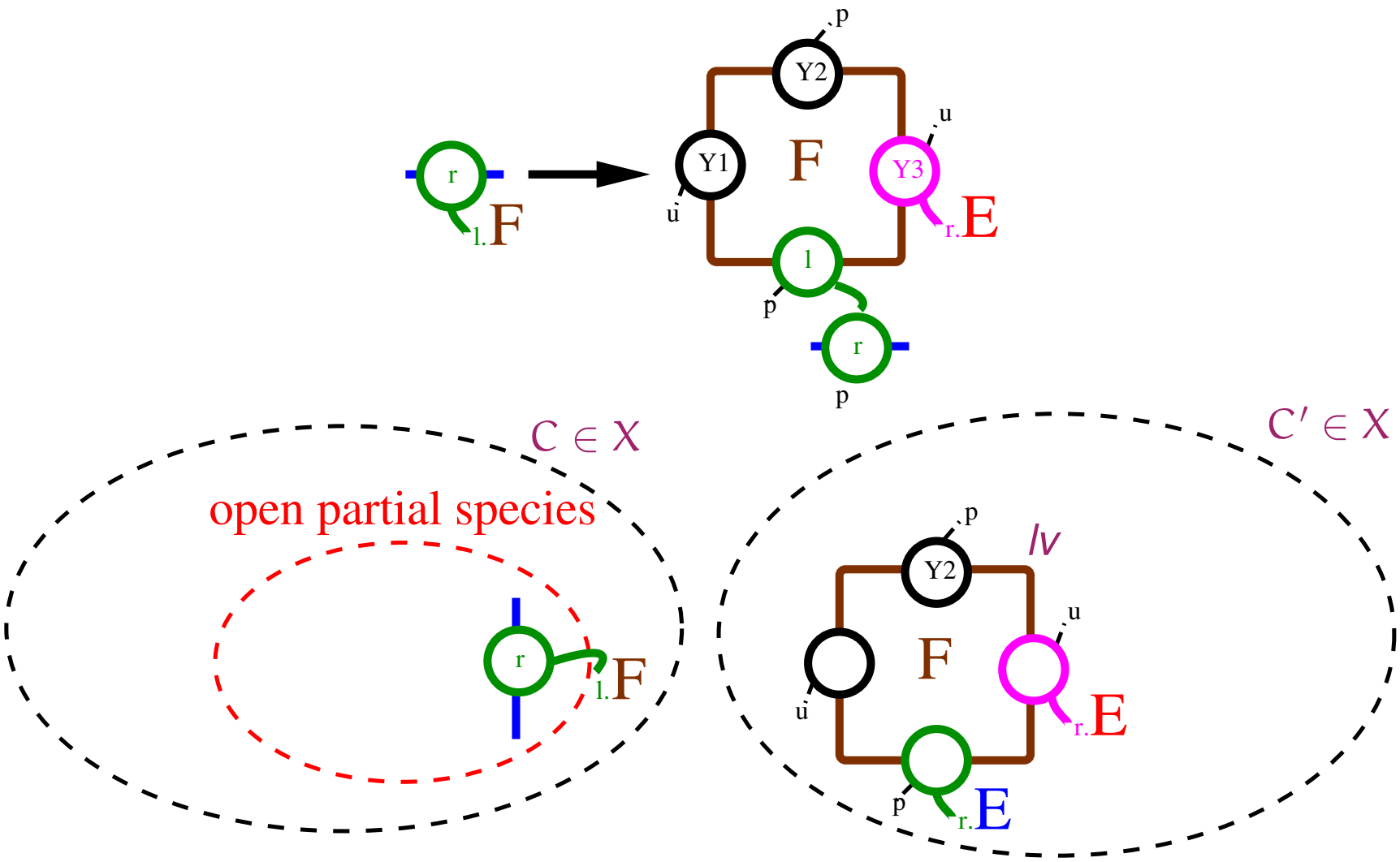
Initialization



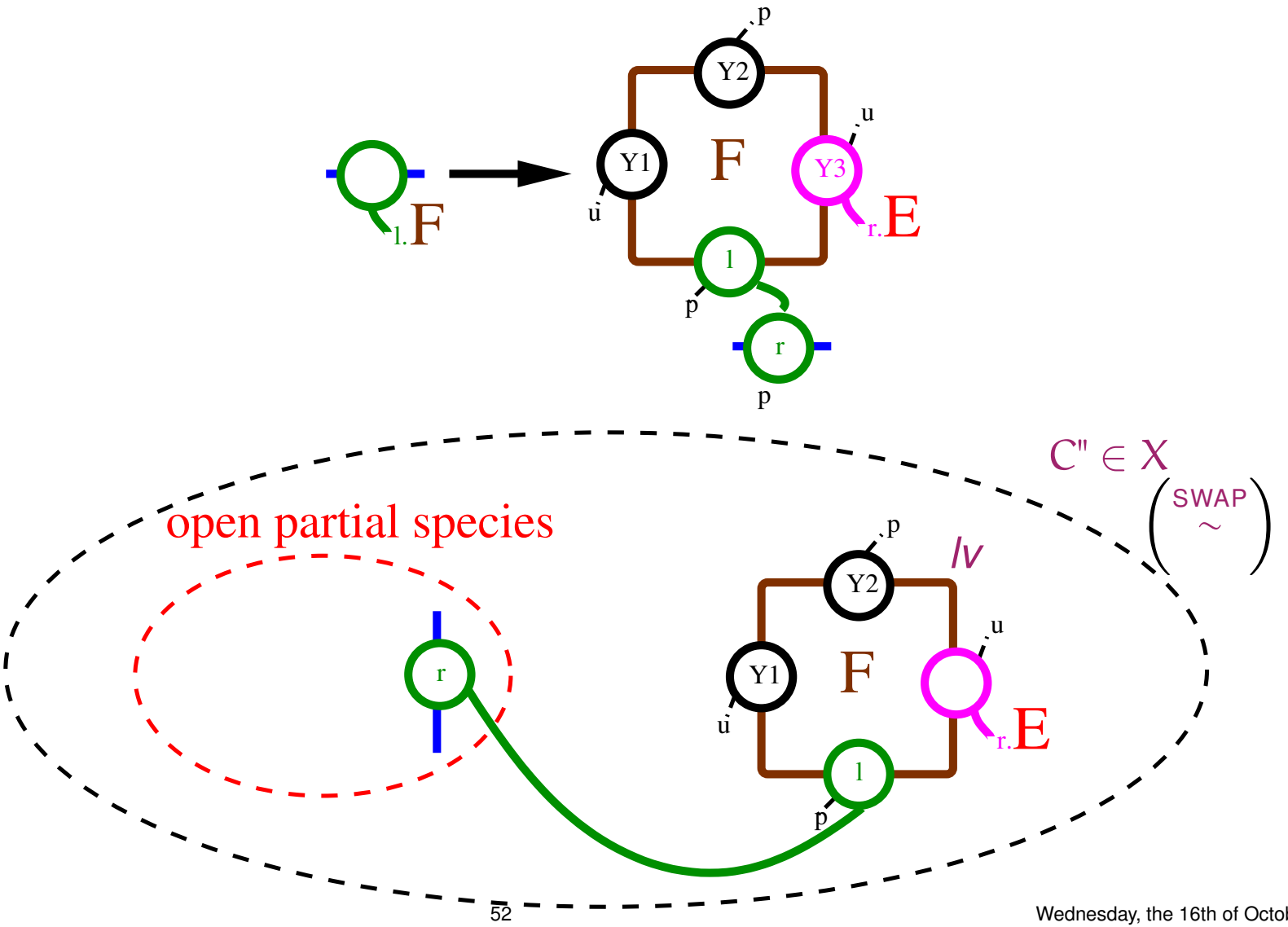


$C \in X$
(since $lv \in \alpha(X)$)

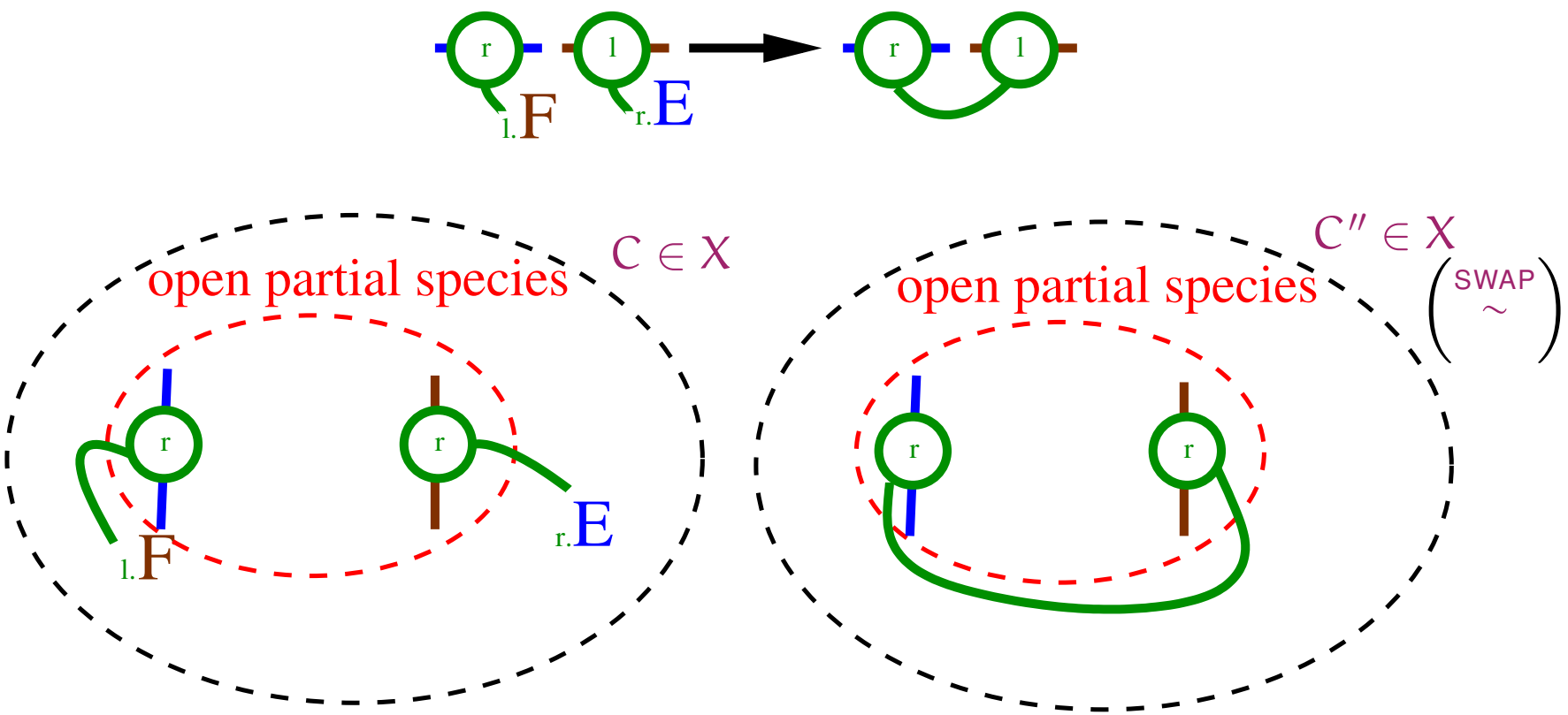
Unfolding a semi-link



Unfolding a semi-link



Binding two semi-links



Consequences

Let $Y \in \wp(\mathit{Local\_view})$ be a set of local views such that $\alpha(\gamma(Y)) = Y$.

1. Each open complex C built with the local views in Y is a sub-complex of a close complex C' in $\gamma(Y)$.

   (by replacing X by $\gamma(Y)$ in the previous proof)

2. When considering the rewrite system that computes $\gamma(Y)$, any partial rewriting sequence can be completed in a successful one.

   Thus:

   (a) $\gamma(Y)$ is finite if and only if the grammar has a finite set of prefixes (and the latter is decidable);

   (b) We have $\mathbb{F}^\sharp \circ \alpha = \alpha \circ \mathbb{F} \circ \gamma \circ \alpha$.

Overview

1. Introduction
2. Language: Kappa
3. Abstraction: Local views
4. Completeness: false positives?
5. Local fragment of Kappa
6. Decontextualization
7. Conclusion

Outline

We have proved that:

- if the set $Species_\omega$ of reachable chemical species is close with respect swapping $\overset{\text{SWAP}}{\sim}$,
- then the reachability analysis is exact (i.e. $Species_\omega = \gamma(lfp_{\alpha(Species_0)}\mathbb{F}^\sharp)$).

Now we give some sufficient conditions that ensure this property.

Sufficient conditions

Whenever the following assumptions:

1. initial agents are not bound;
2. rules are atomic;
3. rules are local:
   - only agents that interact are tested,
   - no cyclic patterns (neither in lhs, nor in rhs);
4. binding rules do not interfere i.e. if both:
   - A(a{m}[.],S),B(b{n}[.],T) $\rightarrow$ A(a{m}[1],S),B(b{n}[1],T)
   - and A(a{m'}[.],S'),B(b{n'}[.],T') $\rightarrow$ A(a{m'}[1],S'),B(b{n'}[1],T'),

   then:
   - A(a{m}[.],S),B(b{n'}[.],T') $\rightarrow$ A(a{m}[1],S),B(b{n'}[1],T');
5. chemical species in $\gamma(\alpha(\mathit{Species}_\omega))$ are acyclic,

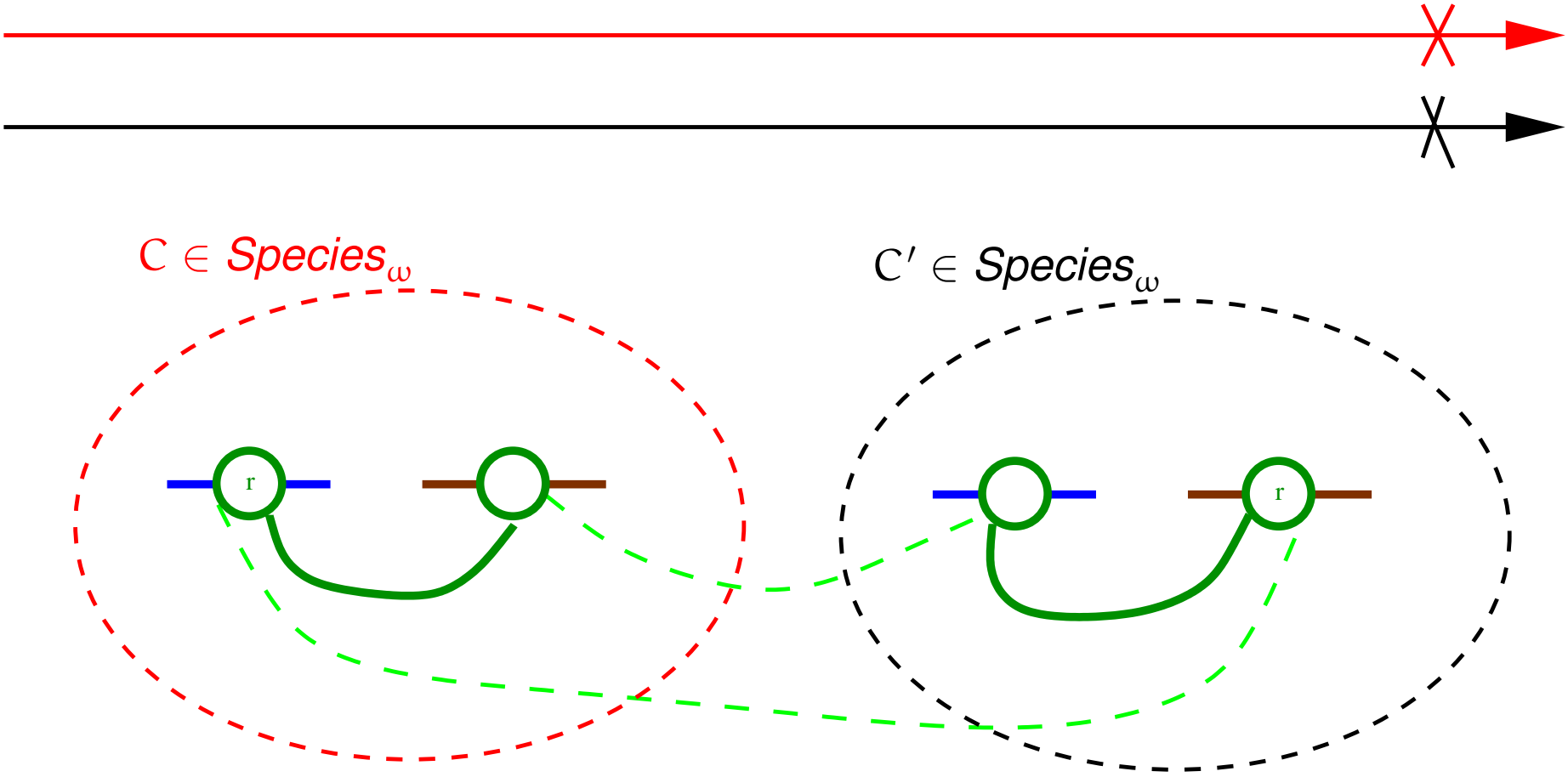
are satisfied, the set of reachable chemical species is local.

Proof outline

We sketch a proof in order to discover sufficient conditions that ensure this property:

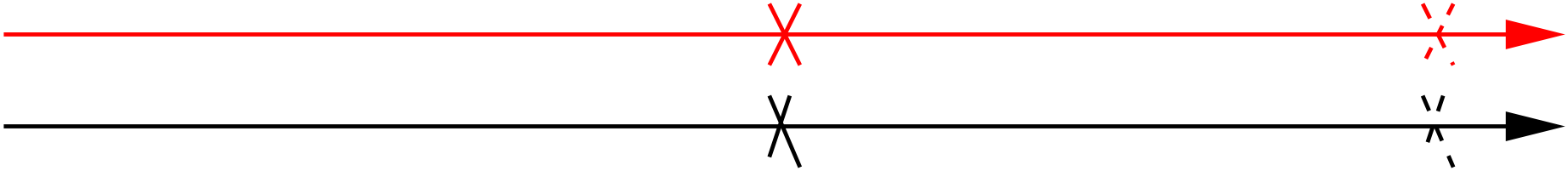
- We consider tuples of complexes in which the same kind of links occur twice.
- We want to swap these links.
- We introduce the history of their computation.
- There are several cases. . .

First case (I/V)

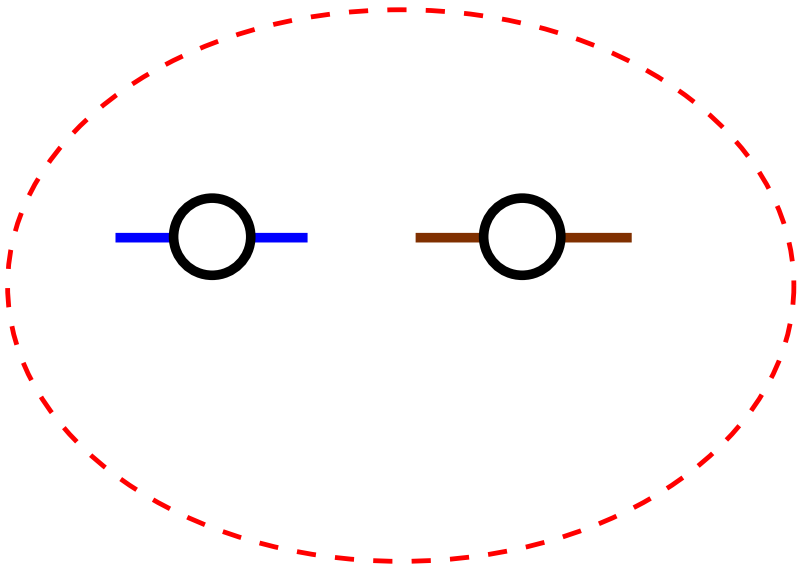


First case (II/V)

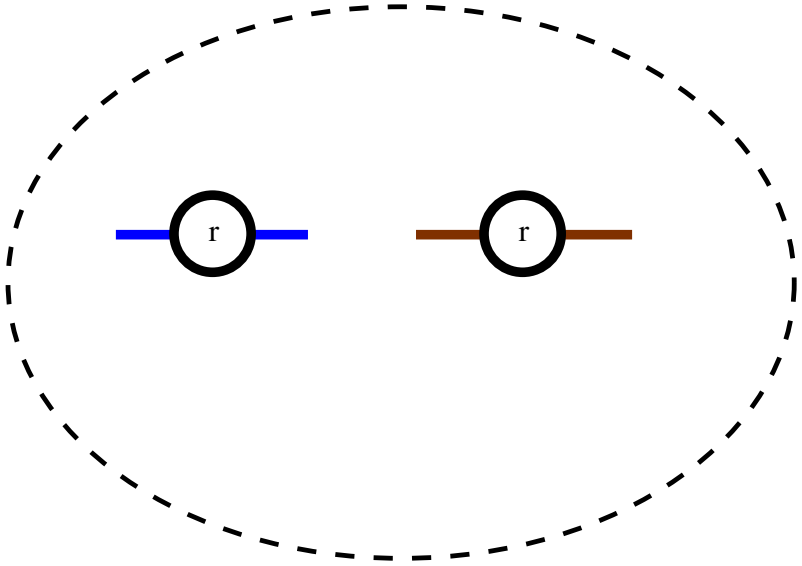
just before the links are made



$\mathcal{C} \in Species^*_\omega$

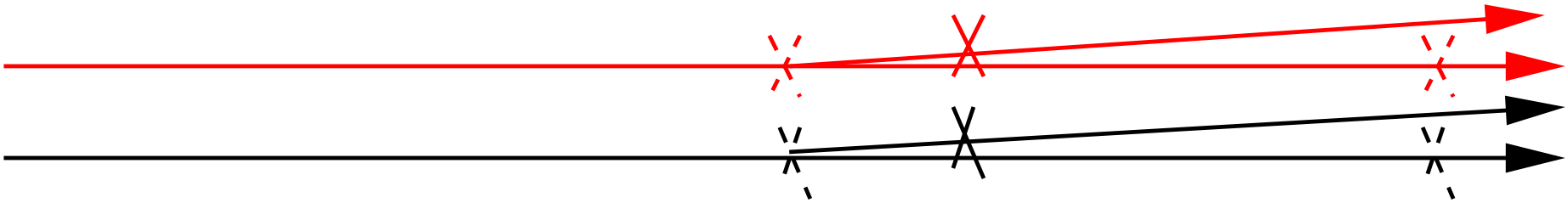


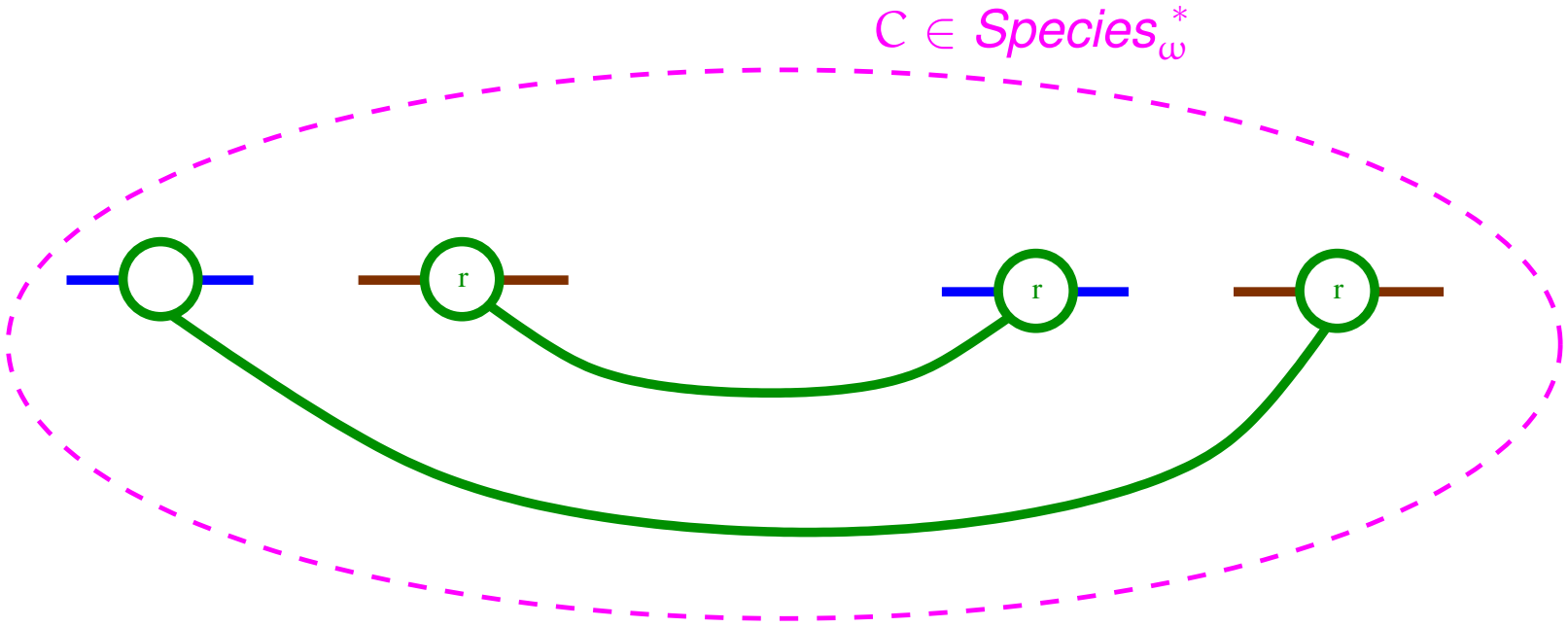
$\mathcal{C}' \in Species^*_\omega$



First case (III/V)

we suppose we can swap the links

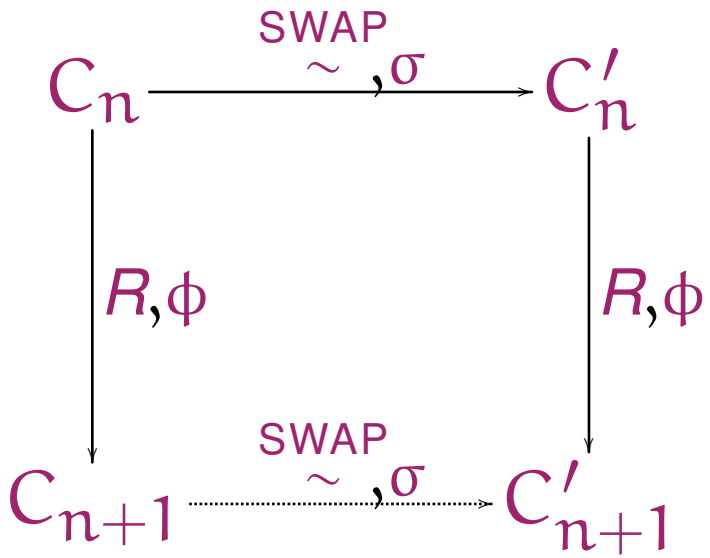




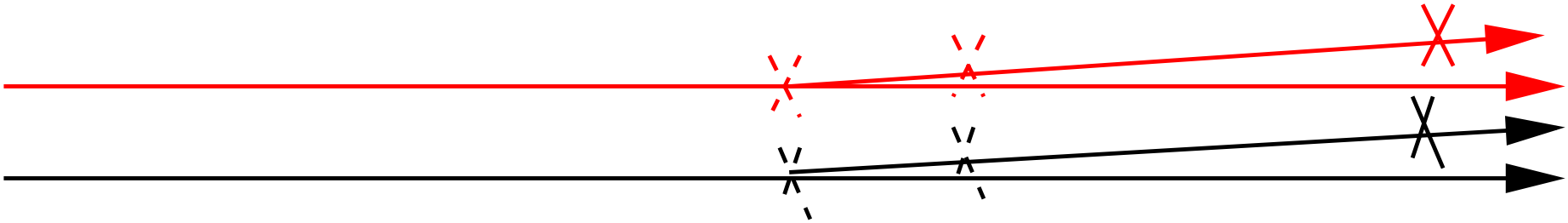
First case (IV/V)

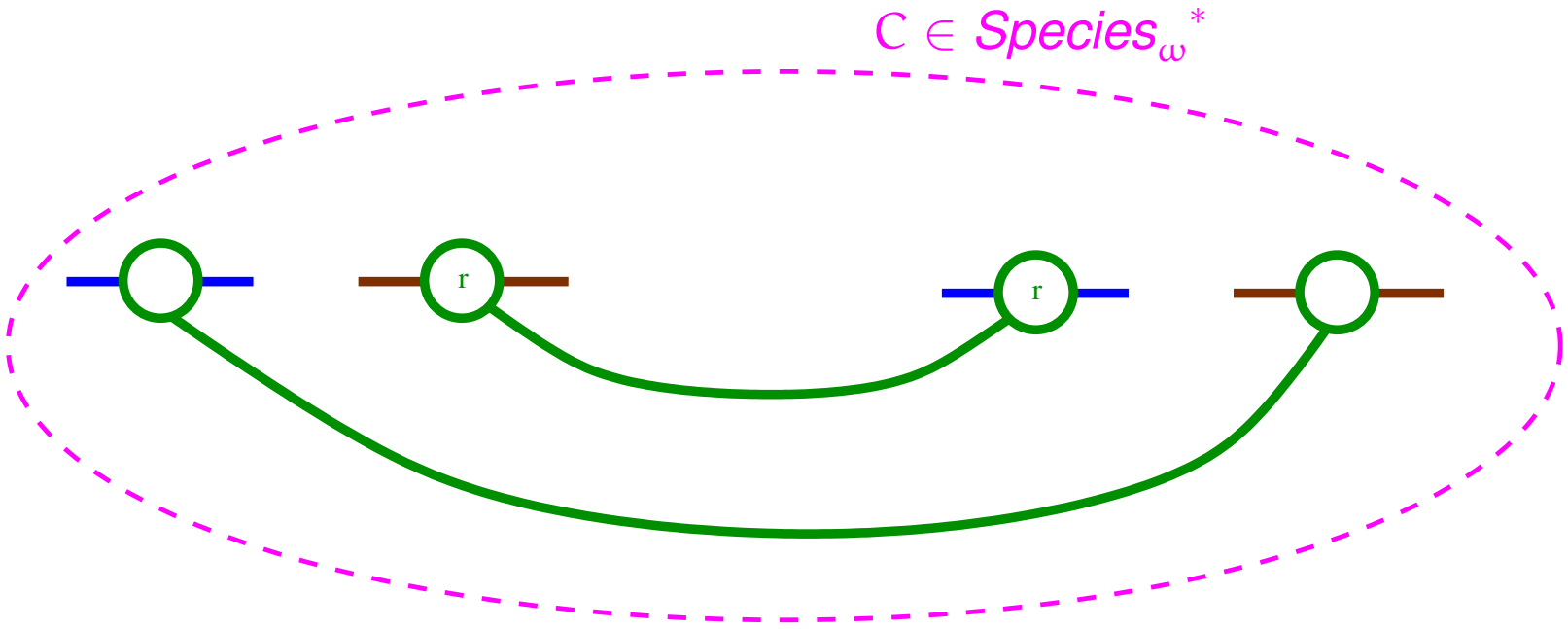
Then, we ensure that further computation steps:

- are always possible;
- have the same effect on local views;
- commute with the swapping relation $\overset{\text{SWAP}}{\sim}$.

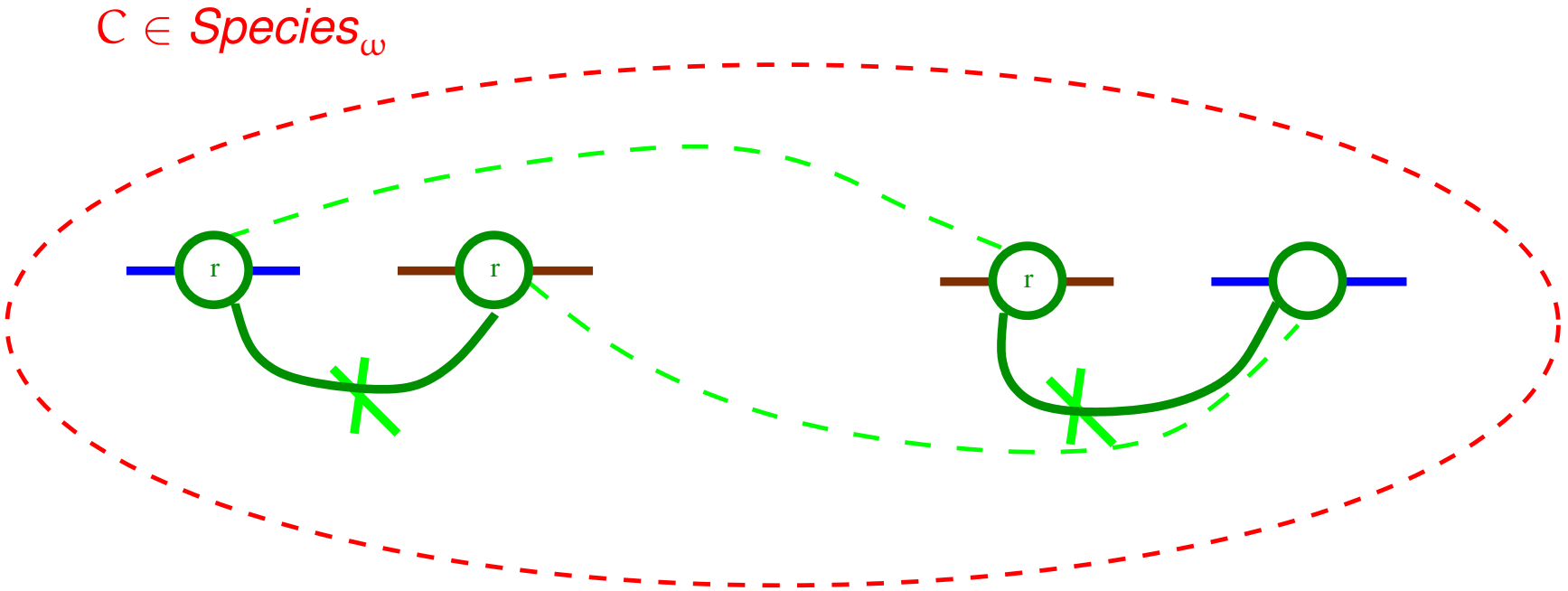


First case (V/V)



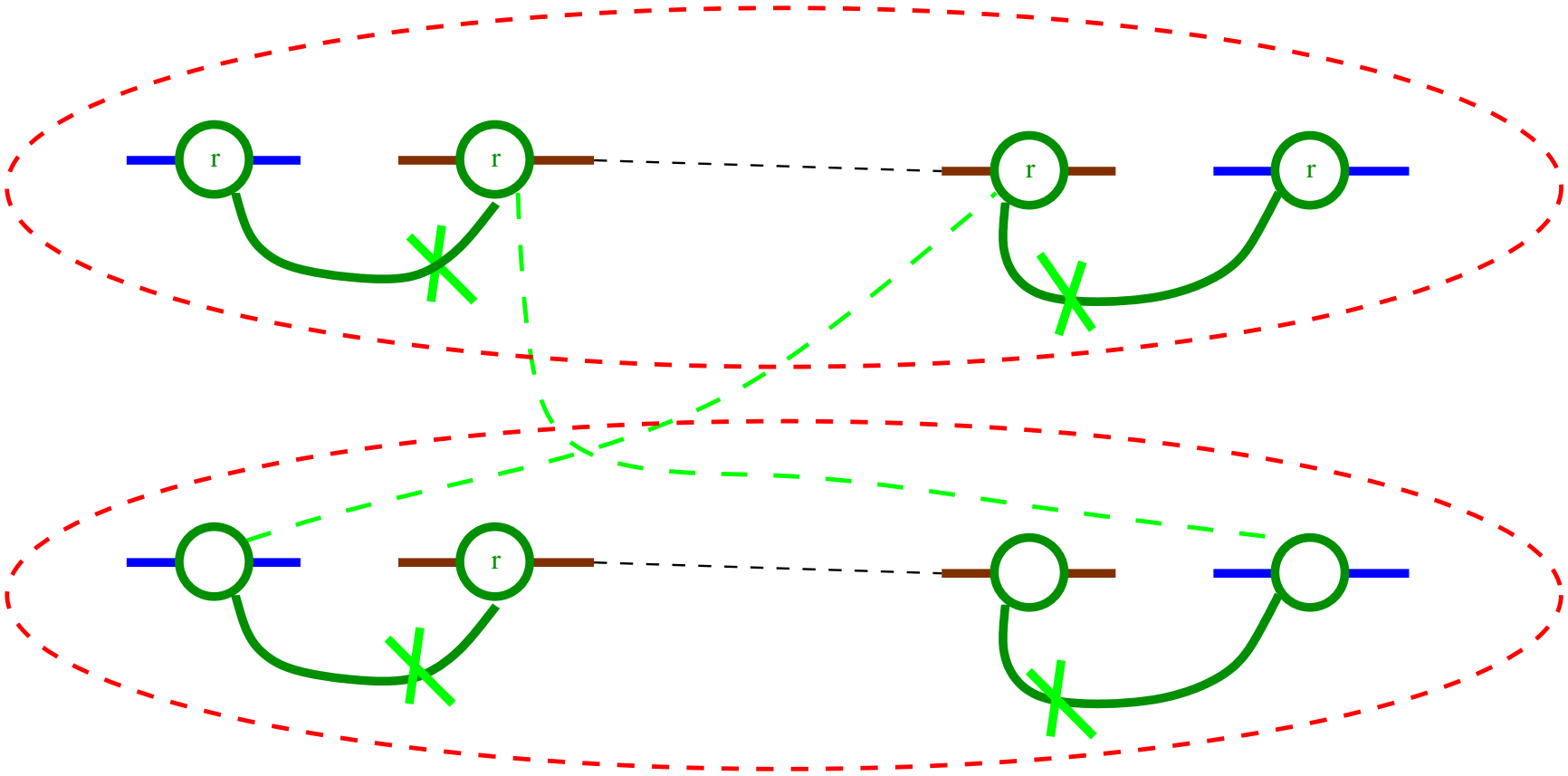


Second case (I/II)



we assume that the chemical species C is acyclic

Second case (II/II)



Sufficient conditions

Whenever the following assumptions:

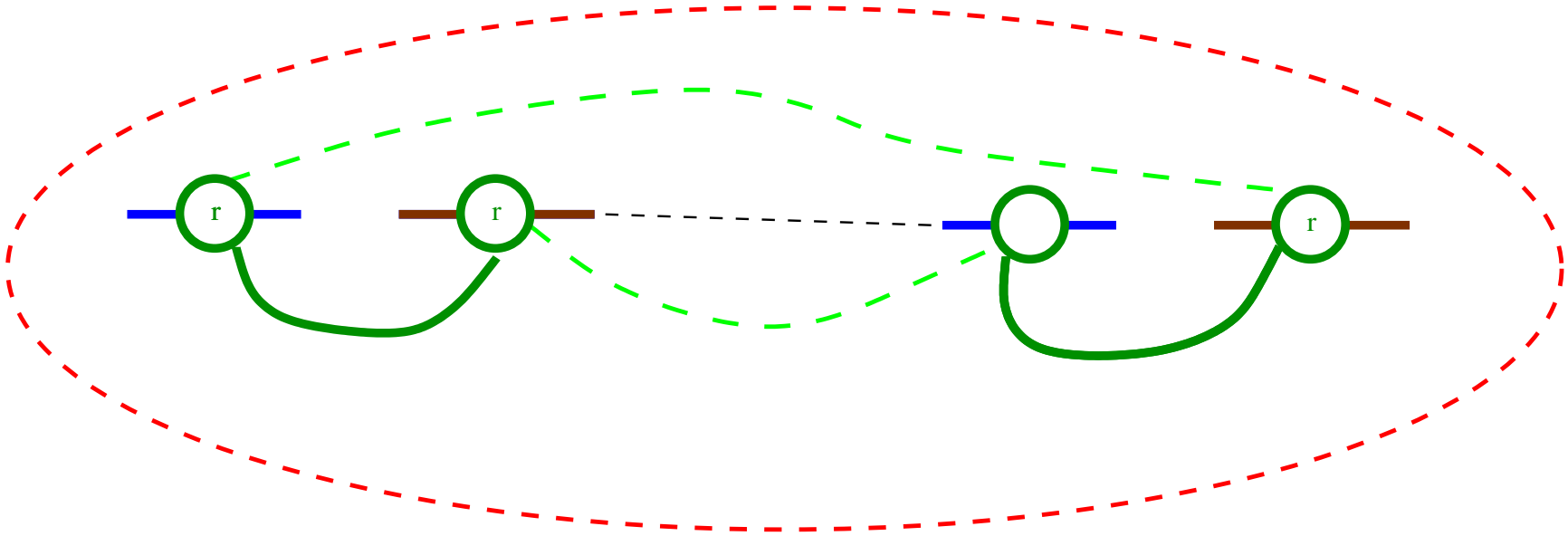
1. initial agents are not bound;
2. rules are atomic;
3. rules are local:
   - only agents that interact are tested,
   - no cyclic patterns (neither in lhs, nor in rhs);
4. binding rules do not interfere i.e. if both:
   - A(a{m}[.],S),B(b{n}[.],T) $\rightarrow$ A(a{m}[1],S),B(b{n}[1],T)
   - and A(a{m'}[.],S'),B(b{n'}[.],T') $\rightarrow$ A(a{m'}[1],S'),B(b{n'}[1],T'),

   then:
   - A(a{m}[.],S),B(b{n'}[.],T') $\rightarrow$ A(a{m}[1],S),B(b{n'}[1],T');
5. chemical species in $\gamma(\alpha(\mathit{Species}_\omega))$ are acyclic,

are satisfied, the set of reachable chemical species is local.

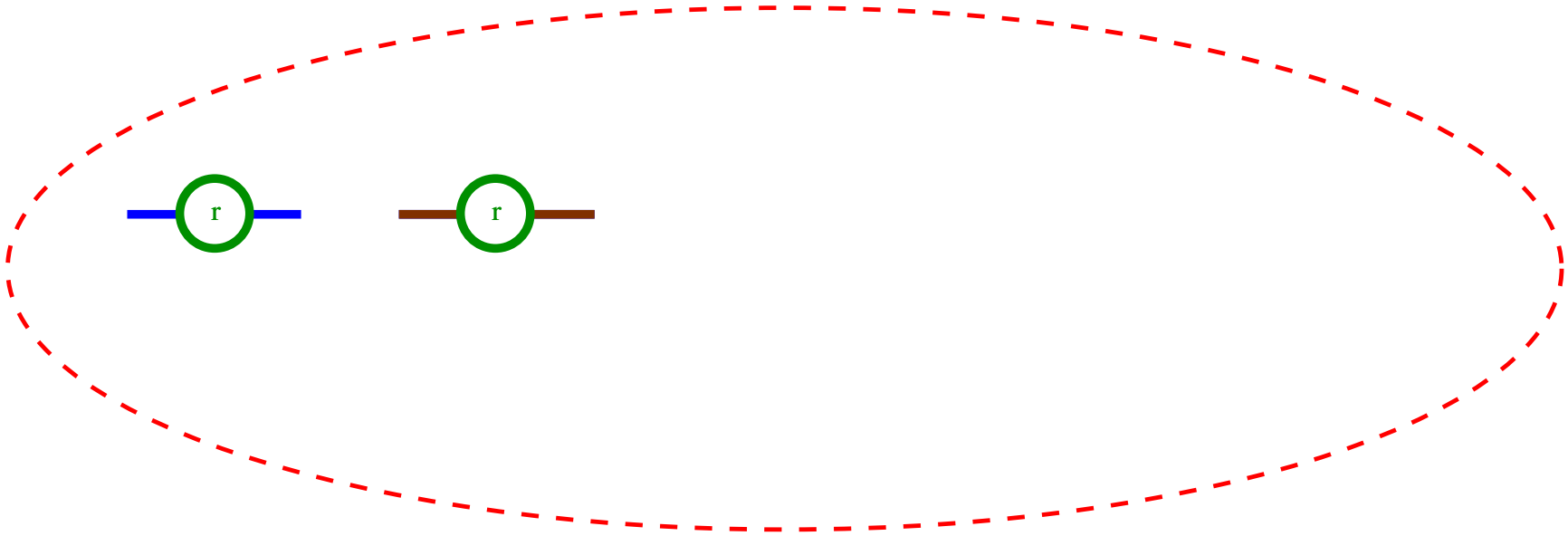
Third case (I/III)

$\mathcal{C} \in Species_\omega$

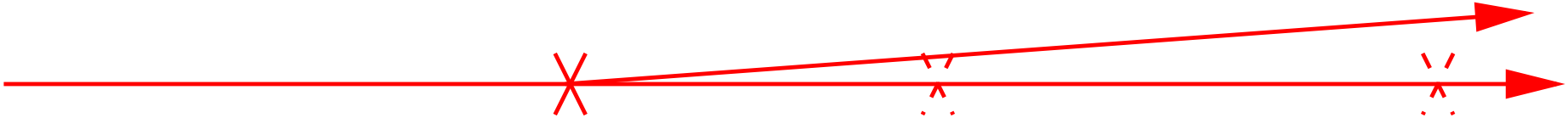


Third case (II/III)

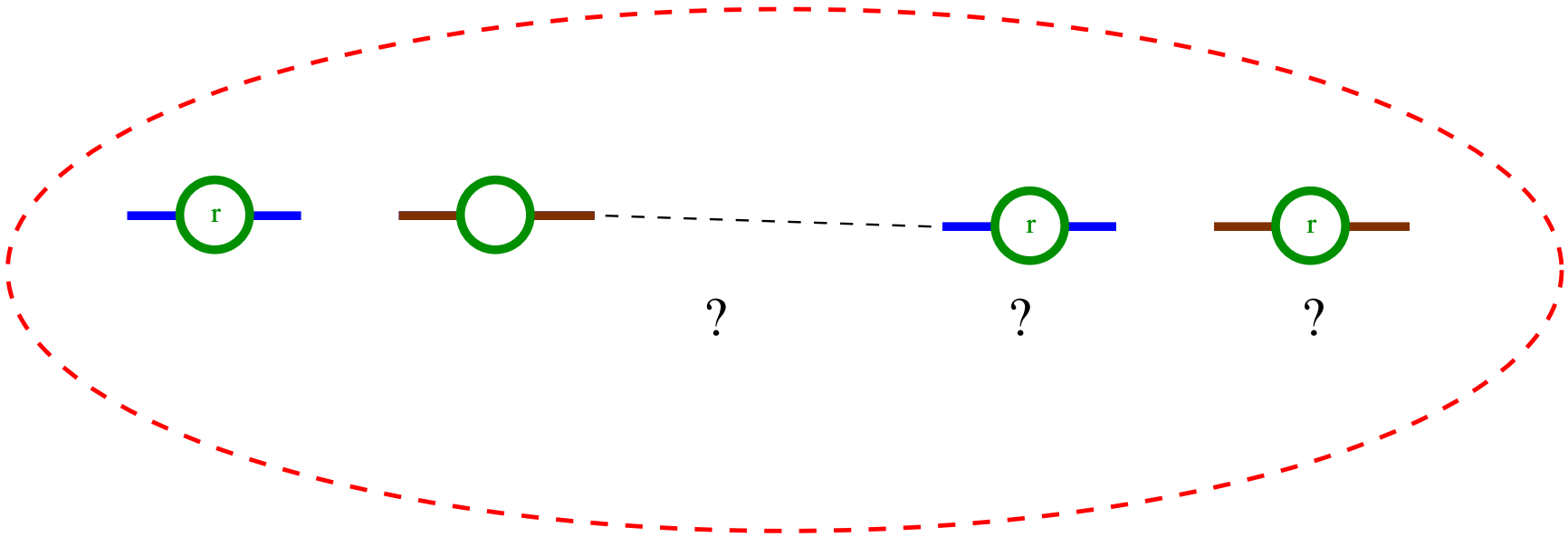
$\mathcal{C} \in Species^{*}_{\omega}$



Third case (II/III)



$\mathbb{C} \in Species^*_\omega$



Dangerous sites

A site is dangerous if it may occur in a cycle within a complex ($\in \gamma(\alpha(\mathit{Species}_\omega))$).

We would weaken the fifth requirement into:

- The binding state of a dangerous site is never tested, unless for binding or unbinding this site.
- When we bind dangerous sites, we only test that these sites are free.

Then, we prove that:

1. we can build any complex with free dangerous sites,
2. then, we can bind them as much as we like.

Non local systems

$$\mathit{Species}_0 \triangleq \mathrm{R(a\{u\}[.])}$$

$$\mathit{Rules} \triangleq \left\{ \begin{array}{lcl} \mathrm{R(a\{u\}[.])} & \leftrightarrow & \mathrm{R(a\{p\}[.])} \\ \mathrm{R(a\{u\}[.]),R(a\{u\}[.])} & \rightarrow & \mathrm{R(a\{u\}[1]),R(a\{u\}[1])} \\ \mathrm{R(a\{p\}[.]),R(a\{u\}[.])} & \rightarrow & \mathrm{R(a\{p\}[1]),R(a\{p\}[1])} \\ \mathrm{R(a\{p\}[.]),R(a\{p\}[.])} & \rightarrow & \mathrm{R(a\{p\}[1]),R(a\{p\}[1])} \end{array} \right.$$

$\mathrm{R(a\{u\}[1]),R(a\{u\}[1])} \in \mathit{Species}_\omega$

$\mathrm{R(a\{p\}[1]),R(a\{p\}[1])} \in \mathit{Species}_\omega$

But $\mathrm{R(a\{u\}[1]),R(a\{p\}[1])} \notin \mathit{Species}_\omega$.

Non local systems

$$\mathit{Species}_0 \triangleq \text{A(a\{u\}[.]),B(a\{u\}[.])}$$

$$\mathit{Rules} \triangleq \left\{ \begin{array}{l} \text{A(a\{u\}[.]),B(a\{u\}[.])} \longrightarrow \text{A(a\{u\}[1]),B(a\{u\}[1])} \\ \text{A(a\{u\}[1]),B(a\{u\}[1])} \longrightarrow \text{A(a\{p\}[1]),B(a\{u\}[1])} \\ \text{A(a\{u\}[1]),B(a\{u\}[1])} \longrightarrow \text{A(a\{u\}[1]),B(a\{p\}[1])} \end{array} \right\}$$

$\text{A(a\{u\}[1]),B(a\{p\}[1])} \in \mathit{Species}_\omega$

$\text{A(a\{p\}[1]),B(a\{u\}[1])} \in \mathit{Species}_\omega$

But $\text{A(a\{p\}[1]),B(a\{p\}[1])} \notin \mathit{Species}_\omega$.

Non local systems

$$Species_0 \triangleq A(a\{u\}[.])$$

$$Rules \triangleq \left\{ \begin{array}{l} A(a\{u\}[.]) \leftrightarrow A(a\{p\}[.]) \\ A(a\{u\}[.]),A(a\{p\}[.]) \rightarrow A(a\{u\}[1]),A(a\{p\}[1]) \end{array} \right\}$$

$A(a\{u\}[1]),A(a\{p\}[1]) \in Species_\omega$
But $A(a\{p\}[1]),A(a\{p\}[1]) \notin Species_\omega$.

Non local systems

$$\mathit{Species}_0 \triangleq \text{R(a[.],b[.])}$$
$$\mathit{Rules} \triangleq \{\ \text{R(a[.],b[.]),R(a[.])} \rightarrow \text{R(a[.],b[1]),R(a[1])}\}$$

R(a[.],b[2]),R(a[2],b[1]),R(a[1],b[.]) $\in \mathit{Species}_\omega$
But R(a[1],b[1]) $\notin \mathit{Species}_\omega$.

Overview

1. Introduction
2. Language: Kappa
3. Abstraction: Local views
4. Completeness: false positives?
5. Local fragment of Kappa
6. Decontextualization
7. Conclusion

Outline

- we have a syntactic criterion in order to ensure that the set of reachable chemical species of a kappa system is local ;
- we now design program transformations to help systems satisfying this criterion ;
  1. decontextualization
     - is fully automatic;
     - preserves the transition system;
     - simplifies rules thanks to reachability analysis.
  2. conjugation
     - manual;
     - preserves the set of reachable chemical species;
     - uses backtrack to add new rules.

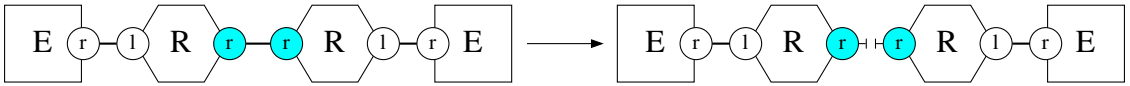
Questions about sets of patterns

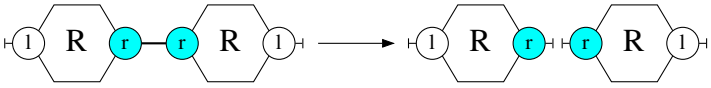
Many features that could help enhancing the confidence in models, such as:

- knowing under which context a given interaction may not be applied?
- knowing under which common context a given interaction may be applied twice (by two distinct rules)?
- knowing under which context one application of a given rule may trigger the application of another given rule?

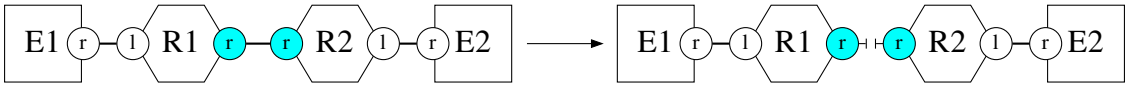
come down to the resolution of equations over sets of patterns.

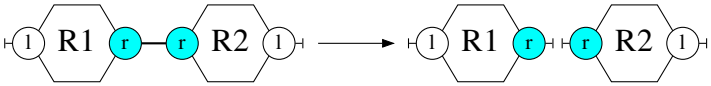
Are these two sets of rules equivalent?



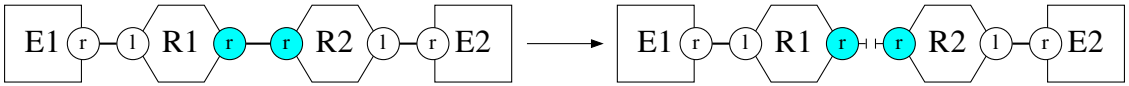


Identifying agents





Encoding in first order logic

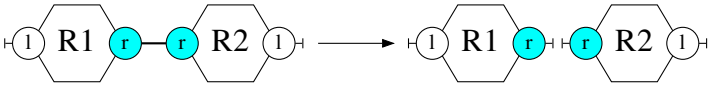


$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R \wedge \textsc{Type}(E1) = E \wedge \textsc{Type}(E2) = E$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{Link}((E1, r), (R1, l)) = .T. \wedge \textsc{Link}((E2, r), (R2, l)) = .T.$

$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R \wedge \textsc{Type}(E1) = E$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{Link}((E1, r), (R1, l)) = .T. \wedge \textsc{State}(R2, l) = \dashv$



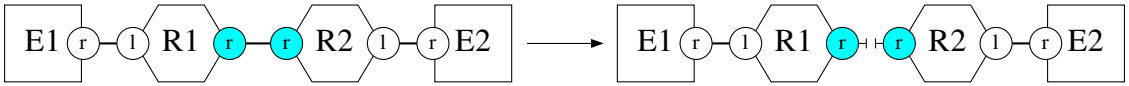
$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{State}(R1, l) = \dashv \wedge \textsc{State}(R2, l) = \dashv$

$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T.$

Reasonning up to iso

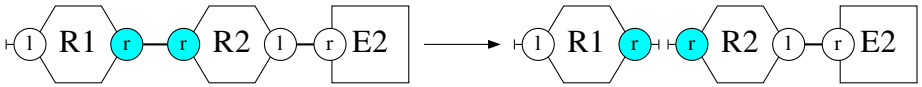


$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R \wedge \textsc{Type}(E1) = E \wedge \textsc{Type}(E2) = E$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{Link}((E1, r), (R1, l)) = .T. \wedge \textsc{Link}((E2, r), (R2, l)) = .T.$

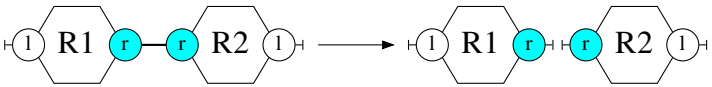
$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R \wedge \textsc{Type}(E1) = E$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{Link}((E1, r), (R1, l)) = .T. \wedge \textsc{State}(R2, l) = \dashv$



$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R \wedge \textsc{Type}(E2) = E$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{Link}((E2, r), (R2, l)) = .T. \wedge \textsc{State}(R1, l) = \dashv$



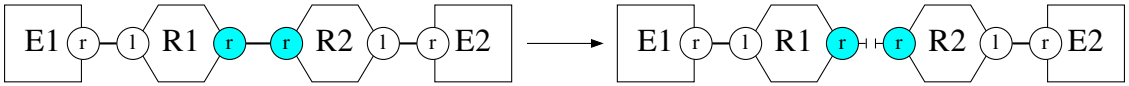
$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{State}(R1, l) = \dashv \wedge \textsc{State}(R2, l) = \dashv$

$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T.$

Using structural and model invariants

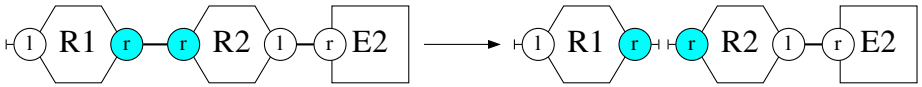


$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R \wedge \textsc{Type}(E1) = E \wedge \textsc{Type}(E2) = E$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{Link}((E1, r), (R1, l)) = .T. \wedge \textsc{Link}((E2, r), (R2, l)) = .T.$

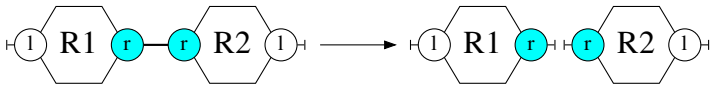
$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R \wedge \textsc{Type}(E1) = E$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{Link}((E1, r), (R1, l)) = .T. \wedge \textsc{State}(R2, l) = \dashv$



$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R \wedge \textsc{Type}(E2) = E$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{Link}((E2, r), (R2, l)) = .T. \wedge \textsc{State}(R1, l) = \dashv$



$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T. \wedge \textsc{State}(R1, l) = \dashv \wedge \textsc{State}(R2, l) = \dashv$

$\textsc{Type}(R1) = R \wedge \textsc{Type}(R2) = R$

$\wedge\ \textsc{Link}((R1, r), (R2, r)) = .T.$

$\wedge\ \rho_{Kappa} \wedge \rho_{Model}$

Issues

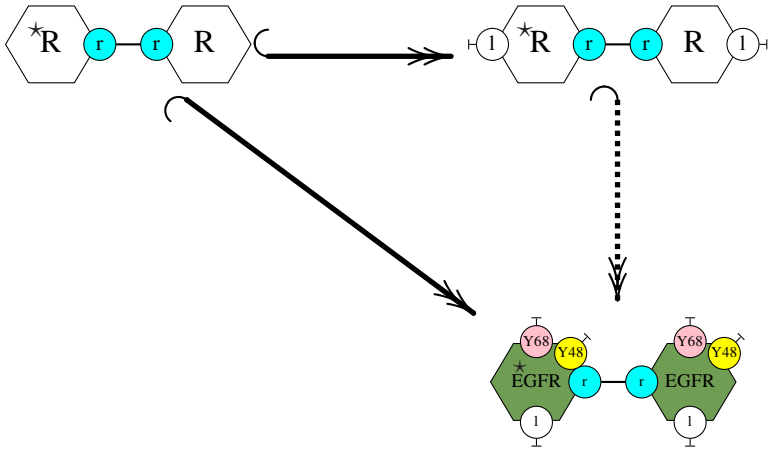
- How to align patterns?
- How to deal with agent identifiers?
- How to account with the constraints coming from Kappa and those coming from the models?

Rooted pattern

- An root is a pattern of interest.
- An anchorage is an embedding from the root to a rooted graph.

An anchorage $\mathcal{A}$ denotes the set $[\mathcal{A}]$ of the embeddings from the root to fully specified patterns (mixtures) that can be factored by this anchorage.

For instance :



Such sets of embeddings may be combined by using set operations.

Attributes about rooted mixtures

$$\textsc{Type}(id) = \begin{cases} \mathcal{A} & \text{Agent } id \text{ is of type } \mathcal{A}. \\ \bot & \text{There is no agent } id. \end{cases}$$

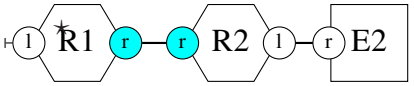
$$\textsc{State}(id, s) = \begin{cases} \mathcal{A}'.s' & \text{The site } s \text{ of agent } id \text{ is bound to} \\ & \text{the site } s\text{' of an agent of type } \mathcal{A}'. \\ \dashv & \text{The site } s \text{ of agent } id \text{ is free.} \\ \bot & \text{Either there is no agent } id, \\ & \text{or this agent has no site } s. \end{cases}$$

$$\textsc{Link}((id, s), (id', s')) = \begin{cases} .\text{T}. & \text{The site } s \text{ of the agent } id \text{ is bound to} \\ & \text{the site } s\text{' of the agent } id'. \\ .\text{F}. & \text{In any other case} \end{cases}$$

Agent identifiers

- In the root, one agent per connected component is identified.
- Agent identifiers are relative paths starting from this agent.

For instance, in the following rooted pattern:



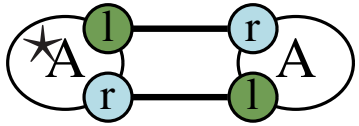
agent identifiers are defined as follows:

- $\mathrm{R1} := \varepsilon;$
- $\mathrm{R2} := \varepsilon.\mathrm{r} - \mathrm{r};$
- $\mathrm{E2} := \varepsilon.\mathrm{r} - \mathrm{r}.\mathrm{l} - \mathrm{r}.$

Cyclic patterns

We stop every relative path after reaching an agent already in that path.
We use the attribute ALIAS to encode pairs of paths that denote the same agents.

For instance the following rooted pattern:



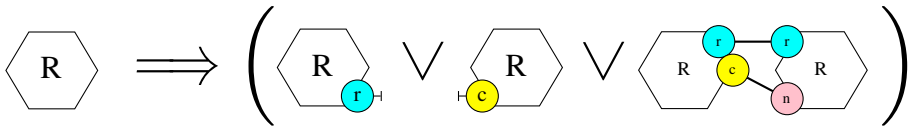
is encoded by the following formula:

$$\textsc{Type}(\varepsilon) = A \wedge \textsc{Type}(\varepsilon.r - l) = A \wedge \textsc{Type}(\varepsilon.l - r) = A \wedge \textsc{Alias}(\varepsilon, \varepsilon.r - l) = .F. \wedge \textsc{Alias}(\varepsilon.r - l, \varepsilon.l - r) = .T.$$

$$\textsc{Alias}(id, id') = \begin{cases} .T. & \text{Agents } id \text{ and } id' \text{ are the same} \\ .F. & \text{There is no agent } id, \text{ there is no agent } id', \text{ or they are distinc} \end{cases}$$

Saturation lemmas

1. Structural lemmas:
   - Relative paths also describe bonds;
   - The prefix of a valid relative path is valid as well;
   - Bonds are symmetric;
   - Aliases propagate;
   - Aliases propagate the other attributes;
   - Information about bonds can be weakened;
2. Model invariants:
   - Implications captured by static analysis, such as:



Issue: These lemmas are universaly quantified over relative paths.

A hierarchy of interpretations

- $[\![\phi]\!]$ is the set of the valuations that satisfy ϕ.
- $[\![\phi]\!]_{\mathcal{K}}$ is the set of the valuations that satisfy ϕ and that are the translation of a (rooted) mixture.
- $[\![\phi]\!]_{\mathcal{K},\mathcal{M}}$ is the set of the valuations that satisfy ϕ and that are the translation of a reachable (rooted) mixture.

$$[\![\phi]\!] = [\![\phi']\!] \Longrightarrow [\![\phi]\!]_{\mathcal{K}} = [\![\phi']\!]_{\mathcal{K}}$$
$$[\![\phi]\!]_{\mathcal{K}} = [\![\phi']\!]_{\mathcal{K}} \Longrightarrow [\![\phi]\!]_{\mathcal{K},\mathcal{M}} = [\![\phi']\!]_{\mathcal{K},\mathcal{M}}$$
Deciding whether $[\![\phi]\!]_{\mathcal{K},\mathcal{M}} = [\![\phi']\!]_{\mathcal{K},\mathcal{M}}$ is undecidable.

Full abstraction

$$[\![\Psi_P]\!]_{\mathcal{K},\mathcal{M}} = [\![\Psi_{P'}]\!]_{\mathcal{K},\mathcal{M}} \iff [P] = [P']$$

where Ψ_P is the translation of the rooted pattern P in first order logic.

Coherence

For any saturation lemma ρ,

$$[\![\phi]\!]_{\mathcal{K},\mathcal{M}} = [\![\phi \wedge \rho]\!]_{\mathcal{K},\mathcal{M}}$$

Reminders:

- $[.]$ maps each rooted pattern to the underlying set of rooted reachable mixtures;
- $[\![.]\!]_{\mathcal{K},\mathcal{M}}$ maps every formula to the set of the valuations that satisfy this formula and that are the translation of a reachable mixture.

Boolean algebra

We define $\mathcal{Id}(\phi)$ as the set of agent identifiers (access paths) that occur in ϕ.

- $\uparrow(\phi) = (\rho_\phi, \phi \wedge \rho_\phi)$

  where $\rho_\phi = \bigwedge\{\rho \mid \rho \text{ saturation lemma s.t. } \mathcal{Id}(\rho) \subseteq \mathcal{Id}(\phi)\}$;

- $\neg(\rho, \phi) = (\rho, (\neg\ \phi) \wedge \rho)$;

- $(\rho_1, \phi_1) \wedge (\rho_2, \phi_2) = (\rho_1 \wedge \rho_2, (\phi_1 \wedge \rho_2) \wedge (\phi_2 \wedge \rho_1))$;

- $(\rho_1, \phi_1) \vee (\rho_2, \phi_2) = (\rho_1 \wedge \rho_2, (\phi_1 \wedge \rho_2) \vee (\phi_2 \wedge \rho_1))$.

Saturation: Soundness

For a given finite conjunction ρ of saturation lemmas.

$$\Phi(\psi_1 \wedge \rho, \ldots, \psi_n \wedge \rho) \wedge \rho \equiv \Phi(\psi_1, \ldots, \psi_n) \wedge \rho$$

Proof:

- $$\begin{aligned} ((\psi_1 \wedge \rho) \wedge (\psi_2 \wedge \rho)) \wedge \rho &\equiv (((\psi_1 \wedge \psi_2) \wedge \rho) \wedge \rho) \wedge \rho \\ &\equiv (\psi_1 \wedge \psi_2) \wedge \rho \end{aligned}$$

- $$\begin{aligned} ((\psi_1 \wedge \rho) \vee (\psi_2 \wedge \rho)) \wedge \rho &\equiv ((\psi_1 \vee \psi_2) \wedge \rho) \wedge \rho \\ &\equiv (\psi_1 \vee \psi_2) \wedge \rho \end{aligned}$$

- $$\begin{aligned} (\neg\, (\psi_1 \wedge \rho)) \wedge \rho &\equiv ((\neg\, \psi_1) \vee (\neg\, \rho)) \wedge \rho \\ &\equiv ((\neg\, \psi_1) \wedge \rho) \vee ((\neg\, \rho) \wedge \rho) \\ &\equiv ((\neg\, \psi_1) \wedge \rho) \end{aligned}$$

Saturation: Parsimony

For a given finite conjunction ρ of saturation lemmas.

$$\Phi(\psi_1 \wedge \rho, \ldots, \psi_n \wedge \rho) \equiv \Phi(\psi_1 \wedge \rho, \ldots, \psi_n \wedge \rho) \wedge \rho$$
where Φ contains no negation.

Proof:

- $((\psi_1 \wedge \rho) \wedge (\psi_2 \wedge \rho)) \wedge \rho \equiv (\psi_1 \wedge \psi_2) \wedge \rho$
- $((\psi_1 \wedge \rho) \vee (\psi_2 \wedge \rho)) \wedge \rho \equiv (\psi_1 \vee \psi_2) \wedge \rho$
- $$\begin{aligned}(\neg\,(\psi_1 \wedge \rho)) \wedge \rho &\equiv ((\neg\,\psi_1) \vee (\neg\,\rho)) \wedge \rho \\ &\equiv ((\neg\,\psi_1) \wedge \rho) \vee ((\neg\,\rho) \wedge \rho) \\ &\equiv (\neg\,\psi_1) \wedge \rho\end{aligned}$$

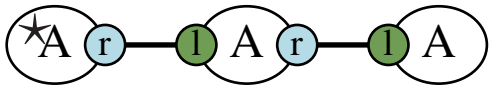
Back to Kappa: conjunction of atoms (unfolded form)

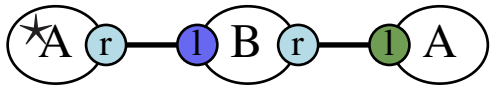
$$\textsf{TYPE}(\varepsilon) = A \wedge \textsf{TYPE}(\varepsilon.\mathrm{l} - \mathrm{r}.\mathrm{l} - \mathrm{r}) = A \wedge \textsf{ALIAS}(\varepsilon, \varepsilon.\mathrm{l} - \mathrm{r}.\mathrm{l} - \mathrm{r}) = .\mathsf{T}.$$

Back to Kappa: conjunction of atoms (unfolded form)

$$\textsc{Type}(\varepsilon) = A \wedge \textsc{Type}(\varepsilon.l - r.l - r) = A \wedge \textsc{Alias}(\varepsilon, \varepsilon.l - r.l - r) = .T.$$

The type of the agent at the relative address $\varepsilon.l - r$ is underspecified.
We have to consider every solution according to the signature of the model:

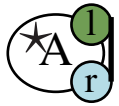


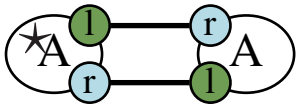


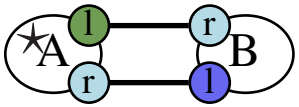
Back to Kappa: conjunction of atoms (Kappa patterns)

$$\textsc{Type}(\varepsilon) = A \wedge \textsc{Type}(\varepsilon.l - r.l - r) = A \wedge \textsc{Alias}(\varepsilon, \varepsilon.l - r.l - r) = .T.$$

Then, we fold agents according to the information about aliases.
When underspecified, we consider every potential case.







Back to Kappa: general case

Logical formulae are encoded into BDDs (MVBDUs).

PROS:

- It offers canonic form (with respect to $[\![.]\!]$);
- Every formula is a disjunction of conjunctions;
- It offers a parsimonious representation
  (whenever an attribute is useless, it is discarded automatically);
- It uses optimal sharing and memoisation.

CONS:

- Formulae (and their traduction back into Kappa) depend on an a priori order over the attributes;
- Worst case computation may be exponential in theory.
  Computation time is difficult to predict in practice.

Conjugation

If a rule R' is equivalent to a rule in the transitive closure of the system.
Then it may be included in the system without modifying reachable states.
To remove the context C of a rule, we try to apply it for another context C' by:

1. removing the context C' (backtrack) ;
2. building the context C ;
3. applying the initial rule ;
4. removing the context C (backtrack) ;
5. building the context C'.

This is proved manually.

Overview

1. Introduction
2. Language: Kappa
3. Abstraction: Local views
4. Completeness: false positives?
5. Local fragment of Kappa
6. Decontextualization
7. Conclusion

Conclusion

- A scalable static analysis to abstract the reachable chemical species.
- A class of models for which the abstraction is complete.
- Many applications:
  - idiomatic description of reachable chemical species;
  - dead rule detection;
  - rule decontextualization;
  - computer-driven kinetic refinement.
- It can also help simulation algorithms:
  - wake up/inhibition map (agent-based simulation);
  - flat rule system generation (for bounded set of chemical species);
  - on the fly flat rule generation (for large/unbounded set)