THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'École normale supérieure

Formal and exact reduction for differential models of signalling pathways in rule-based languages

Ecole doctorale n°386

École doctorale de Sciences Mathématiques de Paris Centre

Spécialité Informatique

COMPOSITION DU JURY :

Mme. YANG, Jean
Carnegie Mellon University, Rapporteur

M. GIACOBAZZI, Roberto
Università di Verona, Rapporteur

M. FAGES, François
INRIA, Membre du jury

M. FERET, Jérôme
INRIA & ÉNS, Membre du jury

M. HARMER, Russ
CNRS & ÉNS Lyon, Membre du jury

M. THIEFFRY, Denis
ÉNS, Membre du jury

Soutenue par
Ferdinanda CAMPORESI
le 23 janvier 2017

Dirigée par **Jérôme FERET**
Radhia COUSOT

ENS
ECOLE NORMALE SUPÉRIEURE

PSL
RESEARCH UNIVERSITY PARIS

Résumé

Le comportement d'une cellule dépend de sa capacité à recevoir, propager et intégrer des signaux, constituant ainsi des voies de signalisations. Les protéines s'associent entre elles sur des sites de liaisons, puis modifient la structure spatiale des protéines voisines, ce qui a pour effet de cacher ou de découvrir leurs autres sites de liaisons, et donc d'empêcher ou de faciliter d'autres interactions.

En raison du grand nombre de différents complexes biomoléculaires, nous ne pouvons pas écrire ou générer les systèmes différentiels sous-jacents. Les langages de réécritures de graphes à sites offrent un bon moyen de décrire ces systèmes complexes. Néanmoins la complexité combinatoire resurgit lorsque l'on cherche à calculer de manière effective ce comportement. Ceci justifie l'utilisation d'abstractions.

Nous proposons deux méthodes pour réduire la taille des modèles de voies de signalisation, décrits en Kappa. Ces méthodes utilisent respectivement la présence de symétries parmi certains sites et le fait que certaines corrélations entre l'état de différentes parties des complexes biomoléculaires n'ont pas d'impact sur la dynamique du système global. Des sites qui ont la même capacité d'interaction sont liés par une relation de symétrie. Nous montrons que cette relation induit une bisimulation qui peut être utilisée pour réduire la taille du modèle initial. L'analyse du flot d'information détecte les parties du système qui influencent le comportement de chaque site. Ceci nous autorise à couper les espèces moléculaires en petits morceaux pour écrire un nouveau système. Enfin, nous montrons comment raffiner cette analyse pour tenir compte d'information contextuelle.

Les deux méthodes peuvent être combinées. La solution analytique du modèle réduit est la projection exacte de la solution originelle. Le calcul du modèle réduit se fait au niveau des règles, en évitant l'exécution du modèle initial.

Abstract

The behaviour of a cell is driven by its capability to receive, propagate and communicate signals. Proteins can bind together on some binding sites. Post-translational modifications can reveal or hide some sites, so new interactions can be allowed or existing ones can be inhibited.

Due to the huge number of different bio-molecular complexes, we can no longer derive or integrate ODE models. A compact way to describe these systems is supplied by rule-based languages. However combinatorial complexity raises again when one attempt to describe formally the behaviour of the models. This motivates the use of abstractions.

We propose two methods to reduce the size of the models, that exploit respectively the presence of symmetries between sites and the lack of correlation between different parts of the system. The symmetries relates pairs of sites having the same capability of interactions. We show that this relation induces a bisimulation which can be used to reduce the size of the original model. The information flow analysis detects, for each site, which parts of the system influence its behaviour. This allows us to cut the molecular species in smaller pieces and to write a new system. Moreover we show how this analysis can be tuned with respect to a context.

Both approaches can be combined. The analytical solution of the reduced model is the exact projection of the original one. The computation of the reduced model is performed at the level of rules, without the need of executing the original model.

Man könnte den ganzen Sinn des Buches etwa in die Worte fassen: Was sich überhaupt sagen lässt, lässt sich klar sagen; und wovon man nicht reden kann, darüber muss man schweigen.

Ludwig Wittgenstein, TLP

Grazie Jérôme.

Contents

Chapter 1

Introduction

1.1 Context and motivation

The behaviour of a cell is driven by its capability to receive, propagate and communicate signals. These communications are carried out by interactions between proteins. Proteins can bind together on some binding sites and modify their spatial structure. This can reveal or hide some binding sites, so new communications can be allowed or existing ones can be inhibited. *Signalling pathways* gather several interactions that describe how a specific signal induces the activation of a given protein. Since each protein has several interaction sites, these systems suffer from combinatorial complexity. This has several impacts and makes these systems very hard to be represented and simulated.

We are interested in mechanistic models that reflect in a direct way the underlying mechanisms in the systems we study. A common way to represent the knowledge we have on this kind of systems is by the means of pathway maps. A pathway map is a graph where each node represents a protein, a bio-molecular complex (that we call a species) or a pattern, and each arc represents some kind of interaction described in some informal language. Pathway maps supply a good way to organise knowledge but no formal semantics is associated with them. Another approach is based on the use of systems of ordinary differential equations (ODEs). Generally each variable denotes the concentration of some species and the system describes the way the variables evolve in time. Since the number of species is huge it is impossible to write the system and a lot of simplifications are made. Typical solutions are to neglect some species or to quotient some others. The way these simplifications are made is often not so clear and once that the model is written we cannot assign it a meaning. So it is not even possible to update

them. Furthermore, a model should not be considered as a single entity. We are indeed more interested in families of models corresponding to some variations of common *meta-models*. Then, it is important to ease the navigation between models in order to test various sets of assumptions about mechanistic details [2, 24, 35]. There are two main reasons for which ODEs models can hardly achieve this goal. The first one is that they do not describe the biochemical structure of components, the second one is due to their size.

Rule-based languages, as Kappa, supply a powerful way to describe formally the knowledge we have on biological systems and allow to model their behaviour. Any kind of modification is a completely local process so it becomes easy to change the behaviour of the system we want to study. The language is close to the representation of the knowledge and for each rule we are able to give a description in natural language. At the same time each rule is a formal object to which we can assign a semantics. Between all the possible choices we focus on differential semantics, where each connected component in the rules is associated to an equation describing the evolution in time of the species concentration related to it. By means of rule-based languages we can write compact and handy models but, sadly, the combinatorial complexity problem appears again if we want to run a simulation of the system. There is need for abstraction.

Abstract Interpretation has been introduced forty years ago by Patrick and Radhia Cousot. This is a theory of sound approximation of mathematical structures. In particular it allows to formally relate semantics models given at different precision levels. Abstract Interpretation has been being used successfully in several areas of computer science [17], ranging among semantics, verification and proof, model-checking, program transformation and optimisation, typing, etc. One of its main application is static analysis of programs. The behaviour of a program is described as the least fixpoint of a monotonic operator $\mathbb{F}$ over the elements of a concrete domain $\mathcal{D}$. The abstraction changes the descriptional level of the program behaviour. This change is performed by introducing a new abstract domain $\mathcal{D}^\sharp$ with an abstract operator $\mathbb{F}^\sharp$ defined on it. This can be formalised in several way. The most common approach is based on Galois connection.

In our work, we use Abstract Interpretation for several purposes.

1. We provide a generic framework to describe exact model reductions of ODEs semantics. The solution of the reduced system is the exact projection of the solution of the initial system and the abstraction is formalised as a change of variables. Abstract Interpretation provides convenient ways to define abstractions compositionally. The principle of concepts separation allows for the design of very complex abstrac-

tions as a composition of simpler ones. Following this guideline, we propose a generic way to combine model reductions.

2. We define an abstraction based on symmetries between sites. Two sites are defined as symmetric when they play the same role in the semantics. For example, if we have k phosphorylation sites that are symmetric and $(s_1, \ldots, s_k)$ is the tuple describing their state, we can replace the tuple with an integer $n \in [1, k]$ specifying the number of them that is phosphorylated and neglecting their position. We show that an equivalence relation based on symmetries induces a bisimulation and we use this bisimulation to reduce the dimension of the ODEs semantics.

3. We define an abstraction that tracks the information flow between different regions of chemical species. Our method is based on the idea that, as described by rules, the transformations are purely local. So instead of considering a full species we can consider just a part of that. From this information, we can detect statically when the correlation between the state of two sites matters for the semantics. Abstracting away useless correlations allows us to cut species into partial species, that we call *fragments*. The set of fragments defines our abstract domain, from we which we can derive a reduced systems.

1.2 Case studies

We begin with examining several case studies in order to facilitate intuitions about the model reductions we aim to achieve. For the sake of simplicity, we use reaction networks. In the following chapters, we will show how to compute these model reductions automatically from a high level specification in Kappa, without ever generating the underlying reaction networks.

1.2.1 A protein with independent sites

Let us start with an example from [25].

Model

We consider a simple set of reactions.

$$\begin{array}{llcll} (1) & A, B & \leftrightarrow & AB & k_A, k'_A \\ (2) & A, BC & \leftrightarrow & ABC & k_A, k'_A \\ (3) & B, C & \leftrightarrow & BC & k_C, k'_C \\ (4) & AB, C & \leftrightarrow & ABC & k_C, k'_C \end{array}$$

This system describes an agent B that can bind simultaneously and reversibly to two other agents, A and C.

The first reaction states that an agent B can bind to an agent A at a rate constant k_A (direct arrow). This reaction is reversible as illustrated by the reverse arrow and the agents can dissociate at a rate constant k'_A. The second reaction states that a complex BC can bind to an agent A at a rate constant k_A and also that this reaction is reversible with a rate constant equal to k'_A. The third reaction states that an agent B can bind to an agent C at a rate constant k_C and also that this reaction is reversible with a rate constant equal to k'_C. The fourth reaction states that a complex AB can bind to an agent C at a rate constant k_C and also that this reaction is reversible with a rate constant equal to k'_C.

Totally, six species are possible: A, B, C, AB, BC and ABC.

The behaviour of this system of reactions can be defined by the means of the *mass action principle*, stating that the reactants and the products of each reaction are consumed and produced according to the rate of this reaction. The rate of a reaction is defined as the product between the rate constant and the concentration of the reactants of this reaction.

For example, the rate of the second reaction, where A binds B, is $k_A[A][B]$ ($[A]$ is the traditional notation for the concentration of A, that is the amount of As per unit volume).

So, we can write the associated system of differential equations describing the time derivative for each of the six possible species.

$$
\begin{aligned}
[A]' &= k'_A([AB] + [ABC]) - k_A[A]([B] + [BC]) \\
[B]' &= k'_A[AB] + k'_C[BC] - [B](k_A[A] + k_C[C]) \\
[C]' &= k'_C([BC] + [ABC]) - k_C[C]([B] + [AB]) \\
[AB]' &= k_A[A][B] + k'_C[ABC] - [AB](k'_A + k_C[C]) \\
[BC]' &= k_C[B][C] + k'_A[ABC] - [BC](k'_C + k_A[A]) \\
[ABC]' &= k_A[A][BC] + k_C[AB][C] - [ABC](k'_A + k'_C).
\end{aligned}
$$

We can observe that the differential system is autonomous, meaning that the time derivatives of variables do not explicitly depend on time.

Once the values of initial state and the rate constants have been fixed, it is possible to integrate the system and to calculate the time course of each variable.

Model reduction

By observing the set of reactions, we can see that the rate constant of the reactions:

$$A, B \to AB$$

and:

$$A, BC \to ABC$$

is equal to k_A in both cases. This can be interpreted as the fact that for a given B, its capability to bind with an A does not depend on the fact that it is already bound to a C or not.

Analoguously, the rate constant of the reactions:

$$AB \to A, B$$

and:

$$ABC \to A, BC$$

is k'_A in both cases. This means that the capability of an A to dissociate from a B does not depend on the fact that B is linked to a C or not.

The same way the rate constant of the reactions:

$$B, C \to BC$$

and:

$$AB, C \to ABC$$

is the same (k_C), and also the one of the reactions:

$$BC \to B, C$$

and:

$$ABC \to AB, C$$

that is equal to k'_C.

So we can imagine that it is possible to split our system into two indepent ones, where we cut B in two parts and for each part we retain only the information about its bond to A (or to C) and we forget all the information about the fact that is bond to C (respectively to A) or not.

The two subsystems are shown in Figure 1.1.

For the moment, let us consider just the subsystem on the left (where we forget the information about C). We introduce two new variables: $[AB?]$ to denote the total concentration of AB and ABC, $[B?]$ to denote the total concentration of B and BC. More specifically:

$$[AB?] := [AB] + [ABC]$$
$$[B?] := [B] + [BC].$$

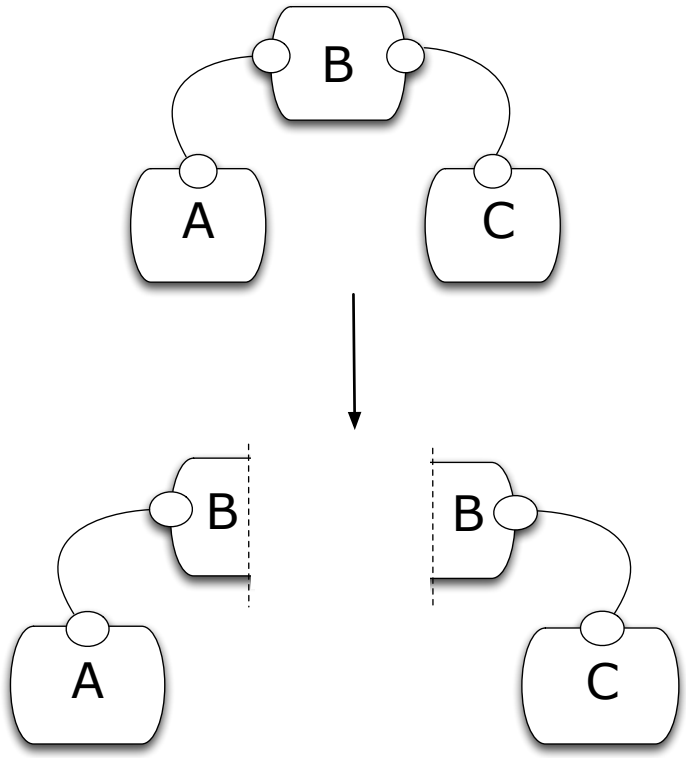


Figure 1.1: The original system is split into two independents subsystems.

So in our subsystem we will have three different "species": A, $B?$ and $AB?$.

We can write the following differential system to describe the time derivative of our new species:

$$\begin{aligned}
&[A]' = k'_A[AB?] - k_A[A][B?] \\
&[AB?]' = [AB]' + [ABC]' = k_A[A][B?] - k'_A[AB?] \\
&[B?]' = [B]' + [BC]' = k'_A[AB?] - k_A[A][B?].
\end{aligned}$$

In the same way it is possible to write a differential system of equations also for the subsystem showed on the right of Figure 1.1. There we will introduce the two following new variables:

$$\begin{aligned}
&[?BC] := [BC] + [ABC] \\
&[?B] := [AB] + [B]
\end{aligned}$$

and obtain the following system of differential equations:

$$\begin{aligned}
&[C]' = k'_C[?BC] - k_C[C][?B] \\
&[?BC]' = [BC]' + [ABC]' = k_C[C][?B] - k'_C[?BC] \\
&[?B]' = [B]' + [AB]' = k'_C[?BC] - k_C[C][?B].
\end{aligned}$$

Dependence index

The example shows us that it is possible to exploit structural features of a given differential system to infer specific linear combinations of its variables with a self-consistent dynamics.

Now we wonder if, once that we have split the original system into two separated ones, we are able to recover the original information. To this purpose we introduce a new variable:

$$[?B?] = [?B] + [?BC] = [B?] + [AB?]$$

to denote the total concentration of B.

So:

1. $\dfrac{[ABC]}{[?B?]}$ gives the proportion of Bs with both an A and a C attached,
2. $\dfrac{[AB?]}{[?B?]}$ gives the proportion of Bs with an A attached,
3. $\dfrac{[?BC]}{[?B?]}$ gives the proportion of Bs with a C attached.

The fact that a protein B is bound to A is independent from the fact that it is bound to C if, and only if, the first expression is the product of the two following ones. That is to say:

$$\frac{[ABC]}{[?B?]} = \frac{[AB?]}{[?B?]} \cdot \frac{[?BC]}{[?B?]}.$$

This is equivalent to:

$$[ABC] \cdot [?B?] = [AB?] \cdot [?BC]$$

and to:

$$\frac{[ABC]}{[?BC]} = \frac{[AB?]}{[?B?]}.$$

That is to say that the probability of B to be linked with A knowing that it is bound to C is equal to the probability of B to be linked with A knowing nothing about C.

By this equation we can define a dependence index χ as follows:

$$\chi := [ABC] \cdot [?B?] - [AB?] \cdot [?BC],$$

and after a computation we obtain:

$$\frac{d\chi}{dt} = -\chi(k_A[A] + k_C[C] + k'_A + k'_C).$$

So the property:

$$[ABC] = \frac{[AB?] \cdot [?BC]}{[?B?]}$$

is an invariant of the system, if it hold at time t, it holds at any time $t' \geqslant 0$.

1.2.2 A protein with information flow

Now we present a second example. This model is a particular instance of a generic example that can be found in [6]. This generic example describes a protein in which several sites, called master sites, control the behaviour of some other sites, that we call subordinate. In such a case, we say that there is a flow of information from the master sites to the subordinate sites. In this section, we focus to the case when there is only one master site and two subordinate sites, but our framework can deal with any number of master sites and any number of subordinate sites.

Model

Now we assume that we have a protein with three sites. We denote a protein as a tuple of symbols among p and u. p denotes a phosporylated site, whereas u denotes an unphosporylated one. So, for instance, a protein that has all its three sites phosporylated is denoted as (p, p, p).

Our system is fully described by the following set of reactions.

$$\begin{array}{llcll}
(1) & (u,u,u) & \rightarrow & (u,p,u) & k_c \\
(2) & (u,p,u) & \rightarrow & (p,p,u) & k_l \\
(3) & (u,p,p) & \rightarrow & (p,p,p) & k_l \\
(4) & (u,p,u) & \rightarrow & (u,p,p) & k_r \\
(5) & (p,p,u) & \rightarrow & (p,p,p) & k_r
\end{array}$$

The set describes a system where, in a protein that has all the sites not activated, the first one to be activated is the central one (as asserted by the first reaction). Once that a protein has the central site activated it can activate also the left and the right ones.

The first reaction states that a protein with no site activated can activate the central one with a rate constant equal to k_c. The second rule states that a protein which has the central site activated can activate the left one with

a rate constant equal to k_l. The third rule states that a protein which has the central and the right site activated can activate the left one with a rate constant equal to k_l. The fourth rule states that a protein which has the central site activated can activate the right one with a rate constant equal to k_r. The fifth rule states that a protein which has the central and the left site activated can activate the right one with a rate constant equal to k_r.

Model reduction

We can notice that the rate constants of the reactions:

$$(u, p, u) \to (p, p, u)$$

and:

$$(u, p, p) \to (p, p, p)$$

are both equal to k_l.

This suggests us that the capability of the left site to be activated does not depend on the fact that the right site is activated or not.

The same way, the rate constants of the reactions:

$$(u, p, u) \to (u, p, p)$$

and:

$$(p, p, u) \to (p, p, p)$$

are both equal to k_r.

This suggests us that the capability of the right site to be activated does not depend either on the fact that the left site is activated or not. As before, we can imagine to split our system into two smaller ones as showed by Figure 1.2.

We introduce two new variables for the subsystem on the left:

$$\begin{aligned}[(u, p, ?)] &:= [(u, p, u)] + [(u, p, p)], \\ [(p, p, ?)] &:= [(p, p, u)] + [(p, p, p)]\end{aligned}$$

and two new variables for the subsystem on the right:

$$[(?, p, u)] := [(u, p, u)] + [(p, p, u)], [(?, p, p)] := [(u, p, p)] + [(p, p, p)].$$

We can compute the dynamic of the new subsystem and, for the one on the left, we obtain:

$$\begin{aligned}[(u, u, u)]' &= -k_c[(u, u, u)], \\ [(u, p, ?)]' &= -k_l[(u, p, ?)] + k_c[(u, u, u)], \\ [(p, p, ?)]' &= k_l[(u, p, ?)].\end{aligned}$$

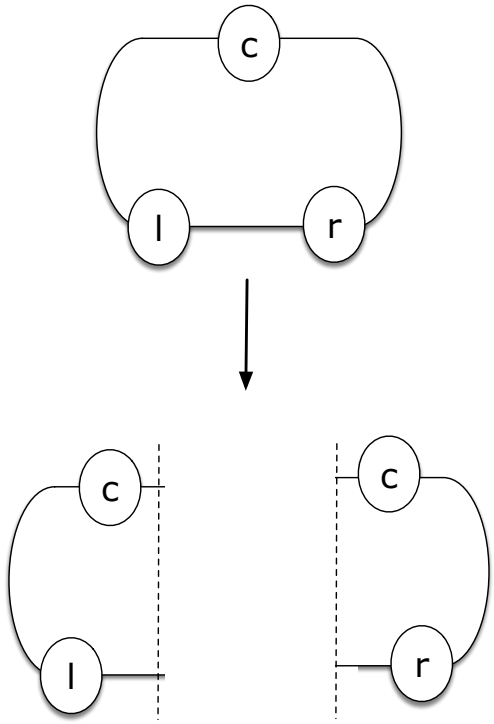


Figure 1.2: The original system is split into two independents subsystems.

Whereas for the one on the right we get:

$$\begin{aligned}
&[(u,u,u)]' = -k_c[(u,u,u)] \\
&[(?,p,u)]' = -k_r[(?,p,u)] + k_c[(u,u,u)] \\
&[(?,p,p)]' = k_r[(?,p,u)].
\end{aligned}$$

Dependence index for the second example

As in the previous example, we wonder if, once that we have split the original system into two separate ones, we are able to recover the original information. Again we introduce a new variable:

$$[(?,p,?)] := [(?,p,u)] + [(?,p,p)] = [(u,p,?)] + [(p,p,?)].$$

$(?,p,?)$ denotes a protein that has the central site phosphorylated, but of which we ignore the status of the left and right sites.

The states of the left site and the right one will be independent if, and only if, the probability that the right site is activated knowing that the left

site is activated will be equal to the probability that the right site is activated knowing nothing about the left site, that is:

$$\frac{[(p,p,p)]}{[(p,p,?)]} = \frac{[(?,p,p)]}{[(?,p,?)]}.$$

By this equation we can define a dependence index χ as follows:

$$\chi := [(p,p,p)] \cdot [(?,p,?)] - [(?,p,p)] \cdot [(p,p,?)]$$

and after a computation we get:

$$\frac{d\chi}{dt} = -\chi \cdot (k_l + k_r) + k_c \cdot [(p,p,p)] \cdot [u,u,u].$$

So the property $\chi = 0$ is not an invariant of the system. This suggest that we can split the system into two subsystems, but we cannot recombine the system without errors, in a time independent way.

1.2.3 A protein with symmetries

The first two examples we have seen so far exploit the same kind of idea to reduce a model. They are based on the fact that whenever the correlations, if any, between different parts of the system have an influence on the behaviour of no other site, it is possible to abstract these correlations away. Which means intentionally that we can split proteins into independent subparts.

Another kind of phenomenon that may appear, and that we can use to reduce a system, is the one of "redundancy". We can have a system with several parts that represent the same kind of information and capabilities, so we can collapse all these different parts in only one and taking in account the cardinality.

Model

For a better understanding, let us have a look to a concrete example. The system showed in Figure 1.3 describes the behaviour of a protein with three sites (x_1, x_2, y). More specifically:

- reactions $1-4$ describe the phosporylation of the site x_1;
- reactions $5-8$ describe the phosporylation of the site x_2;
- reactions $9-12$ describe the phosporylation of the site y.

(1) $(u,u,u) \to (p,u,u)$ k_m	(5) $(u,u,u) \to (u,p,u)$ k_m	(9) $(u,u,u) \to (u,u,p)$ k_0
(2) $(u,u,p) \to (p,u,p)$ k_m	(6) $(u,u,p) \to (u,p,p)$ k_m	(10) $(u,p,u) \to (u,p,p)$ k_1
(3) $(u,p,u) \to (p,p,u)$ k_m	(7) $(p,u,u) \to (p,p,u)$ k_m	(11) $(p,u,u) \to (p,u,p)$ k_1
(4) $(u,p,p) \to (p,p,p)$ k_m	(8) $(p,u,p) \to (p,p,p)$ k_m	(12) $(p,p,u) \to (p,p,p)$ k_2

(a) Phosphorylation of x_1. (b) Phosphorylation of x_2. (c) Phosphorylation of y.

Figure 1.3: A model with symmetric sites.

Model reduction

As we can observe, the activation rate constant of x_1 and x_2 is the same: k_m. Moreover, the activation rate constant of y changes according to the number of the sites x_i that are phosporylated (it is equal to k_0 when both x_1 and x_2 are unphosporylated, to k_1 when just one of them is phosporylated, and to k_2 when they are both phosporylated).

This peculiarity can be used to reduce our model. The idea is to forget about which sites x_i are phosporylated and to remember just how many phosporylated sites we have. When two or more sites have the same capability of interaction, as x_1 and x_2 in our case, we define them symmetric. This equivalence relation over sites can be lifted to an equivalence relation $\sim$ over proteins, so we say that two proteins are $\sim$-equivalent if the number of phosphorylated sites among x_1 and x_2 is the same for both proteins.

In our case we have the following equivalence classes over states:

$$\begin{array}{lccc} (a) & (u,p,u) & \sim & (p,u,u) \\ (b) & (u,p,p) & \sim & (p,u,p), \end{array}$$

and the original system is equivalent to the following reduced one:

$$\begin{array}{llcll} (1) & (u,u,u) & \to & (p,u,u) & 2k_m \\ (2) & (u,u,p) & \to & (p,u,p) & 2k_m \\ (3) & (p,u,u) & \to & (p,p,u) & k_m \\ (4) & (p,u,p) & \to & (p,p,p) & k_m \\ (5) & (u,u,u) & \to & (u,u,p) & k_0 \\ (6) & (p,u,u) & \to & (p,u,p) & k_1 \\ (7) & (p,p,u) & \to & (p,p,p) & k_2. \end{array}$$

Where:

- reaction 1 describes the activation of one site x_i ($i \in \{1,2\}$) in a protein where all sites are unphosporylated;

- reaction 2 describes the activation of one site x_i in a protein where all the sites x_i are unphosporylated the site y is phosporylated;
- reaction 3 describes the activation of a site x_i in a protein where the other site x_j is phosporylated and the site y is unphosporylated;
- reaction 4 describes the activation of a site x_i in a protein where the other site x_j is phosporylated and the site y is phosporylated too;
- reaction 5 describes the activation of the site y where no site x_i is phosporylated;
- reaction 6 describes the activation of the site y in a protein that has one site x_i phosporylated;
- reaction 7 describes the activation of the site y in a protein where all the sites x_i are phosporylated.

So we partitioned the set of reachable configurations for the protein X into $(m+1)2^n\sim$-equivalence classes, where, in our case, $m = 2$ and $n = 1$. From this, we have been able to pass from an original set of twelve reactions to a simplified one with just seven reactions.

1.2.4 A protein with information flow and symmetries

The two kinds of model reduction that we have seen respectively in Sections 1.2.1 and 1.2.2, and in Section 1.2.3 can be combined. Let us introduce an example to illustrate this.

We consider two kinds of agents, P and X. Instances of P denotes phosphate ions, whereas instances of X denote copies of a given protein. We assume that each protein X has two kinds of sites:

- m sites $x_1, .., x_m$;
- n sites $y_1, .., y_n$;

where m and n are two integer parameters of the model.

We assume that each site can recruit at most one phosphate ion P and, then, dissociate from it.

The state of a protein A is denoted as an (ordered) tuple of symbols among $\{u, p\}$. The symbol u indicates an unphosphorylated site, whereas the symbol p indicates a phosphorylated one. For example, if $m = 2$ and $n = 1$, a protein X having the sites x_1 and y_1 phosphorylated and the site x_2 unphosphorylated is denote by $X[p, u, p]$.

$P + X[u,u,u] \leftrightarrow X[p,u,u]$
$P + X[u,u,p] \leftrightarrow X[p,u,p]$
$P + X[u,p,u] \leftrightarrow X[p,p,u]$
$P + X[u,p,p] \leftrightarrow X[p,p,p]$

(a) Phosphorylation and dephosphorylation of the first site.

$P + X[u,u,u] \leftrightarrow X[u,p,u]$
$P + X[u,u,p] \leftrightarrow X[u,p,p]$
$P + X[p,u,u] \leftrightarrow X[p,p,u]$
$P + X[p,u,p] \leftrightarrow X[p,p,p]$

(b) Phosphorylation and dephosphorylation of the second site.

$P + X[u,u,u] \leftrightarrow X[u,u,p]$
$P + X[u,p,u] \leftrightarrow X[u,p,p]$
$P + X[p,u,u] \leftrightarrow X[p,u,p]$
$P + X[p,p,u] \leftrightarrow X[p,p,p]$

(c) Phosphorylation and dephosphorylation of the third site.

Figure 1.4: Chemical reactions for $m = 2$ and $n = 1$.

We assume that the sites $x_1, .., x_m$ can all be phosphorylated at the same rate constant k. This means that for each integer i between 1 and m the phosphorylation of the site x_i does not depend on the phosphorylation state of the other sites. Moreover, we also assume that for every integer j between 1 and n, the phosphorylation of the site y_j depends both on the index j and on the number of sites among $x_1, .., x_m$ which are currently phosphorylated in the protein X. The rate constant of activation of the site y_j in a protein X which have exactly i sites among $x_1, .., x_m$ activated is denoted by $k_{j,i}$. Lastly, we assume that every phosphorylated site can be unphosphorylated at the same rate constant k_d.

In Figure 1.4 we show the set of reactions for the model with parameters $m = 2$ and $n = 1$.

In the general case, there are 2^{m+n} reachable configuration for the protein X. Thus, when $m+n$ gets big, we can no longer enumerate chemical species, nor reactions, and the integration of the differential semantics is impossible due to combinatorial complexity. We notice that we can use symmetries among the sites $x_1, .., x_m$ so as to reduce the dimension of the semantics. Indeed, what is important is how many sites x_i of the protein are phosphorylated and not which of these sites are phosphorylated. Thus we propose to ignore any distinction between them. In this way, the set of reachable configurations for the protein X is partitioned into $(m + 1)2^n$ $\sim$-equivalence classes. A simplified set of reactions can be proposed, by choosing a representative among each $\sim$-equivalence class. Indeed, up to updating reaction rate constants, we may assume that the sites $x_1, .., x_m$ are always phosphorylated in increasing order, and dephosphorylated in decreasing order. In Figure 1.5 we show the set of the so obtained simplified reactions (for $m = 2$ and $n = 1$). We can notice that whenever the sites x_1 and x_2 are both unphosphorylated, only the site x_1 can be phosphorylated (with a rate twice as big as in the initial reaction) and that whenever the sites x_1 and x_2 are both phosphorylated, only the site x_2 can be dephosphorylated (at a rate constant twice bigger than in the initial reaction).

$$P + X[u,u,u] \leftrightarrow X[p,u,u]$$
$$P + X[u,u,p] \leftrightarrow X[p,u,p]$$

(a) Phosphorylation and dephosphorylation of the first site.

$$P + X[p,u,u] \leftrightarrow X[p,p,u]$$
$$P + X[p,u,p] \leftrightarrow X[p,p,p]$$

(b) Phosphorylation and dephosphorylation of the second site.

$$P + X[u,u,u] \leftrightarrow X[u,u,p]$$
$$P + X[p,u,u] \leftrightarrow X[p,u,p]$$
$$P + X[p,p,u] \leftrightarrow X[p,p,p]$$

(c) Phosphorylation and dephosphorylation of the third site.

Figure 1.5: Simplified chemical reactions for $m = 2$ and $n = 1$.

This reduced system can be refined further by noticing that the behaviour of each site y_j is only controlled by the state of the sites x_i (when the integer i ranges among 1 and m). As a consequence, we can safely cut each protein into n parts, each one documenting the state of only one site y_j and the state of every site x_i. As a result, we obtain a self-consistent differential equations made of $2(m+1)n$ variables (instead of 2^{m+n} in the initial model).

1.2.5 Conclusive remarks

We have described four examples where we have been able to reduce the dimension of the underlying systems of ODEs.

In the two first ones we have been able to split the original system into two independent ones, by exploiting a lack of flow of information between some sites. More specifically, in the first example we got two independent subsystems. The transformation is invertible and we are able to recover the concentration of any species. This is a strong property which is hard to prove and which is hardly ever satisfied. In the second example we obtained two self-consistent subsystems. There we abstracted away some information and we are not able to recover the concentration of any species any longer. This reduction is based on a weaker property, that we are more likely to encounter in realistic models.

In the third example we exploited another kind of property. We observed that some sites exhibit the same capabilities so we can forget about their position and consider just their cardinality. Unlike the two first examples, we do not cut the species, but still we carry less information about them.

In the fourth example, we showed that it is possible to combine the reductions that are based on the lack of information flow and the ones that are based on symmetries.

By these examples we showed that it is possible to exploit structural features of a given differential system to identify a set of macro-variables, i.e. a specific combination of the system variables, that have a self-consistent dynamic. In the rest of the thesis we will show also how this can be done in the

case where the dynamic is described by rules, and not just by reactions. As we will see, working with rules as a big advantage. In fact it is possible to exploit the high level of description offered by the rules, so as to derive directly the reduction from the rules without having to ever explicitly consider the ground set of corresponding reactions.

1.3 Overview

1.3.1 Kappa

Kappa is a formal language based on site graph rewriting. Proteins are typically denoted by agents. The properties of a given agent are expressed by some sites that may carry an internal state. Moreover, potential links between proteins are specified by bonds between agent sites. Interactions between sites are described by the means of rules. These rules are context-free, which means that there is no need to describe the whole state of each chemical species, but only the information that matters to trigger or to tune the speed of the interaction.

For example, a rule of the form:

$$\mathtt{A_1(b)}, \mathtt{B_2(a)} \quad \rightarrow \quad \mathtt{A_1(b^{B_2@a})}, \mathtt{B_2(a^{A_1@b})}$$

describes how the proteins A and B bind together. This can happen when the site b in A and the site a in B are both free. In the rule, there is no need to consider the sites of the proteins the state of which has no influence on the reaction. For instance, we may assume (as in the example of Section 1.2.1) that the protein B has another site that can bind to a protein C and that this potential bond does not influence the capability of B to bind with A. Under this assumption, our rule is a symbolic representation for the following set of reactions:

$$\begin{array}{lll} (1) & A, B & \rightarrow \quad AB, \\ (2) & A, BC & \rightarrow \quad ABC. \end{array}$$

Kappa is formally introduced in Chapter 3. There we describe the effect of applying a rule to a pattern. This is used to define the set of reactions induced by a set of Kappa rules. Then we introduce the differential semantics of a Kappa model as the solution of the set of differential equations that are obtained by applying the principle of mass action. This construction is required to state and prove the soundness of our analyses. Yet we will never compute effectively neither the underlying set of reactions, nor the set of differential equations.

In Chapter 5, we show that we can factor the consumption and the production of a given pattern as an expression of other patterns. It is worth noting that we deal faithfully with potential side effects, i.e. transformations that are not explicitly described by the rules (for instance, in the case we delete an agent without checking that all of its sites are free).

1.3.2 Symmetries based reduction

Some biological systems exhibit proteins with sites having the same capabilities of interaction. For example, a protein can have several phosphorylation sites, but the only relevant information is given by the number of them that are phosphorylated and not what is their position. We define this kind of sites as *symmetric*. Roughly speaking, we say that two sites x and y are symmetric for a system of rules $\mathcal{R}$ if any potential interaction with the site x can be done with the site y instead, and at the same rate constant.

This notion is formalised in Chapter 4. There we introduce an operation of transposition that permutes the state of sites in agent instances. Then this operation is lifted to the level of patterns, rules, agent matchings, pattern embeddings and transition labels. It follows that our notion of symmetry is compatible with the operational semantics: it induces a bisimulation.

Bisimulations [43] are an established form of equivalence for transition systems. Here we use a quantitative version of bisimulation that accounts for ODEs semantics. Given two systems, a bisimulation is a couple of equivalence relations between their states and their transitions. A system with symmetries is equivalent to a reduced system where the states are lumped according to their equivalence classes.

1.3.3 Context-insensitive flow analysis

In computations, the final value of a variable x may depend on the initial value of a variable y. This is captured by the notion of *information flow* [1]. In our specific example we say that the variable y flows into x. When a variable x flows in n variables $y_1, \ldots, y_n$ we face to a potential correlation. Symmetrically, when n variables $y_1, \ldots, y_n$ flow in a single variable z, the correlation among $y_1, \ldots, y_n$ may constrain the behaviour of the variable z.

In Chapter 6 we propose a method to over-approximate all useful correlations. The result is a linear change of variables which reduces the dimension of the initial semantics. The reduced system is computed directly at the level of rules and we do not need any effective computation of the original one. Given a set of species and a set of rules we firstly compute a symbolic representation of all the possible species. We call this representation *contact*

map. Then, scanning rule by rule, we annotate the contact map with some information between the sites that are tested and sites that are modified. This annotation induces a set of patterns, that we call fragments. Roughly speaking, fragments are patterns that are obtained by starting from a single site and extending the pattern by following the information flow backward. That is to say that a fragment describes only the states of the sites that impact the behaviour of the starting site. By construction, we can prove that the consumption and the production of every fragment can be expressed as an expression of the concentration of the other fragments.

1.3.4 Context-sensitive flow analysis

The analysis in Chapter 6 is context-insensitive. It gathers all the information about the sites of each protein in a single node, regardless the state of its sites.

Between a concrete representation of a set of species and the one that is based on the contact map, there are several intermediate representations. We show that it is possible to tune the description of the information flow by changing the level of representation. In Chapter 7, we introduce Σ-graphs to generalise the concept of contact map. In Σ-graphs, each kind of agent may occur several times, which enables to refine the annotation of the flow of information according to different contextual conditions.

In Chapter 8, we show how to abstract the flow of information at different levels of resolution with respect to the choice of a Σ-graph and we extend the definition of fragments accordingly. We notice that, in this more general setting, the constraints on the flow of information are not enough to ensure the self-consistency of the fragments. Indeed, we notice that the soundness of our approach relies on the fact that the grain of resolution in the abstraction of the reactants of a reaction should always be at least as refined as the one of the products of this reaction: this is the principle of backward compatibility. We propose a closure operator about the annotations of Σ-graphs, which computes the least annotation that is backward compatible. As a result, we get a sound annotation of arbitrary Σ-graphs, that induces self-consistent sets of fragments.

Our method is a kind of partitioning [7], a generic method for refining abstractions.

1.4 Related works

1.4.1 Bisimulation

In [9] is proposed a notion of bisimulation for weighted labeled transition systems. It is shown that it is possible to reduce them by merging individual states into equivalence classes while the behaviour of the complete automaton is preserved. Our notion of bisimulation is induced by an equivalence on variables, so less general. Yet, we do this restriction in order to have means to discover efficiently these bisimulations. The framework presented in [9] provides a very general definition of bisimulation, but this is not clear how to find these bisimulations in practice.

In [13], equivalence relations among variables are used to induce bisimulations also in the context of reaction networks. Indeed the characterisation of the most general bisimulation that can be expressed this way is provided, as well as an effective algorithm to compute it. Yet, the models we address in this thesis are highly combinatoric and they usually cannot be described compactly as a reaction network. Thus, the approach that is proposed in [13] does not scale for our purpose. In [44], a more general framework to discover linear changes of variables in reaction networks is proposed. The same problem of scalability persists.

In [12], it is shown how to relate two reaction networks, automatically, by the means of a bisimulation induced by an equivalence relation over the variables of the system. Even this method needs the computation of the reaction networks and suffers from the same problems of scalability.

The concept of bisimulation is highly related to the one of lumpability [8]. A Markov chain is lumpable with respect to a given aggregation (quotienting) of its states, if the lumped chain preserves the Markov property. The relation between lumpability and bisimulation is studied in [29].

1.4.2 Information flow

Dependencies between sites and reactions have been used in systematic methods for (hand-)writing coarse-grained models. In [6] it is showed, by examples, that a mechanistic reduction of the system can by computed using hierarchical control relationships between sites. In [15] a reduced model is written directly from the so called control graph, characterising the information flow. However these methods are not automated (there is no semantics) and suffer from combinatorial blow up in the case of chains of agents and site modifications that propagate through bindings [16]. This problem can be solved by neglecting some species but then the dynamics of the system is

not preserved. The framework in [5], for automatically reducing the ODEs of protein networks, suffers from the same lack of soundness when site modifications propagate through bindings. Our approach deals efficiently with this case: even in the case of site modifications that propagate through bindings, we capture just enough information to ensure soundness, but few enough information to ensure scalability.

Our context-insensitive information flow analysis improves and extends the fully context insensitive ones that have been proposed in [25, 28]. There the species are cut in a homogeneous way just according to a type information. It is worth noting that in [28] it is considered a set of 71 rules expanding into 18 051 984 143 555 729 567 species that is reduced into 175 988 fragments. Our analysis is a strict improvement of the one presented there.

[36] proposes a fully context sensitive analysis where the abstraction of the information flow is done on the concrete, thanks to a direct interaction on molecular species. Then a self consistent change of variables is computed by saturation. Moreover, it applies on pure Kappa only (without side effects). Our framework provides a full span of intermediary abstractions and accounts for side effects.

1.4.3 Stochastics semantics

Reduced models have been proposed also for stochastics semantics. In [30], stochastics fragments are proposed to track the flow in CTMCs models. The reduction is not very impressive. This is due to the fact that more potential correlations become meaningful in a stochastic setting.

In [39, 40], it is shown that the differential semantics is the expected behaviour of the stochastic one when both the initial mixture size and the volume diverge with a constant ratio. In this case, ground refinements with lower arities, corresponding to epis which do not preserve the number of components, are negligible anyway.

1.4.4 Other model reduction

Other kinds of reduction have been studied. In [32], it is proposed a comparison between models based on topological properties. The perspective is more on explaining which transformations are necessary to go from a model to another one, regardless of quantitave properties.

Concentrations and time-scales separation plays a very important role in model reduction. The framework in [34, 45] applies concentration and time-scales separation to reduce linear reaction networks. The reduced system is build on the base of a dominant kinetic system over the constant rate of the

original one. Tropicalisation [46,47] permits to generalise these approaches to arbitrary systems of ODEs. Those methods rely on a solid physical justification. Yet, they perform numerical approximations without providing explicit bounds on the accumulated errors. There are valid under the asymptotic assumption that fast reactions are infinitely quicker than slow one. In [3], these methods are applied to Kappa models but no proof of soundness is given.

In [41, 42], the authors use time-scale separation to reduce models of metabolic networks. They rely on the fact that some species always reach a fast equilibrium, to replace their transient concentration, by their concentration at equilibrium.

1.5 Outline

In Chapter 2, we describe a generic framework to define and reduce differential semantics for systems of ODEs. We also explain how to combine different kinds of model reduction. In Chapter 3, we introduce Kappa. We define its syntax, semantics and we show how the consumption and the production of species can be expressed directly at the level of rules. In Chapter 4, we propose a framework that exploits symmetries between sites, showing how a relation based on sites having the same capabilities of interaction induces a bisimulation. In Chapter 5, we show how to express the consumption and the production of patterns as an expression of the concentration of other patterns. In Chapter 6, we show how to construct an abstract semantics tracking the flow of information between different regions of chemical species. In Chapter 7, we introduce Σ-graphs as a means to describe relations among the sites of species, at different levels of resolution. In particular we study structure preserving functions to build correspondences between Σ-graphs and between relations on their respective sites. In Chapter 8, we study a framework to tune the analysis accordingly to a context and we supply a way to find smaller reduced differential models.

Chapter 2

Reduction of differential semantics

In this chapter we describe a generic framework to define and reduce differential semantics. A first version of this framework has been introduced in [25, 28]. We have extended this framework in [10, 11].

2.1 Concrete semantics

Let $\mathcal{V}$ be a finite set. Maps from $\mathcal{V}$ to $\mathbb{R}$ form a normed vector space with the norm $\| \cdot \|$ that is defined as follows:

$$\|\rho\| := max_{X \in \mathcal{V}} |\rho(X)|$$

where, for every real number x, $|x|$ denotes the absolute value of x.

For U a subset of $\mathcal{V} \to \mathbb{R}$, we define $\|U\| = sup_{\rho \in U} \|\rho\| \leqslant +\infty$. If $\mathcal{V}$ is a set of species and $\rho(X)$ the concentration of the species X, then $\|\rho\|$ controls the total number (per unit volume) of species in the system. A map ρ such that for all $X \in \mathcal{V}$, $\rho(X) \geqslant 0$ is called a *state*.

Let $\mathcal{V}$ be a set of variables. We define a *state* over $\mathcal{V}$ as a mapping ρ from $\mathcal{V}$ to $\mathbb{R}$ such that:

$$\rho(a) \geqslant 0 \text{ for all } a \in \mathcal{V}.$$

We define an *autonomous system* as a pair $(\mathcal{V}, \mathbb{F})$, where $\mathcal{V}$ is a finite set of variables and $\mathbb{F}$ is a continuously differentiable function from $\mathcal{V} \to \mathbb{R}$ to $\mathcal{V} \to \mathbb{R}$.

Example 2.1.1. *We illustrate this definition on the case study of Section 1.2.1. Following the principle of mass action law, the reactions:*

$$\begin{array}{llll}(1) & A, B & \leftrightarrow & AB & k_A, k'_A\\ (2) & A, BC & \leftrightarrow & ABC & k_A, k'_A\\ (3) & B, C & \leftrightarrow & BC & k_C, k'_C\\ (4) & AB, C & \leftrightarrow & ABC & k_C, k'_C.\end{array}$$

are inducing the following system of differential equations:

$$\begin{array}{l}[A]' = k'_A([AB] + [ABC]) - k_A[A]([B] + [BC])\\ [B]' = k'_A[AB] + k'_C[BC] - [B](k_A[A] + k_C[C])\\ [C]' = k'_C([BC] + [ABC]) - k_C[C]([B] + [AB])\\ [AB]' = k_A[A][B] + k'_C[ABC] - [AB](k'_A + k_C[C])\\ [BC]' = k_C[B][C] + k'_A[ABC] - [BC](k'_C + k_A[A])\\ [ABC]' = k_A[A][BC] + k_C[AB][C] - [ABC](k'_A + k'_C).\end{array}$$

This system of differential ordinary equations can be described as an autonomous system $(\mathcal{V}, \mathbb{F})$ *where:*

- $\mathcal{V} := \{A, B, C, AB, BC, ABC\}$;

$$\text{- } \mathbb{F} := \begin{cases} \mathbb{R}^{\mathcal{V}} & \to & \mathbb{R}^{\mathcal{V}} \\ & & \\ \rho & \mapsto & \begin{cases} \mathcal{V} & \to & \mathbb{R} \\ A & \mapsto & k'_A(\rho(AB) + \rho(ABC)) \\ & & \quad -k_A\rho(A)(\rho(B) + \rho(BC)) \\ B & \mapsto & k'_C(\rho(BC) + \rho(ABC)) \\ & & \quad -k_C\rho(C)(\rho(B) + \rho(AB)) \\ C & \mapsto & k'_A\rho(AB) + k'_C\rho(BC) \\ & & \quad -\rho(B)(k_A\rho(A) + k_C\rho(C)) \\ AB & \mapsto & k_A\rho(A)\rho(B) + k'_C\rho(ABC) \\ & & \quad -\rho(AB)(k'_A + k_C\rho(C)) \\ BC & \mapsto & k_C\rho(B)\rho(C) + k'_A\rho(ABC) \\ & & \quad -\rho(BC)(k'_C + k_A\rho(A)) \\ ABC & \mapsto & k_A\rho(A)\rho(BC) + k_C\rho(AB)\rho(C) \\ & & \quad -\rho(ABC)(k'_A + k'_C). \end{cases} \end{cases}$$

By the Cauchy-Lipschitz theorem [37], for any state ρ_0 over $\mathcal{V}$, the system of equations:

$$\begin{cases}\rho'(t) = \mathbb{F}(\rho(t))\\ \rho(0) = \rho_0\end{cases}$$

has a unique maximal differentiable solution: $f_{\rho_0} : [0, T_{\rho_0}) \to (\mathcal{V} \to \mathbb{R})$, with $T_{\rho_0} \leqslant +\infty$.

The *concrete semantics* of the system $(\mathcal{V}, \mathbb{F})$ is the mapping $[\![\mathcal{V}, \mathbb{F}]\!]$ which associates the unique maximal differentiable solution f_{ρ_0} to each state ρ_0 over $\mathcal{V}$ (in this context, ρ_0 is called the initial state). An autonomous system $(\mathcal{V}, \mathbb{F})$ is said to be *positive* if, and only if, for any state ρ_0 over $\mathcal{V}$ and any $t \in [0, T_{\rho_0})$, $f_{\rho_0}(t)$ is a state over $\mathcal{V}$ as well.

It is worth noting that any autonomous system that is defined by the means of the mass action law principle is positive.

2.2 Exact reduction

Starting from a positive autonomous differential system, we want to define a specific notion of linear reduction for it. To do this we recur to ideas from Abstract Interpretation, a general framework to approximate the semantics of programs [18]. This way, the notion of system reduction that we use can be seen as an *abstraction map*, that is a transformation from a concrete state space to an abstract one. For us, the concrete state space is given by the set of species concentrations, whereas the abstract one is given by the concentrations of some abstract observables.

If we consider two sets $\mathcal{V}$ and $\mathcal{V}^\sharp$, a map ψ from $\mathcal{V} \to \mathbb{R}$ to $\mathcal{V}^\sharp \to \mathbb{R}$ is:

- *positive* if for all $\rho \geqslant 0$, $\psi(\rho) \geqslant 0$;
- *expansive* if for all subset U of $\mathcal{V} \to \mathbb{R}^+$, $\|\psi(U)\| < +\infty$ implies $\|U\| < +\infty$.

We write $\psi : \mathbb{R}^A \multimap \mathbb{R}^B$ whenever ψ is a linear positive expansive map from the normed vector space $\mathbb{R}^A$ to the normed vector space $\mathbb{R}^B$.

We define a *model reduction* as a tuple $(\mathcal{V}, \mathbb{F}, \mathcal{V}^\sharp, \psi, \mathbb{F}^\sharp)$ where:

- the pair $(\mathcal{V}, \mathbb{F})$ is an autonomous system,
- $\mathcal{V}^\sharp$ is a (finite) set of abstract variables,
- ψ is a linear positive expansive map from $(\mathcal{V} \to \mathbb{R})$ to $(\mathcal{V}^\sharp \to \mathbb{R})$,
- $\mathbb{F}^\sharp$ a continuously differentiable function from $(\mathcal{V}^\sharp \to \mathbb{R})$ to $(\mathcal{V}^\sharp \to \mathbb{R})$,

and such that the following diagram commutes:

$$\begin{array}{ccc} \mathbb{R}^{\mathcal{V}} & \xrightarrow{\mathbb{F}} & \mathbb{R}^{\mathcal{V}} \\ {\scriptstyle\psi}\downarrow & & \downarrow{\scriptstyle\psi} \\ \mathbb{R}^{\mathcal{V}^\sharp} & \xrightarrow{\mathbb{F}^\sharp} & \mathbb{R}^{\mathcal{V}^\sharp} \end{array}$$

Since ψ is positive we have that (concrete) states are mapped to (abstract) states.

Example 2.2.1 (continued)**.** *Keeping on with the example of Section 1.2.1, the autonomous system $(\mathcal{V}, \mathbb{F})$ can be completed into a model reduction as follows.*

The set of observables $\mathcal{V}^\sharp$ is defined as $\{A, AB?, B?, C, ?BC, ?B\}$. The concentration of these observables is defined by the following constraints:

$$\begin{aligned} [AB?] &= [AB] + [ABC] \\ [B?] &= [B] + [BC] \\ [?BC] &= [BC] + [ABC] \\ [?B] &= [B] + [AB] \end{aligned}$$

which can be formalised by the following abstraction function ψ:

$$\psi \;:\; \left\{ \begin{array}{rcl} \mathbb{R}^{\mathcal{V}} & \to & \mathbb{R}^{\mathcal{V}^\sharp} \\ \rho & \mapsto & \left\{ \begin{array}{rcl} \mathcal{V} & \to & \mathbb{R} \\ A & \mapsto & \rho(A) \\ AB? & \mapsto & \rho(AB) + \rho(ABC) \\ B? & \mapsto & \rho(B) + \rho(BC) \\ C & \mapsto & \rho(C) \\ ?BC & \mapsto & \rho(BC) + \rho(ABC) \\ ?B & \mapsto & \rho(AB) + \rho(B). \end{array} \right. \end{array} \right.$$

The behaviour of the observables is defined by the following equations:

$$\begin{aligned} &[A]' = k'_A[AB?] - k_A[A][B?] \\ &[AB?]' = k_A[A][B?] - k'_A[AB?] \\ &[B?]' = k'_A[AB?] - k_A[A][B?] \qquad [C]' = k'_C[?BC] - k_C[C][?B] \\ &[?BC]' = k_C[C][?B] - k'_C[?BC] \\ &[?B]' = k'_C[?BC] - k_C[C][?B]. \end{aligned}$$

These equations can be encoded in the function $\mathbb{F}^\sharp$ that is defined as fol-

lows:

$$\mathbb{F}^\sharp : \left\{ \begin{array}{rcl} \mathbb{R}^{\mathcal{V}^\sharp} & \to & \mathbb{R}^{\mathcal{V}^\sharp} \\ \rho^\sharp & \mapsto & \left\{ \begin{array}{rcl} \mathcal{V}^\sharp & \to & \mathbb{R} \\ A & \mapsto & k'_A\rho^\sharp(AB?) - k_A\rho^\sharp(A)\rho^\sharp(B?) \\ AB? & \mapsto & k_A\rho^\sharp(A)\rho^\sharp(B?) - k'_A\rho^\sharp(AB?) \\ B? & \mapsto & k'_A\rho^\sharp(AB?) - k_A\rho^\sharp(A)\rho^\sharp(B?) \\ C & \mapsto & k'_C\rho^\sharp(?BC) - k_C\rho^\sharp(C)\rho^\sharp(?B) \\ ?BC & \mapsto & k_C\rho^\sharp(C)\rho^\sharp(?B) - k'_C\rho^\sharp(?BC) \\ ?B & \mapsto & k'_C\rho^\sharp(?BC) - k_C\rho^\sharp(C)\rho^\sharp(?B). \end{array} \right. \end{array} \right.$$

The following theorem holds.

Theorem 2.2.2 (Soundness)**.** *Let $(\mathcal{V}, \mathbb{F}, \mathcal{V}^\sharp, \psi, \mathbb{F}^\sharp)$ be a model reduction. Let ρ_0 be an initial state over $\mathcal{V}$. We denote as $f : [0, T) \to \mathbb{R}^{\mathcal{V}}$ the maximal solution of the autonomous system $(\mathcal{V}, \mathbb{F})$ with initial condition $f(0) = \rho_0$ and as $f^\sharp : [0, T^\sharp) \to \mathbb{R}^{\mathcal{V}^\sharp}$ the maximal solution of the autonomous system $(\mathcal{V}^\sharp, \mathbb{F}^\sharp)$ with initial condition $f^\sharp(0) = \psi(\rho_0)$.*

With these notations, we have: $T = T^\sharp$ and $f^\sharp = \psi \circ f$.

Proof. Since f is differentiable on $[0, T)$ and because ψ is linear, the function $\psi \circ f$ is diferentiable on $[0, T)$.

Moreover, for $t < T$, one has:

$$\begin{array}{rcl} (\psi \circ f)'(t) & = & \psi(f'(t)) \\ & = & \psi(\mathbb{F}(f(t))) \\ & = & \mathbb{F}^\sharp(\psi(f(t))) \end{array}$$

because ψ is linear, f is a solution of $\mathbb{F}$ and $\psi \circ \mathbb{F} = \mathbb{F}^\sharp \circ \psi$ by assumption. Hence $\psi \circ f$ is differentiable on $[0, T)$, and it is the (unique) solution of the autonomous system $(\mathcal{V}^\sharp, \mathbb{F}^\sharp)$ for the initial condition $\psi(\rho_0)$ on $[0, T)$. In other words, on $[0, T)$ we have $\psi \circ f = f^\sharp$. It follows that $T \leqslant T^\sharp$. But, in fact, $T = T^\sharp$. To see this, suppose $T < \infty$, then $\|f(t)\|$ diverges as t tends to T and ψ being expansive, so does $\|\psi(f(t))\|$. □

It follows from Theorem 2.2.2 that if the system $(\mathcal{V}, \mathbb{F})$ is positive, then the system $(\mathcal{V}^\sharp, \mathbb{F}^\sharp)$ is positive as well.

The reduction guarantees that:

1. trajectories of abstract variables can be computed directly in the abstract without loss of information;

2. positivity is preserved;

3. the life-time of the system is also preserved.

This way, our reduction not only preserves the value of the observables, but also the life-time of the initial model. We do not consider, as a valid model reduction, a change of variables that may lose a vertical asymptote. This situation can be compared to what happens with program slicing. With syntactic program slicing, a slice of a non-terminating program may terminate, whereas some version of semantics slicing ensures by definition that the slicing process preserves non-termination [14, 33].

2.3 Projections-based reductions

In this section, we specialise this framework to the case where the model reduction can be induced by a bisimulation relation induced by an equivalence relation over the variables of the system.

We start by taking:

1. a concrete autonomous system $(\mathcal{V}, \mathbb{F})$,

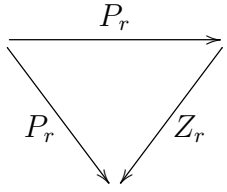
2. a function r from $\mathcal{V}$ to $\mathcal{V}$ such that r is idempotent (i.e. $r \circ r = r$).

We use the function r to define an equivalence relation over $\mathcal{V}$, by saying that $v_1 \sim_r v_2$ if and only if $r(v_1) = r(v_2)$. Moreover, for any variable $v \in \mathcal{V}$ the variable $r(v)$ is called the representative of the equivalence class of v.
We define two linear projections P_r and Z_r as follows:

$$P_r(\rho) : \begin{cases} \mathcal{V} \to \mathbb{R} \\ v \mapsto \sum\{\rho(v') \mid r(v') = v\}, \end{cases} \quad \text{and} \quad Z_r(\rho) : \begin{cases} \mathcal{V} \to \mathbb{R} & \\ v \mapsto \rho(v) & \text{if } r(v') = v \\ v \mapsto 0 & \text{otherwise.} \end{cases}$$

P_r is used to gather the values of each $\sim_r$-equivalent variable and to store the result to the value of the representative of each $\sim_r$-equivalence class. Z_r ignores the values of the variables which are not the representative of their $\sim_r$-equivalence class.

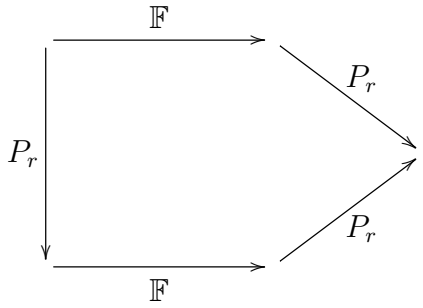
We notice that $P_r : \mathbb{R}^{\mathcal{V}} \multimap \mathbb{R}^{\mathcal{V}}$ and that the following diagram commutes:



Now we provide additional requirements over the relation r so that it induces a bisimulation relation over the states of the system.

Definition 2.3.1. *Given an autonomous system $(\mathcal{V}, \mathbb{F})$ and a relation r, we say that that r induces a bisimulation over $(\mathcal{V}, \mathbb{F})$ if and only if for any pair (ρ, ρ') of states over $\mathcal{V}$, if $P_r(\rho) = P_r(\rho')$, then $P_r(\mathbb{F}(\rho)) = P_r(\mathbb{F}(\rho'))$.*

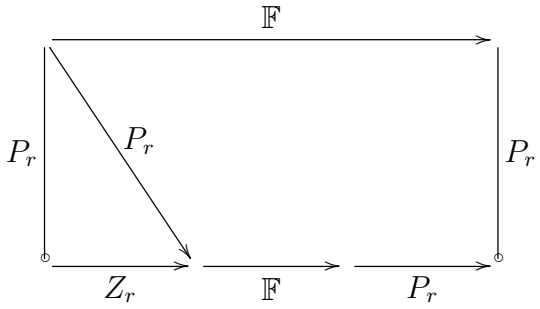
The previous definition is equivalent to say that the relation r induces a bisimulation if, and only if, the following diagram commutes:



The following theorem states that whenever the relation r induces a bisimulation, then the projections P_r and Z_r can be used to define a model reduction.

Theorem 2.3.2. *If r induces a bisimulation, then $(\mathcal{V}, \mathbb{F}, \mathcal{V}, P_r, P_r \circ \mathbb{F} \circ Z_r)$ is a model reduction.*

Proof. The proof is given by the following commutative diagram:



More precisely, given $\rho \in \mathbb{R}^{\mathcal{V}}$, we have:

$$\begin{aligned} P_r(\mathbb{F}(\rho)) &= P_r(\mathbb{F}(P_r(\rho))) \\ &= P_r(\mathbb{F}(Z_r(P_r(\rho)))) \\ &= [P_r \circ \mathbb{F} \circ Z_r](P_r(\rho)). \end{aligned}$$

□

$(u,u) \to (p,u)\ @k_1$
$(u,p) \to (p,p)\ @k_1$

(a) Phosphorylation of the first site.

$(u,u) \to (u,p)\ @k_1$
$(p,u) \to (p,p)\ @k_1$

(b) Phosphorylation of the second site.

$(p,p) \to \quad @k_2$

(c) Degradation.

Figure 2.1: Chemical reactions.

$(u,u) \to (p,u)\ @2k_1$

(a) Phosphorylation of the first site.

$(p,u) \to (p,p)\ @k_1$

(b) Phosphorylation of the second site.

$(p,p) \to \quad @k_2$

(c) Degradation.

Figure 2.2: Simplified chemical reactions.

Example 2.3.3. *We instantiate our framework on a simple example.*

In this example we consider one kind of protein. We suppose that this protein has two sites. The state of a protein is denoted as an ordered couple of symbols among $\{u,p\}$*. The symbol* u *indicates an unphosphorylated site, whereas the symbol* p *indicates a phosphorylated one. For example, a protein having both sites phosphorylated is denoted as* (p,p)*.*

We start by considering the reactions that are given in Figure 2.1. Each site of a protein can be phosphorylated with the same rate constant k_1*. Moreover, we assume that a protein that has both sites phosphorylated can be degraded with a rate constant* k_2*. Intuitively, both sites of the protein play exactly the same role. Our goal is to justify that we can abstract each instance of a protein by the number of its sites that are phosphorylated. This way, we would get the simplified rules that are given in Figure 2.2.*

The set of variables $\mathcal{V}$ *is defined as:*

$$\{(u,u),(u,p),(p,u),(p,p)\}.$$

We want to identify the proteins that have exactly one activated site, no matter if it is the first or the second one. Thus, we define the mapping r *by*

$$r(u,u) = (u,u),$$
$$r(u,p) = r(p,u) = (u,p),$$
$$r(p,p) = (p,p).$$

Since we have only unary reactions, the underlying differential system is linear, and we can use matrix notation (but our framework also apply to non linear systems).

We denote each element ρ of $\mathbb{R}^{\mathcal{V}}$ as a vector $[\rho(u,u), \rho(u,p), \rho(p,u), \rho(p,p)]$. We get, by using a matrix notation:

$$\mathbb{F} = \begin{bmatrix} -2k_1 & 0 & 0 & 0 \\ k_1 & -k_1 & 0 & 0 \\ k_1 & 0 & -k_1 & 0 \\ 0 & k_1 & k_1 & -k_2 \end{bmatrix}, P_r = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } Z_r = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Then we can check that:

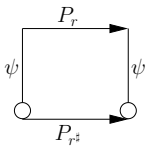
$$P_r \times \mathbb{F} \times Z_r = \begin{bmatrix} -2k_1 & 0 & 0 & 0 \\ 2k_1 & -k_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & k_1 & 0 & -k_2 \end{bmatrix},$$

thats to say that the matrix $P_r \times \mathbb{F} \times Z_r$ is exactely the one that is induced by the set of the simplified reaction in Figure 2.2.

2.4 Combining model reductions

An existing model reduction can be abstracted further thanks to a bisimulation induced by an equivalence relation over the concrete variables.

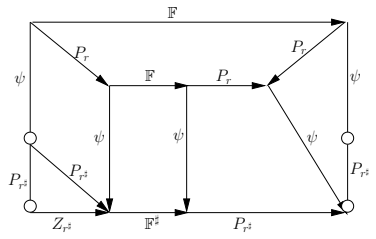
Theorem 2.4.1. *Let $(\mathcal{V}, \mathbb{F}, \mathcal{V}^\sharp, \psi, \mathbb{F}^\sharp)$ be a model reduction, r be an idempotent mapping over $\mathcal{V}$ such that r induces a bisimulation over the autonomous system $(\mathcal{V}, \mathbb{F})$, and $r^\sharp$ be an idempotent mapping over $\mathcal{V}^\sharp$. We assume that the following square:*



commutes.

Under this assumption, the tuple $(\mathcal{V}, \mathbb{F}, \mathcal{V}^\sharp, P_{r^\sharp} \circ \psi, P_{r^\sharp} \circ \mathbb{F}^\sharp \circ Z_{r^\sharp})$ is a model reduction.

Proof. The proof is given by the following commutative diagram:



More precisely, given $\rho \in \mathbb{R}^{\mathcal{V}}$, we have that:

$$\begin{aligned}
[P_{r^\sharp} \circ \psi](\mathbb{F}(\rho)) &= [\psi \circ P_r](\mathbb{F}(\rho)) \\
&= \psi(P_r(\mathbb{F}(\rho))) \\
&= \psi(P_r(\mathbb{F}(P_r(\rho)))) \\
&= P_{r^\sharp}(\psi(\mathbb{F}(P_r(\rho)))) \\
&= P_{r^\sharp}(\mathbb{F}^\sharp(\psi(P_r(\rho)))) \\
&= P_{r^\sharp}(\mathbb{F}^\sharp(P_{r^\sharp}(\psi(\rho)))) \\
&= P_{r^\sharp}(\mathbb{F}^\sharp(Z_{r^\sharp}(P_{r^\sharp}(\psi(\rho)))))
\end{aligned}$$

□

This way, the abstraction function is obtained as composing the initial abstraction function ψ with the projection $P_{r^\sharp}$ over the abstract observables. Moreover, the abstract counterpart is defined as $P_{r^\sharp} \circ \mathbb{F}^\sharp \circ Z_{r^\sharp}$, meaning that:

1. firstly, we ignore the contribution of each observable which is not the representative of its equivalence class;

2. then, we apply the former abstract counterpart function $\mathbb{F}^\sharp$;

3. finally, we collect the contribution of each observable according to their representative.

It is interesting to observe that no commutative diagram is required to relate the functions $\mathbb{F}^\sharp$ and $P_{r^\sharp}$. As a conseguence, we just need to prove that r induces a bisimulation in the concrete. Then, to derive this construction, we have to prove that $P_{r^\sharp} \circ \psi = \psi \circ P_r$. To prove this, only the structure of the abstract variables matters (not their dynamics), so the proof is quite simple.

Example 2.4.2. *We can use our framework to formalise the reduction we have introduced for the example in Section 1.2.4.*

In this example, the protein X has $m + n$ sites: m sites $x_1,\ldots,x_m$ which are pair-wisely symmetric, and n sites $y_1,\ldots,y_n$ which are controlled only by the state of the sites $x_1,\ldots,x_m$.

We define the function r that maps each configuration of the protein X, to the configuration in which:

1. *the state of the site y_j is the same for every i between 1 and n,*

2. *the number of phosphorylated sites among the sites $x_1,\ldots,x_m$ is the same,*

3. *there exists* k *between* 1 *and* m*, such that* x_i *is phosphorylated if and only if* i *ranges between* 1 *and* k*.*

Since the state of each y_j *is only controlled by the state of the sites* x_i*. We can cut each configuration of the protein* X *into* n *fragments. Each fragment documents the state of a site* y_j *and the state of all sites* x_i*. The abstraction function* ψ *maps each state to the function mapping each fragment to the sum of the concentration of the configurations which have the same state for the site* y_j *and the same state for every site* x_i*.*

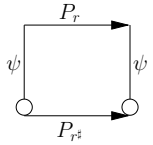
It is worth noting, that whenever a fragment describes the state of a site, it also describes the state of any site that is symmetric to this site. As a consequence we can define, the function $r^\sharp$ *which maps each fragment of the protein* X*, to the fragment in which:*

1. *the same site* y_j *is documented and with the same state,*

2. *the number of phosphorylated sites among the sites* $x_1,\ldots,x_m$ *is the same,*

3. *there exists* k *between* 1 *and* m*, such that* x_i *is phosphorylated if and only if* i *ranges between* 1 *and* k*.*

We can check that the following diagram:



commutes.

As a conclusion, we can simplify the fragments further by taking into account symmetric sites. In fine*, we get* $2(m+1)n$ *fragments.*

2.5 Conclusion

We have introduced a generic framework to formalise model reductions for the ODEs semantics of reaction networks. We have considered two kinds of reduction. The first one is based on generic changes of variables. The second one has more structure, it is induced by a bisimulation that is defined by the means of an equivalence relation over the variables of the ODEs. We have shown how to combine these two kinds of model reduction.

We have stated the soundness of our approach: the maximal solutions of the reduced models are the exact projection of the solutions of the initial system.

We are left to profide effective ways of discovering changes of variables and species-based bisimulations. Whereas some frameworks in the literature [44] have proposed to explore the set of potential reductions directly at the description level of reaction networks, we will use the high level representation offered by Kappa. This way, our algorithm will only scan the set of Kappa rules and we will never compute the underlying set of chemical reactions.

Chapter 3

Kappa

We introduce Kappa. Kappa is a formal language, based on site graph-rewriting, that is suitable to represent and simulate models of signalling pathways [22, 26].

In Kappa chemical species are described as graphs, made of agents. These agents may denote proteins and they may contain some sites. These sites can be bound pair-wise, which encodes for links between proteins. Additionally, sites may carry internal states, that denote properties (as a phosphorylation state for instance). Unlike other site graph-rewriting languages as BNGL [4], in Kappa, in each agents, sites are fully identified by a unique site identifier. The main advantage is that it reduces the potential number of embeddings between site-graphs. Indeed, the number of embeddings from a connected site-graph into another connected site-graph is at most linear in the number of agents of the latter one, whereas it can be exponential with more general classes of site-graph.

Interactions between sites are described by the means of rules. These rules are context-free, which means that there is no need to describe the whole state of each chemical species, but only the information that matters to trigger or to tune the speed of the interaction. Because of this, Kappa offers a high level description of chemical interactions. We will see in further chapters that this granularity of description is helpful to discover important properties of interest, that can then be used to reduce the combinatorial complexity of models.

3.1 Example

Let us recall the example from Section 1.2.1.

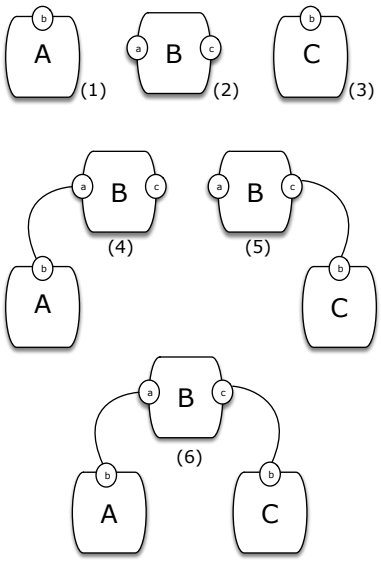


Figure 3.1: Set of species.

We had the following set of reactions:

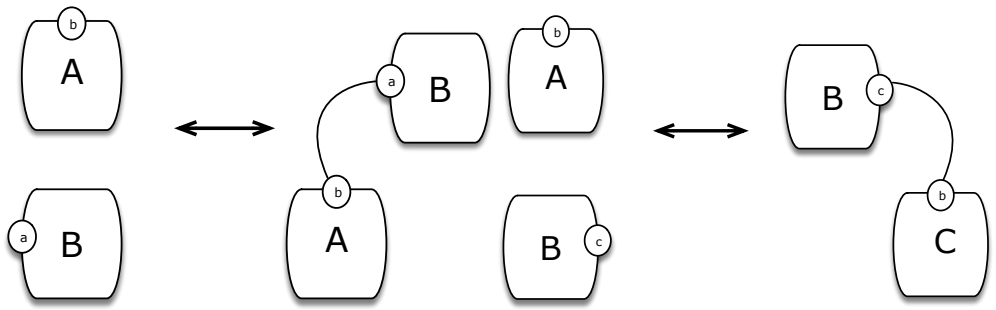
$$\begin{array}{llcll} (1) & A, B & \leftrightarrow & AB & k_A, k'_A \\ (2) & A, BC & \leftrightarrow & ABC & k_A, k'_A \\ (3) & B, C & \leftrightarrow & BC & k_C, k'_C \\ (4) & AB, C & \leftrightarrow & ABC & k_C, k'_C \end{array}$$

over the following set of species:

1. A
2. B
3. C
4. AB
5. BC
6. ABC.

In Figure 3.1 we can see a graphical representation of this set. In Kappa, these species are syntactically denoted as follows:

1. $\mathtt{A_1(b)}$



(a) $\mathtt{A_1(b), B_2(a)} \leftrightarrow \mathtt{A_1(b^{B_2@a}), B_2(a^{A_1@b})}$ (b) $\mathtt{B_1(c), C_2(b)} \leftrightarrow \mathtt{B_1(c^{C_2@b}), C_2(b^{B_1@c})}$

Figure 3.2: Graphical representation of the rules in the example.

2. $\mathtt{B_1(a, c)}$
3. $\mathtt{C_1(b)}$
4. $\mathtt{A_1(b^{B_2@a}), B_2(a^{A_1@b}, c)}$
5. $\mathtt{B_1(a, c^{C_2@b}), C_2(b^{B_1@c})}$
6. $\mathtt{A_1(b^{B_2@a}), B_2(a^{A_1@b}, c^{C_3@b}), C_3(b^{B_2@c})}$.

In Kappa rules, the "don't care, don't write" principle matters: we do not have to specify the whole reactants involved in a reaction but it is enough to express the minimal context that influences this interaction. So a rule can be seen as a symbolical representation for the set of the reactions that can be obtained as a refinement [23, 24] of this rule.

In our example, the set of reactions can be transformed in the following set of rules:

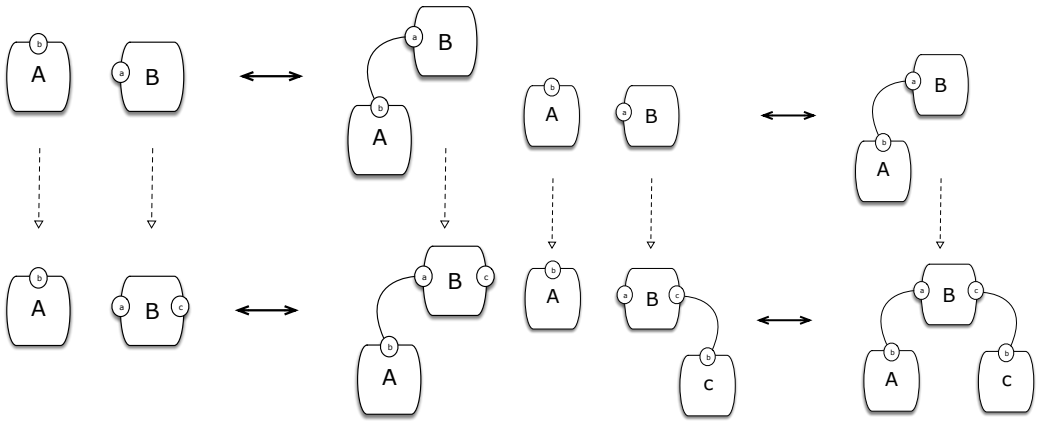
$$\begin{array}{lllll} (a) & \mathtt{A_1(b), B_2(a)} & \leftrightarrow & \mathtt{A_1(b^{B_2@a}), B_2(a^{A_1@b})} & k_A, k'_A \\ (b) & \mathtt{B_1(c), C_2(b)} & \leftrightarrow & \mathtt{B_1(c^{C_2@b}), C_2(b^{B_1@c})} & k_C, k'_C. \end{array}$$

The first rule states that whenever an agent A has a site b that is free, A can bind, by the site b, to an agent B that has its site a free (and neglecting the state of the site c of B, that is to say neglecting whether or not the agent B is already bound to an agent C).

That was expressed by both following reactions:

$$\begin{array}{lllll} (1) & A, B & \leftrightarrow & AB & k_A, k'_A \\ (2) & A, BC & \leftrightarrow & ABC & k_A, k'_A. \end{array}$$

The same way, the second rule states that whenever an agent C has a site c that is free, C can bind, by the site c, to an agent B that has its site c free



(a) When the second site of B is free. (b) When the second site of B is bound.

Figure 3.3: Reactions for binding the first site of B.

(and neglecting the state of the site a of B, i.e. neglecting if the agent B is already bound to an agent A).

That was expressed by both following reactions:

$$\begin{array}{lllll} (3) & B, C & \leftrightarrow & BC & k_C, k'_C \\ (4) & AB, C & \leftrightarrow & ABC & k_C, k'_C. \end{array}$$

In Figures 3.2(a) and 3.2(b) we can see a graphical representation of the two rules. In Figure 3.3 we give an intuition of the rewriting procedure that allow us to pass from rules to reactions. This will be explained deeper in the next paragraphs.

3.2 Syntax

In Kappa, proteins are typically represented by agents. The properties of a given agent are expressed by some sites that may carry an internal state. Moreover, potential links between proteins are specified by bonds between agent sites.

3.2.1 Signature

Now let us formally introduce the syntax of Kappa.

We assume that we are given:

1. a finite set of agents types $\mathcal{A}$ (representing different kinds of proteins);

2. a finite set of sites $\mathcal{S}$ (corresponding to protein domains);

$$
\begin{array}{rcll}
a & ::= & N_l(\sigma) & \text{(agent)} \\
s & ::= & n_\iota^\lambda & \text{(site)} \\
N & ::= & A \in \mathcal{A} & \text{(agent type)} \\
n & ::= & x \in \mathcal{S} & \text{(site name)} \\
l & ::= & i \in \mathbb{N} \mid \bar{i} \in \bar{\mathbb{N}} & \text{(agent identifier)} \\
\lambda & ::= & \epsilon \mid N_l@n \mid N@n \mid - \mid ? & \text{(binding state)} \\
\sigma & ::= & \epsilon \mid s, \sigma & \text{(interface)} \\
\iota & ::= & \epsilon \mid \mathtt{w} \in \mathbb{I} & \text{(internal state)}
\end{array}
$$

Figure 3.4: Syntax for agents.

3. a finite set $\mathbb{I}$ of non empty strings (corresponding to internal states);

4. two signature maps Σ_ι and Σ_λ, from $\mathcal{A}$ to $\mathcal{P}(\mathcal{S})$ assigning a set of sites to each agent type.

Intuitively, $\Sigma_\iota(A)$ is the set of sites which can bear a modifiable internal state $w \in \mathbb{I}$ (such as a level of energy), whereas $\Sigma_\lambda(A)$ is the set of sites which can be bound to some other sites. We also denote by Σ the signature map that associates to each agent type $A \in \mathcal{A}$ the combined interface $\Sigma_\iota(A) \cup \Sigma_\lambda(A)$.

3.2.2 Patterns

The full syntax for agents is given by the grammar in Figure 3.4.

An *agent identifier* l belongs to the set $\mathbb{N}$ of natural numbers, or to a copy $\overline{\mathbb{N}}$ of the set of natural numbers. Most agents will be identified by numbers in $\mathbb{N}$, whereas identifiers in $\overline{\mathbb{N}}$ will be used temporary when agents are created, before that a proper identifier is allocated.

An *interface* σ is a list of sites with internal states (denoted as subscripts) and binding states (denoted as superscripts). For every agent A, the set of its internal sites is given by $\Sigma_\iota(A)$, whereas the set of its binding sites is given by $\Sigma_\lambda(A)$. A site x that has an internal state $\mathtt{w} \in \mathbb{I}$ is written as $x_{\mathtt{w}}$.

When a site is written as x_ϵ it means that it does not have an internal state. This can happen:

1. if $x \in \Sigma(A) \backslash \Sigma_\iota(A)$,

2. or if the internal state is not specified.

We usually omit the symbol 'ϵ' in examples.

If a site x is in $\Sigma_\lambda(A)$, x can be free or bound to some other site. There are several levels of knowledge about the binding states and also several levels of information about the bonds. More specifically:

1. The symbol 'ϵ' means that the site is free.
2. The question mark '?' is used when there is no information about the binding site (it can be free or bound).
3. A *binding type* $B@y$ is used to denote that the site is linked to the site y of some agent B.
4. A site address $B_l@y$ means that the site is linked to the site y of the agent B with identifier l.
5. A *wildcard bond* '$-$' is used to denote a site that is bound but there is no information about its pattern.

As in the case of internal states, the symbol 'ϵ' is generally omitted. Sometimes we refer to a site as a triple (A, i, x) in $\mathcal{A} \times \mathbb{N} \times \mathcal{S}$.

Definition 3.2.1 (Agent)**.** *An agent is defined by a type A in $\mathcal{A}$, an identifier l and an interface σ. Such an agent is denoted by $A_l(\sigma)$.*

Definition 3.2.2 (Expression)**.** *An expression E is a sequence of agents such that:*

1. *each agent is different, by the type or by the identifier, from all the others;*
2. *for every interface σ, each site appears at most once in the interface σ;*
3. *each site name occurring in the interface of the agent A occurs in $\Sigma(A)$;*
4. *each site name which occurs in the interface of the agent A with an internal state distinct from 'ϵ' occurs in $\Sigma_\iota(A)$;*
5. *each site name which occurs in the interface of the agent A with a binding state distinct from 'ϵ' occurs in $\Sigma_\lambda(A)$.*

Given an expression E and an agent A, we define by $\mathbf{agents}(E, A)$ the set of identifiers l such that there exists an agent A in E with identifier l.

Definition 3.2.3 (Structural equivalence)**.** *The structural equivalence is defined as the smallest equivalence relation on expressions such that:*

1. $E, A_l(\sigma, s, s', \sigma'), E' \equiv E, A_l(\sigma, s', s, \sigma'), E'$

2. $E, a, a', E' \equiv E, a', a, E'$

This equivalence states that the order of sites inside interfaces and the order of agents inside expressions do not matter.

Definition 3.2.4 (Pattern)**.** *A pattern is an expression E such that if there is an agent A_l that holds a site x wich carries a bound $A'_{l'}@x'$, then there is an agent in E of type A' with identifier l', that exhibits a site x' with a bound $A_l@x$.*

So a pattern has no dangling reference. A pattern is said to be *proper* if it has only proper agent identifiers (in $\mathbb{N}$).

A pattern is *disconnected* if $E \equiv E', E''$ where E' and E'' are both not empty patterns. Otherwise, it is *connected*. A disconnected pattern can be decomposed in a set of connected patterns, called its *connected components*.

Definition 3.2.5 (Pattern component)**.** *A pattern component is a connected pattern.*

Definition 3.2.6 (Mixture)**.** *A mixture M is a proper pattern that is fully specified, i.e. such that for every agent A in M the interface of A is fully documented (that is to say it describes the binding state of each site in $\Sigma_\lambda(A)$ and the internal state of each site in $\Sigma_\iota(A)$).*

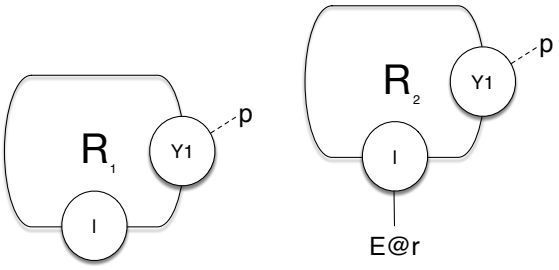
Definition 3.2.7 (Species)**.** *A species is a non-empty connected mixture or, equivalently, a fully specified non-empty pattern component.*

In Figure 3.5 we can see a graphical representation of two agents, respectively $\mathtt{R_1(Y1_p, l)}$ and $\mathtt{R_2(Y1_p, l^{E@r})}$. Both agents are of type R with two sites: $Y1$, that expresses an internal state, and l, that expresses a binding state. The agent of Figure 3.5(a) has the internal state of $Y1$ phosphorylated (this is denoted by p) whereas the state of the site l is free. The agent of Figure 3.5(b) has the internal state of $Y1$ phosphorylated as well, but the site l is linked to the site r of some agent E (this is denoted by the binding type $E@r$).

In Figure 3.6 we can see a species composed by two agents, R_1 and E_1, that are connected by a bond (there is a link between the site l of the agent R_1 and the site r of the agent E_1).

3.2.3 Rules

The dynamic of a system is described by the means of rules. A rule is given by a pair of patterns (E_ℓ, E_r) and a rate constant k (which is a non negative real number), and is usually written as $E_\ell \xrightarrow{k} E_r$ (or, equivalently, $E_\ell \to E_r \ @k$).



(a) $\mathtt{R_1(Y1_p, l)}$ (b) $\mathtt{R_2(Y1_p, l^{E@r})}$

Figure 3.5: Two agents.

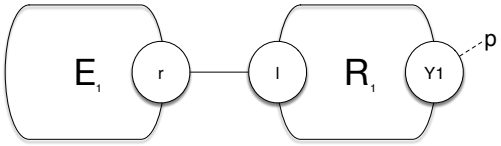


Figure 3.6: A species.

The left hand part (lhs), E_ℓ, describes the context of application for a rule. The right hand part (rhs), E_r, describes the transformations to be performed.

Definition 3.2.8. *(Constraints on rules) Given a rule $E_\ell \xrightarrow{k} E_r$, it has to be possible to obtain E_r from E_ℓ in the following way (the order matters):*

1. creation: *some agents $A_{\bar{i}}(\sigma)$ with their full interfaces $\Sigma(A)$, with all sites free and all sites $s \in \Sigma_\iota$ defined, are added ($\bar{i} \in \bar{\mathbb{N}}$);*
2. unbinding: *some occurences of the wildcard '$-$', some binding type and some site addresses $A_i@n$ are removed;*
3. deletion: *some agents are removed;*
4. modification: *some (non empty) internal states are replaced with other (non empty) internal states;*
5. binding: *some free sites are bound pairwise by using appropriate site addresses.*

Between the agents that are neither removed, nor created, there is a bijective correspondence assured by agent types and identifiers. Any two proper agents in correspondence have the same type and their interfaces document the same sites. Note that, according to Definition 3.2.8, binding types and wildcards can be removed but not created.

Rules with side effects.

We say that a system of rules has side effects if it is possible to modify the state of sites that are not documented in the left hand side of the rules. More precisely, it may happen only in the two following cases:

1. when we remove an agent without checking the binding state of all its sites;

2. when we remove a bond denoted by a binding type or the symbol '−'.

In the first case a side effect is possible (may semantics), in the latter one a side effect is sure (must semantics).

Definition 3.2.9 (May set)**.** *Given a rule r we define $\mathcal{MAY}(r)$ as the set of sites in r that may raise side effects when the rule is applied.*

Definition 3.2.10 (Must set)**.** *Given a rule r we define $\mathcal{MUST}(r)$ as the set of sites in r that raise side effects whenever the rule is applied.*

Example 3.2.11. *Let us consider the following rule r_1:*

$$\mathtt{A}_1() \xrightarrow{\mathtt{k}} \varnothing$$

that is deleting an agent of type A *without any check. If we apply the rule to a mixture of the form:*

$$\mathtt{A}_1(\mathtt{b}),$$

it happens that all the agents in the mixture are documented in the rule. Instead, if we apply the rule to the following mixture:

$$\mathtt{A}_1(\mathtt{b}^{\mathtt{B}_1@\mathtt{a}}), \mathtt{B}_1(\mathtt{a}^{\mathtt{A}_1@\mathtt{b}}),$$

it happens that the agent $\mathtt{B}_1$ *is modified even if it is not documented in the rule. So $\mathcal{MAY}(r_1)$ is equal to the site* b *of the agent* A*.*

Let us consider now the following rule r_2:

$$\mathtt{A}_1(\mathtt{b}^{\mathtt{B}@\mathtt{a}}) \xrightarrow{\mathtt{k}} \varnothing$$

that is deleting an agent of type A *with a link to the site* a *of some agents* B*. In this case, even if it is not directly documented by the rule, an agent of type* B *is modified every time that the rule is applied. So $\mathcal{MUST}(r_2)$ is equal to the site* b *of the agent* A*.*

3.3 Semantics

Now we describe the semantics of Kappa. Firstly, we formalise the effect of applying a rule on a pattern. We use this to define the set of (ground) reactions that are induced by a set of rules in Kappa. Then, we introduce the differential semantics of a Kappa model, as the solution of the set of differential equations that are obtained by applying the principle of mass action over the underlying set of reactions.

3.3.1 Operational semantics

Now we will describe how to apply a rule r, $E_\ell \xrightarrow{k} E_r$, to a mixture E. First of all, it is necessary that the context expressed by the lhs is present in E. Technically speaking, we will say that E_ℓ embeds into E.

Example 3.3.1. *To have an intuition, we can have a look to Figure 3.7 and observe that the pattern described on the left matches, in some intuitive way, the species described on the right. There we have (on the left) an agent of type R that documents two sites: $Y1$ that is specified to be unphosphorylated and l that shows a bond to an agent E by some site s. So we can imagine that the agent R on the left can be matched with the agent R on the right. This one exhibits exactly the same two sites, $Y1$ and l.*

The state of $Y1$ is the same (u) both for the pattern on the left and for the species on the right, so they match. The binding state, denoted by l, corresponds to a binding type in the pattern and to an explicit link in the species. Since the binding type of the pattern denotes a bond to an agent E through the site r whereas the species exhibits an explicit link to the agent E_1 by the site r, they match as well.

Let us describe those concepts formally.

Definition 3.3.2 (Substitution)**.** *A* substitution ϕ *is a partial mapping from pairs agent/identifier (A, l) (in $\mathcal{A} \times (\mathbb{N} \cup \bar{\mathbb{N}})$) to agent identifiers (in $\mathbb{N} \cup \bar{\mathbb{N}}$).*

Clearly, given a pattern E, a substitution ϕ can be applied only if $(A, l) \in \mathbf{dom}(\phi)$ for any $l \in \mathbf{agents}(E, A)$.

The purpose of a substitution ϕ is to replace the agent identifier l with $\phi(A, l)$ (in the agent).

Definition 3.3.3 (Injective substitution)**.** *A substitution ϕ is injective if, and only if, for any agent type A and for any two identifiers l, l', we have $\phi(A, l) = \phi(A, l') \Rightarrow l = l'$.*

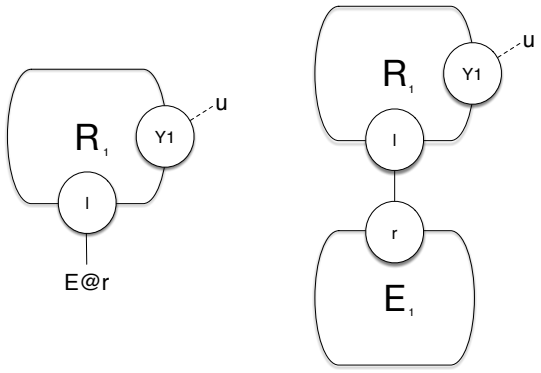


Figure 3.7: Example of a pattern that can be embedded in a species.

$$
\begin{array}{rcl}
\iota = \iota_\ell, \epsilon & \Longrightarrow & \iota \models \iota_\ell \\
\lambda = \lambda_\ell & \Longrightarrow & \lambda \models \lambda_\ell \\
N_l@n & \models & N@n \\
N_l@n & \models & - \\
\lambda & \models & ? \\
\iota \models \iota_\ell \wedge \lambda \models \lambda_\ell & \models & n_\iota^\lambda \models n_{\iota_\ell}^{\lambda_\ell} \\
\sigma & \models & \epsilon \\
s \models s_\ell \wedge \sigma \models \sigma_\ell & \Longrightarrow & s, \sigma \models s_\ell, \sigma_\ell \\
\sigma \models \sigma_\ell & \Longrightarrow & N_l(\sigma) \models N_l(\sigma_\ell)
\end{array}
$$

Figure 3.8: Agent matching.

An injective substitution is a good candidate to identify the agents of two patterns. More precisely, each agent $A_\ell(\sigma_\ell)$ in the first pattern can be identified with the agent $A_l(\sigma)$ if:

1. agents identifiers are the same, i.e. $l = \phi(A, \ell)$;

2. the signature σ contains more information than the signature σ_ℓ.

This second property is formalized by the matching relation $\models$ given in Figure 3.8.

Yet, since interfaces are defined up to permutations of sites, one may have to reorder the sites before applying the matching relation. This is possible thanks to the structural equivalence defined in Definition 3.2.3.

Now we are ready to define an embedding between two patterns.

Definition 3.3.4 (Embedding)**.** *An* embedding ϕ *between two patterns* E_ℓ *and* E *is an injective substitution such that:*

1. $\mathbf{dom}(\phi) = \{(A, l) \mid A \in \mathcal{A}, l \in \mathbf{agents}(E_\ell, A)\}$,

2. *for any* $(A, l) \in \mathbf{dom}(\phi)$, *there exists an agent* a' *such that* $a \equiv a'$ *and* $a' \models \bar{\phi}(a_\ell)$, *where* a_ℓ *is the unique agent in* E_ℓ *of type* A *with identifier* l *and* a *is the unique agent in* E *of type* A *with identifier* $\phi(A, l)$.

Example 3.3.5 (continued)**.** *In our example (eg. see Figure 3.5), the embedding between the pattern* $\mathtt{R_1(Y1_p, l)}$ *on the left, and the pattern* $\mathtt{R_2(Y1_p, l^{E@r})}$ *on the right is induced by the injective substitution that maps the pair* $(R, 1)$ *to the agent identifier* 2.

Definition 3.3.6 (Clean substitution)**.** *A* clean substitution *is a substitution such that for any couple* $(A, \bar{l}) \in (\mathcal{A} \times \bar{\mathbb{N}}) \cap dom(\phi)$, *we have:*

$$\phi(A, \bar{l}) = \bar{l}.$$

Definition 3.3.7 (Clean embedding)**.** *An embedding is* clean *if it is a clean substitution.*

Definition 3.3.8 (Isomorphic embedding)**.** *Whenever there exist an embedding* ϕ *between* E_ℓ *and* E *and an embedding* ϕ' *between* E *and* E_ℓ *we say that* ϕ *is an* isomorphic embedding.

We notice that embeddings between two species are isomorphic embeddings.

Generally, given two pattern components C_1 and C_2 there can be several embeddings of the former into the latter. We will write $[C_1, C_2]$ for the set of such embeddings. In usual graph theory, the number of embeddings between two connected graphs may be exponential with respect to the size of these graphs. In Kappa, the number is at worst linear due to the fact that sites are uniquely named. We call this the rigidity principle.

Lemma 3.3.9 (rigidity)**.** *An embedding of a pattern component* C_1 *into a pattern component* C_2 *is fully defined by the image of one agent. That is to say, whenever there are* i, ϕ, *and* ϕ' *such that* $\phi \in [C_1, C_2]$, $\phi' \in [C_1, C_2]$ *and* $\phi(i) = \phi'(i)$, *then* $\phi = \phi'$.

Proof. Given two agents, n and n' we say that there is a basic path from A_0 to A_1 if there exist a site x belonging to n and a site x' belonging to n that are bound together. We denote it as $n.x \frown x'.n'$.

Given two agents, n and n' we say that there is a path from n to n', if there exists a finite sequence of agents A_i such that $A_0 = n$ and for any i it exists a basic path from A_i to A_{i+1}.

We denote a path between n and n' as a sequence

$$n.x \frown (x_i.A_{i+1}.x_{i+1} \frown)^\star x'.n'$$

$$\begin{array}{rcl} \lambda = \epsilon, N@n, -, ? & \Longrightarrow & \bar{\phi}(\lambda) = \lambda \\ \bar{\phi}(N_l@n) & = & N_{\phi(N,l)}@n \\ \bar{\phi}(n_\iota^\lambda) & = & n_\iota^{\bar{\phi}(\lambda)} \\ \bar{\phi}(s, \sigma) & = & \bar{\phi}(s), \bar{\phi}(\sigma) \\ \bar{\phi}(N_l(\sigma)) & = & N_{\phi(N,l)}(\bar{\phi}(\sigma)) \end{array}$$

Figure 3.9: Agent substitution.

where n is the starting agent, x is the site of n that is linked to the successor, n' is the final agent and x' is its entry site. Between n and n' there could be a possibly empty list of agents denoted as $x_i.n_i.x_{i+1}$, where n_i is the agent, x_i is the entry site and x_{i+1} is the exit site.

Now, let C_1 and C_2 be two pattern components and ϕ, ϕ' be two different embeddings in $[C_1, C_2]$.

Let us suppose it exists $n \in \mathcal{A}_{C_1}$ such that $\phi(n) = \phi'(n)$. If there is just one agent in C_1 then $\phi = \phi'$ and we have done. Otherwise, since C_1 is connected, it exists an agent n' in C_1 such that there is a path from n to n' in C_1.

Let $p = n.x \frown x_1.n_1.x_2 \frown \cdots \frown x'.n'$ be this path. So the image of p by ϕ is in C_1.

We know that $\phi(n) = \phi'(n)$, so $\phi(n).x = \phi'(n).x$ and $\phi(n_1) = \phi'(n_1)$. Since this is true for all the steps in the path, we have $\phi = \phi'$. □

In general an epimorphism (epi) is an embedding $\phi \in [C_1, C_2]$ such that, for all $\phi_1 \in [C_2, C_3]$, $\phi_2 \in [C_2, C_3]$, $\phi_1\phi = \phi_2\phi$ implies $\phi_1 = \phi_2$.

Lemma 3.3.10. *An embedding between two non-empty connected components is necessarily an epimorphism. That is to say, if C is a pattern component, $\phi \in [C', C]$ is an epi if and only if C' is not empty.*

Proof. We have to prove that if $\phi_1 \circ \phi = \phi_2 \circ \phi$, then $\phi_1 = \phi_2$.

Let C be a pattern component, C' be non-empty and $\phi \in [C', C]$. Let C'' be another pattern component and $\phi_1, \phi_2 \in [C, C'']$. Since C' is not empty we can consider $n \in C'$ such that $\phi_1(\phi(n)) = \phi_2(\phi(n))$.

Let $n' = \phi(n)$. We have $\phi_1(n') = \phi_2(n')$. Since C'' is a pattern component we can apply Lemma 3.3.9, so $\phi_1 = \phi_2$.

□

Corollary 3.3.11. *$\phi \in [Z_1, Z_2]$ is an epi if and only if the image of Z_1 intersects each component of Z_2.*

$$
\begin{array}{rcl}
\iota[\epsilon] = \iota & & \iota[\mathtt{w}_r] = \mathtt{w}_r \\
\lambda[\epsilon] = \epsilon & & \lambda[N_l@n] = N_l@n \\
\lambda[N@n] & = & \lambda \\
\lambda[-] & = & \lambda \\
\lambda[?] & = & \lambda \\
n_\iota^\lambda[n_{\iota_r}^{\lambda_r}] & = & n_{\iota[\iota_r]}^{\lambda[\lambda_r]} \\
\sigma[\epsilon] & = & \sigma \\
(s,\sigma)[s_r,\sigma_r] & = & s[s_r],\sigma[\sigma_r] \\
N_l(\sigma)[N_l(\sigma_r)] & = & N_l(\sigma[\sigma_r])
\end{array}
$$

Figure 3.10: Agent replacement.

Now, let E be a pattern, let $r := E_\ell \xrightarrow{k} E_r$ be a rule and let ϕ be an embedding between E_ℓ and E. Since for any agent A we have $\mathbf{agents}(E_\ell, A) \subseteq \mathbb{N}$, ϕ is clean. We are ready to define the result of the application of the rule r to the pattern E.

In order to do this, we need to consider three different kinds of agents:

1. the *agents $A_{l_\ell}(\sigma_\ell)$ that are preserved by the rule*: $A_{l_\ell}(\sigma_\ell)$ appears in E_ℓ and there exists an interface σ_r such that $A_{l_\ell}(\sigma_r)$ appears in E_r;

2. the *agents $A_{l_r}(\sigma_r)$ that are created by the rule*: $A_{l_r}(\sigma_r)$ appears in E_r but there is no agent of type A with identifier l_r that appears in E_ℓ;

3. the *agents $A_{l_\ell}(\sigma_\ell)$ that are deleted by the rule*: $A_{l_\ell}(\sigma_\ell)$ appears in E_ℓ but there is no agent of type A with identifier l_ℓ that appears in E_r.

We need to extend the embedding ϕ in order to be able to deal with newly created agents. Thus we define the clean injective substitution $\phi^\star$ over $\mathbf{dom}(\phi) \cup (\mathcal{A} \times \bar{\mathbb{N}})$ by $\phi^\star(A,i) = \phi(A,i)$ for any $(A,i) \in \mathbf{dom}(\phi)$ and $\phi^\star(A,\bar{i}) = \bar{i}$ when $\bar{i} \in \mathbf{agents}(E_r, A) \cap \bar{\mathbb{N}}$ (this way, $\phi^\star$ preserves temporary identifiers).

Then:

1. for any agent $A_{l_\ell}(\sigma_\ell)$ that is removed, the agent of type A with identifier $\phi(A, l_\ell)$ is removed in E;

2. for any agent $A_r(\sigma_r)$ that is created, the agent $\phi^\star(A_r(\sigma_r))$ is added in E;

3. and for any agent $A_{l_\ell}(\sigma_\ell)$ that is preserved:

(a) we denote by $A_r(\sigma_r)$ and $A(\sigma)$ the agents in E_r and E which have the same type and the same identifier as the agent $A_\ell(\sigma_\ell)$;

(b) then we select an agent $A'(\sigma')$ (the choice does not matter) such that $A(\sigma) \equiv A'(\sigma')$ and $A'(\sigma') \models \bar{\phi}(A_\ell(\sigma_\ell))$;

(c) then the agent $A(\sigma)$ is replaced with agent $A'(\sigma')[\bar{\phi}^\star(A_r(\sigma_r))]$, where .[.] is the replacement function defined in Figure 3.10.

We denote by $E[E_r]_\phi$ the obtained expression (which is well defined up to $\equiv$-equivalence). One shall notice that $E[E_r]_\phi$ might not be a pattern, because there might be some pending bonds, which are sites with a binding state of the form $A_l@x$ such that:

1. either the agent of type A and identifier l has been removed;

2. or the site x of the agent A and identifier l has been made free.

In order to remove pending bonds, we introduce a function **clean** such that **clean**(E) is obtained by replacing with the symbol ϵ each site address $A_l@x$ such that:

1. there is no agent A of identifier l in E;

2. or the site x of the agent type A with identifier l is free.

When we apply a rule to a mixture, we expect that the result of the application of the rule will be a mixture as well. Anyway, we notice that **clean**$(E[E_r]_\phi)$ might not be a mixture because of temporary identifiers. To allocate fresh proper identifiers for the newly created agents, we introduce a function **fresh** between patterns. **fresh**(E) is obtained by replacing any temporary identifier $\bar{i}$ of the agent A by $M(A) + i + 1$, where $M(A)$ is the maximum element of the set $\{0\} \cup \{\mathbb{N} \cap \mathbf{agents}(E, A)\}$.

Now we can define the operational semantics as a labeled transition system.

The states of the system are mixtures (up to $\equiv$). We shall notice that the impact of applying a rule $E_\ell \xrightarrow{k} E_r$ on a mixture is fully defined (up to $\equiv$) by the clean embedding ϕ between the lhs E_ℓ of the rule and the mixture E. So, we define the set $\mathbb{L}$ of labels as the set of tuples (r, E, ϕ), where:

1. r is a rule,

2. E is a state, and

3. ϕ is an embedding between the left hand part of the rule and E.

In such a case we write:

$$E \xrightarrow{(r,E,\phi)} \mathbf{fresh}(\mathbf{clean}(E[E_r]_\phi))$$

Example 3.3.12. *Let us consider the following example.*

We consider the following signature:

$$\begin{array}{rcl} \mathcal{A} & = & \{\mathtt{A}\} \\ \mathbb{I} & = & \varnothing \\ \mathcal{S} & = & \{\mathtt{a},\mathtt{b}\} \\ \Sigma_\lambda(\mathtt{A}) & = & \{\mathtt{a},\mathtt{b}\} \\ \Sigma_\iota(\mathtt{A}) & = & \varnothing, \end{array}$$

the following mixture:

$$\mathtt{A}_1(\mathtt{a}^{\mathtt{A}_2@\mathtt{b}},\mathtt{b}^{\mathtt{A}_3@\mathtt{a}}),\mathtt{A}_2(\mathtt{a},\mathtt{b}^{\mathtt{A}_1@\mathtt{a}}),\mathtt{A}_3(\mathtt{a}^{\mathtt{A}_1@\mathtt{b}},\mathtt{b})$$

and the following rule:

$$\mathtt{A}_1(\mathtt{a}) \xrightarrow{\mathtt{k}} \mathtt{A}_{\bar{0}}(\mathtt{a},\mathtt{b}).$$

Intuitively, this rule can be applied with an agent of type A*, that has the site* a *which is free (and without considering the state of the site* b*). Applying the rule consists in removing the agent* $\mathtt{A}_1(\mathtt{a})$ *before creating a new agent* $\mathtt{A}_0(\mathtt{a},\mathtt{b})$ *of type* A *with both sites* a *and* b *free (no site is preserved).*

There exists only one embedding between the lhs $\mathtt{A}_1(\mathtt{a})$ *of the rule and the mixture* E*, that is:*

$$\phi = [\mathtt{A}, 1 \to 2].$$

$E[E_\ell]_\phi$ *is equal to the expression*

$$\mathtt{A}_1(\mathtt{a}^{\mathtt{A}_2@\mathtt{b}},\mathtt{b}^{\mathtt{A}_3@\mathtt{a}}),\mathtt{A}_{\bar{0}}(\mathtt{a},\mathtt{b}),\mathtt{A}_3(\mathtt{a}^{\mathtt{A}_1@\mathtt{b}},\mathtt{b}).$$

This expression has a dangling bond on the site a *of the agent* $\mathtt{A}_1$*:*

$$\mathtt{A}_1(\mathtt{a}^{\mathtt{A}_2@\mathtt{b}},\mathtt{b}^{\mathtt{A}_3@\mathtt{a}}),$$

so the primitive ***clean*** *is applied to remove it.*

Indeed the result of:

$$\mathbf{clean}(E[E_r]_\phi)$$

is equal to the expression:

$$\mathtt{A}_1(\mathtt{a},\mathtt{b}^{\mathtt{A}_3@\mathtt{a}}),\mathtt{A}_{\bar{0}}(\mathtt{a},\mathtt{b}),\mathtt{A}_3(\mathtt{a}^{\mathtt{A}_1@\mathtt{b}},\mathtt{b}).$$

Then the temporary identifier $\bar{0}$ *is replaced with* 4 *by the primitive* ***fresh*** *and the result of:*

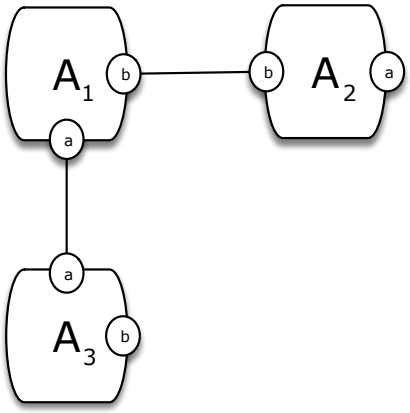


Figure 3.11: Mixture E of the example.

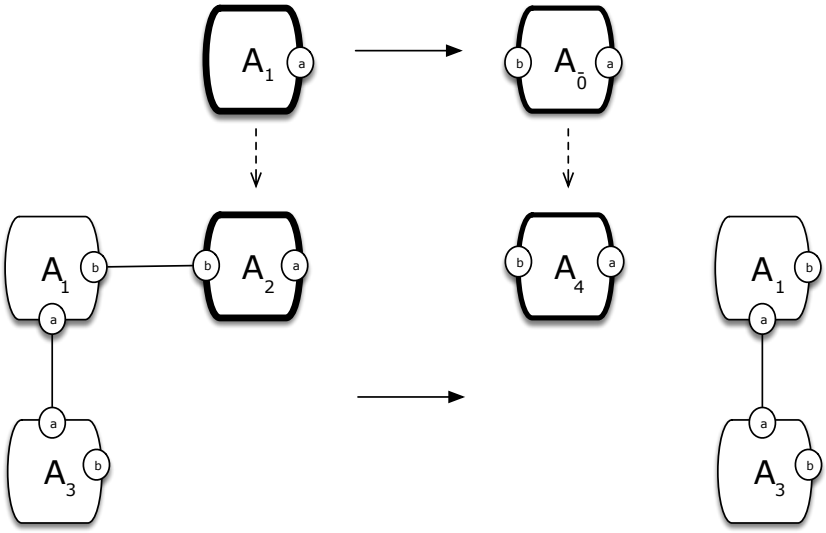


Figure 3.12: Application of the rule to the mixture E.

$$\mathbf{fresh}(\mathbf{clean}(E[E_r]_\phi))$$

is equal to:

$$\mathtt{A}_1(\mathtt{a}^{\mathtt{A}_2@\mathtt{b}}, \mathtt{b}^{\mathtt{A}_3@\mathtt{a}}), \mathtt{A}_4(\mathtt{a}, \mathtt{b}), \mathtt{A}_3(\mathtt{a}^{\mathtt{A}_1@\mathtt{b}}, \mathtt{b}),$$

as depicted in Figure 3.12.

3.3.2 Differential semantics

In order to define the concrete differential semantics for Kappa, we need to consider a set of rules $\mathcal{R}$ and a finite set of species $\mathcal{V}$ such that:

1. $\mathcal{V}$ is closed under application of the rules in $\mathcal{R}$,
2. $\mathcal{V}$ has at most one representative for each species isomorphism class.

More formally:

1. for any mixture E, any pattern component which is isomorphic to an element in $\mathcal{V}$, any rule $E_\ell \to E_r$, and any embedding ϕ between E_ℓ and E, each pattern component in $\mathbf{clean}(E[E_r]_\phi)$ is isomorphic to an element in $\mathcal{V}$;
2. for any pair (v, v') of elements in $\mathcal{V}$, if there exists an embedding between v and v', then $v = v'$.

In order to ensure the finiteness of the set $\mathcal{V}$, we assume that species contain no cycle. Formally, for any species S, we define the non-oriented graph $G(S)$, by the following constraints:

1. the nodes of $G(S)$ are the agents of the species S;
2. there is an edge between to nodes (A, i) and (A', i') in $G(S)$ if and only if, there exists two site names x and x' such that there is a link between the site (A, i, x) and (A', i', x') in the species S.

For any species S, we assume that:

1. for every two agents (A, i) and (A', i'), there is at most one link in the species S, between the sites of the agent (A, i) and the sites of the agent (A', i');
2. the graph $G(S)$ is acyclic.

The states ρ of the system are mappings from chemical species $v \in \mathcal{V}$ to real numbers in $\mathbb{R}^+$ ($\rho(v)$ denotes the concentration of the species v). So as to define the function $\mathbb{F}$ which specifies the behavior of the system, we consider the set of chemical reactions which are generated by the set of rules $\mathcal{R}$.

Given a rule $r := E_\ell \xrightarrow{k} E_r$ in $\mathcal{R}$, we may assume without any loss of generality that E_ℓ is written as $C_1, \ldots, C_k$ where each C_i is a pattern component.

A reaction is obtained, by choosing for any integer i between 1 and k, a reachable species $v_i \in \mathcal{V}$ and an embedding ϕ_i between C_i and v_i. The expression $v_1, \ldots, v_k$ might not be a mixture because distinct species may share some agent identifiers. In order to define the product of a reaction, we choose k species $w_1, \ldots, w_k$ and k embedding $\psi_1, \ldots, \psi_k$ such that $w_1, \ldots, w_k$ is a mixture, and that for any i between 1 and k, ψ_i is an embedding between v_i and w_i. This way, we form a composite embedding $\phi = \sum_i \psi_i \circ \phi_i$ between E_ℓ and $w_1, \ldots, w_k$. The result of the application of the rule r on $w_1, \ldots, w_k$ along ϕ is isomorphic to a tuple of species in $\mathcal{V}$ that we denote by $p_1, \ldots, p_l$ (we can check that $p_1, \ldots, p_l$ does not depend on the choice of the $w_1, \ldots, w_k$).

Then the function $\mathbb{F}$ is obtained by summing up the contribution of each reaction. The consumption of the reactants will give the following negative contribution:

$$\mathbb{F}(\rho)(v_j) \stackrel{-}{=} \gamma \prod_i (\rho(v_i) \mid 1 \leqslant i \leqslant k).$$

whereas the positive contribution is given by the production of species in $\mathcal{V}$:

$$\mathbb{F}(\rho)(p_{j'}) \stackrel{+}{=} \gamma \prod_i (\rho(v_i) \mid 1 \leqslant i \leqslant k).$$

Where:

1. γ is the quotient between k and the number of embeddings from E_ℓ to itself,
2. j ranges between 1 and k, and
3. j' ranges between 1 and l.

The obtained autonomous system $(\mathcal{V}, \mathbb{F})$ is *positive*. Indeed, for any species $v \in \mathcal{V}$, $\mathbb{F}(v)$ can be written as:

$$-\rho(v) \cdot P[\rho(v_1), \ldots, \rho(v_m)] + Q[\rho(v'_1), \ldots, \rho(v'_n)],$$

where P and Q are two polynomial mappings with positive coefficients and $v_1, \ldots, v_m, v'_1, \ldots, v'_n$ are $m + n$ species in $\mathcal{V}$.

3.4 Conclusion

We have introduced the syntax and the semantics of Kappa. We have seen that Kappa offers a compact description of reaction networks. Indeed each rule denotes a symbolic representation of a set of reactions, that applies the same transformation in different contexts of application.

We have introduced the ODEs semantics of a set of rules as the ODEs system that is obtained by applying the principle of mass action law to the underlying set of reactions. This is a mathematical definition that is mandatory to state the soundness of our approach. We will never compute the underlying set of reactions of a Kappa model explicitly.

Chapter 4

Symmetries

In this chapter we investigate a class of sites with specific properties of symmetry. Given two sites x and y in an agent A and a set of rules $\mathcal{R}$, we will say that they are symmetric if they play the same role respect to the set of rules $\mathcal{R}$.

After providing some intuitions over well-chosen examples, we will formalise the notion of "playing the same role". Then, we show that the equivalence relation, that identifies the species that can be obtained by swapping symmetric sites, induces a bisimulation over the semantics of $\mathcal{R}$, hence a model reduction.

4.1 Case studies

We start with a series of examples to illustrate how to detect when some sites have the same capability of interaction. This information will be used to transform the original system into a more compact one.

4.1.1 First case study

In our first example, we consider one kind of agent, A. We suppose that an agent A has two sites:

1. a site x,
2. a site y.

The state of a protein A is denoted as an ordered couple of symbols among $\{u, p\}$. The symbol u indicates an unphosphorylated site, whereas the symbol p indicates a phosphorylated one. For example, a protein A having both sites x and y phosphorylated is denoted as $A[p, p]$.

$A[u,u] \xrightarrow{k_1} A[p,u]$
$A[u,p] \xrightarrow{k_1} A[p,p]$

(a) Phosphorylation of the first site.

$A[u,u] \xrightarrow{k_1} A[u,p]$
$A[p,u] \xrightarrow{k_1} A[p,p]$

(b) Phosphorylation of the second site.

$A[p,p] \xrightarrow{k_2}$

(c) Degradation.

Figure 4.1: Chemical reactions for first case study.

$A[u,u] \xrightarrow{2 \cdot k_1} A[p,u]$

(a) Phosphorylation of the first site.

$A[p,u] \xrightarrow{k_1} A[p,p]$

(b) Phosphorylation of the second site.

$A[p,p] \xrightarrow{k_2}$

(c) Degradation.

Figure 4.2: Simplified chemical reactions for first case study.

We start by considering the set of reactions given in Figure 4.1. The sites x and y of the protein A can all be phosphorylated at the same rate constant k_1. Moreover, we assume that a protein A that has both sites phosphorylated can be degraded with a rate constant k_2.

This model can be encoded in Kappa thanks to the following set of rules:

1. $\mathtt{A_1(x_u)} \rightarrow \mathtt{A_1(x_p)}$ @ $\mathtt{k_1}$
2. $\mathtt{A_1(y_u)} \rightarrow \mathtt{A_1(y_p)}$ @ $\mathtt{k_1}$
3. $\mathtt{A_1(x_p, y_p)} \rightarrow$ @ $\mathtt{k_2}$.

The first spontaneous observation is that x and y have exactly the same activation rate.

Now we wonder if x and y play an identical role in each instance of the protein A. So, for each rule, we propose to replace the occurrence of each site with all the combinations among x and y, while dividing the kinetic rates by the number of combinations (in all the set). By performing this change, we need to take care of gain and loss of automorphisms.

In our specific case the site x in the first rule will be substituted twice, once with x and once with y. Since there are two possible combinations the rate constant k_1 will be divided by 2.

From our set of rules we obtain the following substitutions:

1. $\mathtt{A_1(x_u)} \rightarrow \mathtt{A_1(x_p)}$ @ $\mathtt{k_1}$

(a) $\mathtt{A_1(x_u)} \rightarrow \mathtt{A_1(x_p)}$ @ $\frac{\mathtt{k_1}}{2}$ (by changing nothing)

(b) $\mathtt{A_1(y_u)} \rightarrow \mathtt{A_1(y_p)}$ @ $\frac{\mathtt{k_1}}{2}$ (by replacing x with y)

2. $\mathtt{A(y_u)} \rightarrow \mathtt{A(y_p)}$ @ $\mathtt{k_1}$

(a) $\mathtt{A(x_u)} \rightarrow \mathtt{A(x_p)}$ @ $\frac{\mathtt{k_1}}{2}$ (by replacing y with x)

(b) $\mathtt{A(y_u)} \rightarrow \mathtt{A(y_p)}$ @ $\frac{\mathtt{k_1}}{2}$ (by changing nothing)

3. $\mathtt{A_1(x_p, y_p)} \rightarrow$ @ $\mathtt{k_2}$

(a) $\mathtt{A_1(x_p, y_p)} \rightarrow$ @ $\frac{\mathtt{k_2}}{2}$ (by changing nothing)

(b) $\mathtt{A_1(y_p, x_p)} \rightarrow$ @ $\frac{\mathtt{k_2}}{2}$ (by replacing x with y and y with x).

By summing up the rules that are syntactically equivalent, we obtain the following set:

1. $\mathtt{A_1(x_u)} \rightarrow \mathtt{A_1(x_p)}$ @ $\frac{\mathtt{k_1}}{2} + \frac{\mathtt{k_1}}{2}$ (given by $1.(a) + 2.(a)$)

2. $\mathtt{A_1(y_u)} \rightarrow \mathtt{A_1(y_p)}$ @ $\frac{\mathtt{k_1}}{2} + \frac{\mathtt{k_1}}{2}$ (given by $1.(b) + 2.(b)$)

3. $\mathtt{A_1(x_p, y_p)} \rightarrow$ @ $\frac{\mathtt{k_2}}{2} + \frac{\mathtt{k_2}}{2}$ (given by $3.(a) + 3.(b)$).

So we find a set of rules that is identical to the one we started from. By this, we can conclude that the sites x and y play exactly the same role and that the set of reactions showed in Figure 4.2 is equivalent to the one showed in Figure 4.1.

4.1.2 Second case study

As in the first case study, we will consider an agent A carrying two sites, x and y, that can be phosphorylated or not. Anyway the behaviour of the system will be slightly modified. Like before the sites can be both activated, but that time x has an activation rate constant equal to k_1, whereas y has

an activation rate constant equal to k_2. Once that a protein A has both sites x and y phosphorylated, it can be degraded with a rate constant equal to k_3.

The behaviour of the system is described by the set of reactions given in Figure 4.3 and it corresponds to the following set of Kappa rules.

1. $\mathtt{A(x_u)} \rightarrow \mathtt{A(x_p)}$ @ $\mathtt{k_1}$
2. $\mathtt{A(y_u)} \rightarrow \mathtt{A(y_p)}$ @ $\mathtt{k_2}$
3. $\mathtt{A(x_p, y_p)} \rightarrow$ @ $\mathtt{k_3}$.

As before, we wonder if x and y play exactly the same role for each instance of the protein A. So, again, we propose to replace each rule with all the combinations of rules that can be obtained by replacing (independently) each occurrence of x and y with x or y and dividing the kinetic rates by the number of combinations (in all the set). By performing this change, we need to take care of gain and loss of automorphisms.

With our example we obtain the following set of substitutions.

1. $\mathtt{A_1(x_u)} \rightarrow \mathtt{A_1(x_p)}$ @ $\mathtt{k_1}$
   (a) $\mathtt{A_1(x_u)} \rightarrow \mathtt{A_1(x_p)}$ @ $\frac{\mathtt{k_1}}{2}$ (by changing nothing)
   (b) $\mathtt{A_1(y_u)} \rightarrow \mathtt{A_1(y_p)}$ @ $\frac{\mathtt{k_1}}{2}$ (by replacing x with y)
2. $\mathtt{A_1(y_u)} \rightarrow \mathtt{A_1(y_p)}$ @ $\mathtt{k_2}$
   (a) $\mathtt{A_1(x_u)} \rightarrow \mathtt{A_1(x_p)}$ @ $\frac{\mathtt{k_2}}{2}$ (by changing nothing)
   (b) $\mathtt{A_1(y_u)} \rightarrow \mathtt{A_1(y_p)}$ @ $\frac{\mathtt{k_2}}{2}$ (by replacing x with y)
3. $\mathtt{A_1(x_p, y_p)} \rightarrow$ @ $\mathtt{k_3}$
   (a) $\mathtt{A_1(x_p, y_p)} \rightarrow$ @ $\frac{\mathtt{k_3}}{2}$ (by changing nothing)
   (b) $\mathtt{A_1(y_p, x_p)} \rightarrow$ @ $\frac{\mathtt{k_3}}{2}$ (by replacing x with y and y with x)

And we get the following set of rules.

1. $\mathtt{A_1(x_u)} \rightarrow \mathtt{A_1(x_p)}$ @ $\frac{\mathtt{k_1}}{2} + \frac{\mathtt{k_2}}{2}$ (given by $1.(a) + 2.(a)$)

$A[u,u] \xrightarrow{k_1} A[p,u]$
$A[u,p] \xrightarrow{k_1} A[p,p]$

(a) Phosphorylation of the first site.

$A[u,u] \xrightarrow{k_2} A[u,p]$
$A[p,u] \xrightarrow{k_2} A[p,p]$

(b) Phosphorylation of the second site.

$A[p,p] \xrightarrow{k_3}$

(c) Degradation.

Figure 4.3: Chemical reactions for the second case study.

2. $\mathtt{A_1(y_u)} \rightarrow \mathtt{A_1(y_p)}$ @ $\frac{\mathtt{k_1}}{2} + \frac{\mathtt{k_2}}{2}$ (given by $1.(b) + 2.(b)$)

3. $\mathtt{A_1(x_p, y_p)} \rightarrow$ @ $\frac{\mathtt{k_3}}{2} + \frac{\mathtt{k_3}}{2}$ (given by $3.(a) + 3.(b)$)

As we can observe, in order to recover the set of rules that we started from, it is necessary that $k_1 = k_2$. So, the site x and y play exactly the same role if and only if $k_1 = k_2$. In this case, we are back to the example of Section 4.1.1.

4.1.3 Third case study

In our third example, we consider a protein A carrying two sites, x and y. This two sites can be free or can be bound to another instance of the same site (in a different protein). Here we face with a case of polymerisation, so it is not possible anymore to write the full set of reactions. Anyway the behaviour of the system can be summarised by the following set of Kappa rules.

1. $\mathtt{A_1(x), A_2(x)} \rightarrow \mathtt{A_1(x^{A_2@x}), A_2(x^{A_1@x})}$ @ $\mathtt{k}$

2. $\mathtt{A_1(y), A_2(y)} \rightarrow \mathtt{A_1(y^{A_2@y}), A_2(y^{A_1@y})}$ @ $\mathtt{k}$.

As in the previous cases we would like to understand if x and y play exactly the same role in each instance of the protein A. Again, we propose to replace each rule with all the combinations of rules that can be obtained by replacing (independently) each occurrence of x and y with x or y and dividing the kinetic rates by the number of combinations (in all the set). By performing this change, we need to take care of gain and loss of automorphisms.

From our example we obtain the following set of substitutions.

1. $\mathtt{A_1(x), A_2(x)} \rightarrow \mathtt{A_1(x^{A_2@x}), A_2(x^{A_1@x})}$ @ $\mathtt{k}$

(a) $\mathtt{A}_1(\mathtt{x}), \mathtt{A}_2(\mathtt{x}) \to \mathtt{A}_1(\mathtt{x}^{\mathtt{A}_2@\mathtt{x}}), \mathtt{A}_2(\mathtt{x}^{\mathtt{A}_1@\mathtt{x}}) \ @ \ \frac{\mathtt{k}}{4}$

(by changing nothing)

(b) $\mathtt{A}_1(\mathtt{y}), \mathtt{A}_2(\mathtt{x}) \to \mathtt{A}_1(\mathtt{y}^{\mathtt{A}_2@\mathtt{x}}), \mathtt{A}_2(\mathtt{x}^{\mathtt{A}_1@\mathtt{y}}) \ @ \ \frac{\mathtt{k}}{4} \cdot \frac{1}{2}$

(by replacing x with y in A_1)

(c) $\mathtt{A}_1(\mathtt{x}), \mathtt{A}_2(\mathtt{y}) \to \mathtt{A}_1(\mathtt{x}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{x}}) \ @ \ \frac{\mathtt{k}}{4} \cdot \frac{1}{2}$

(by replacing x with y in A_2)

(d) $\mathtt{A}_1(\mathtt{y}), \mathtt{A}_2(\mathtt{y}) \to \mathtt{A}_1(\mathtt{y}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{y}}) \ @ \ \frac{\mathtt{k}}{4}$

(by replacing x with y in both A_1 and A_2)

2. $\mathtt{A}_1(\mathtt{y}), \mathtt{A}_2(\mathtt{y}) \to \mathtt{A}_1(\mathtt{y}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{y}}) \ @ \ \mathtt{k}$

(a) $\mathtt{A}_1(\mathtt{y}), \mathtt{A}_2(\mathtt{y}) \to \mathtt{A}_1(\mathtt{y}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{y}}) \ @ \ \frac{\mathtt{k}}{4}$

(by changing nothing)

(b) $\mathtt{A}_1(\mathtt{x}), \mathtt{A}_2(\mathtt{y}) \to \mathtt{A}_1(\mathtt{x}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{x}}) \ @ \ \frac{\mathtt{k}}{4} \cdot \frac{1}{2}$

(by replacing y with x in A_1)

(c) $\mathtt{A}_1(\mathtt{y}), \mathtt{A}_2(\mathtt{x}) \to \mathtt{A}_1(\mathtt{y}^{\mathtt{A}_2@\mathtt{x}}), \mathtt{A}_2(\mathtt{x}^{\mathtt{A}_1@\mathtt{y}}) \ @ \ \frac{\mathtt{k}}{4} \cdot \frac{1}{2}$

(by replacing y with x in A_2)

(d) $\mathtt{A}_1(\mathtt{x}), \mathtt{A}_2(\mathtt{x}) \to \mathtt{A}_1(\mathtt{x}^{\mathtt{A}_2@\mathtt{x}}), \mathtt{A}_2(\mathtt{x}^{\mathtt{A}_1@\mathtt{x}}) \ @ \ \frac{\mathtt{k}}{4}$

(by replacing y with x in both A_1 and A_2)

and we obtain the following set of rules:

1. $\mathtt{A}_1(\mathtt{x}), \mathtt{A}_2(\mathtt{x}) \to \mathtt{A}_1(\mathtt{x}^{\mathtt{A}_2@\mathtt{x}}), \mathtt{A}_2(\mathtt{x}^{\mathtt{A}_1@\mathtt{x}}) \ @ \ \frac{\mathtt{k}}{2}$

(given by $1.(a) + 2.(d)$)

2. $\mathtt{A}_1(\mathtt{y}), \mathtt{A}_2(\mathtt{y}) \to \mathtt{A}_1(\mathtt{y}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{y}}) \ @ \ \frac{\mathtt{k}}{2}$

(given by $2.(a) + 1.(d)$)

3. $\mathtt{A_1(x), A_2(y)} \rightarrow \mathtt{A_1(x^{A_2@y}), A_2(y^{A_1@x})}$ @ $\frac{\mathtt{k}}{2}$

   (given by $1.(b) + 1.(c) + 2.(b) + 2.(c)$).

So, since the third rule does not belong to the original set, the two systems are equivalent if, and only if, k is equal to 0.

4.1.4 Fourth case study

In our last example, we consider again a protein A carrying two sites, x and y. The two sites can be bounded together or to another instance of the same site. As before, that system allows polymerization, so it is not possible to write the full set of reactions. Anyway, we can describe its behavior by the following set of Kappa rules.

1. $\mathtt{A_1(x), A_2(x)} \rightarrow \mathtt{A_1(x^{A_2@x}), A_2(x^{A_1@x})}$ @ $\mathtt{k_1}$
2. $\mathtt{A_1(y), A_2(y)} \rightarrow \mathtt{A_1(y^{A_2@y}), A_2(y^{A_1@y})}$ @ $\mathtt{k_2}$
3. $\mathtt{A_1(x), A_2(y)} \rightarrow \mathtt{A_1(x^{A_2@y}), A_2(y^{A_1@x})}$ @ $\mathtt{k_3}$
4. $\mathtt{A_1(y), A_2(x)} \rightarrow \mathtt{A_1(y^{A_2@x}), A_2(x^{A_1@y})}$ @ $\mathtt{k_4}$.

If we apply the transformation that replaces each rule with all the combinations of rules which can be obtained by replacing (independently) each occurence of x and y with x or y, we get this result.

1. $\mathtt{A_1(x), A_2(x)} \rightarrow \mathtt{A_1(x^{A_2@x}), A_2(x^{A_1@x})}$ @ $\mathtt{k_1}$

   (a) $\mathtt{A_1(x), A_2(x)} \rightarrow \mathtt{A_1(x^{A_2@x}), A_2(x^{A_1@x})}$ @ $\frac{\mathtt{k_1}}{4}$

   (by changing nothing)

   (b) $\mathtt{A_1(y), A_2(x)} \rightarrow \mathtt{A_1(y^{A_2@x}), A_2(x^{A_1@y})}$ @ $\frac{\mathtt{k_1}}{4} \cdot \frac{1}{2}$

   (by replacing x with y in A_1)

   (c) $\mathtt{A_1(x), A_2(y)} \rightarrow \mathtt{A_1(x^{A_2@y}), A_2(y^{A_1@x})}$ @ $\frac{\mathtt{k_1}}{4} \cdot \frac{1}{2}$

   (by replacing x with y in A_2)

   (d) $\mathtt{A_1(y), A_2(y)} \rightarrow \mathtt{A_1(y^{A_2@y}), A_2(y^{A_1@y})}$ @ $\frac{\mathtt{k_1}}{4}$

   (by replacing x with y both in A_1 and A_2)

2. $\mathtt{A_1(y), A_2(y)} \to \mathtt{A_1(y^{A_2@y}), A_2(y^{A_1@y})}\ @\ \mathtt{k_2}$

   (a) $\mathtt{A_1(y), A_2(y)} \to \mathtt{A_1(y^{A_2@y}), A_2(y^{A_1@y})}\ @\ \frac{\mathtt{k_2}}{4}$

   (by changing nothing)

   (b) $\mathtt{A_1(x), A_2(y)} \to \mathtt{A_1(x^{A_2@y}), A_2(y^{A_1@x})}\ @\ \frac{\mathtt{k_2}}{4} \cdot \frac{1}{2}$

   (by replacing y with x in A_1)

   (c) $\mathtt{A_1(y), A_2(x)} \to \mathtt{A_1(y^{A_2@x}), A_2(x^{A_1@y})}\ @\ \frac{\mathtt{k_2}}{4} \cdot \frac{1}{2}$

   (by replacing y with x in A_2)

   (d) $\mathtt{A_1(x), A_2(x)} \to \mathtt{A_1(x^{A_2@x}), A_2(x^{A_1@x})}\ @\ \frac{\mathtt{k_2}}{4}$

   (by replacing y with x both in A_1 and A_2)

3. $\mathtt{A_1(x), A_2(y)} \to \mathtt{A_1(x^{A_2@y}), A_2(y^{A_1@x})}\ @\ \mathtt{k_3}$

   (a) $\mathtt{A_1(x), A_2(y)} \to \mathtt{A_1(x^{A_2@y}), A_2(y^{A_1@x})}\ @\ \frac{\mathtt{k_3}}{4}$

   (by changing nothing)

   (b) $\mathtt{A_1(y), A_2(y)} \to \mathtt{A_1(y^{A_2@y}), A_2(y^{A_1@y})}\ @\ \frac{\mathtt{k_3}}{4} \cdot 2$

   (by replacing x with y in A_1)

   (c) $\mathtt{A_1(x), A_2(x)} \to \mathtt{A_1(x^{A_2@x}), A_2(x^{A_1@x})}\ @\ \frac{\mathtt{k_3}}{4} \cdot 2$

   (by replacing y with x in A_2)

   (d) $\mathtt{A_1(y), A_2(x)} \to \mathtt{A_1(y^{A_2@x}), A_2(x^{A_1@y})}\ @\ \frac{\mathtt{k_3}}{4}$

   (by replacing x with y in A_1 and y with x in A_2)

4. $\mathtt{A_1(y), A_2(x)} \to \mathtt{A_1(y^{A_2@x}), A_2(x^{A_1@y})}\ @\ \mathtt{k_4}$

   (a) $\mathtt{A_1(y), A_2(x)} \to \mathtt{A_1(y^{A_2@x}), A_2(x^{A_1@y})}\ @\ \frac{\mathtt{k_4}}{4}$

   (by changing nothing)

   (b) $\mathtt{A_1(x), A_2(x)} \to \mathtt{A_1(x^{A_2@x}), A_2(x^{A_1@x})}\ @\ \frac{\mathtt{k_4}}{4} \cdot 2$

   (by replacing y with x in A_1)

(c) $\mathtt{A}_1(\mathtt{y}), \mathtt{A}_2(\mathtt{y}) \rightarrow \mathtt{A}_1(\mathtt{y}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{y}}) \ @ \ \frac{\mathtt{k}_4}{4} \cdot 2$

(by replacing x with y in A_2)

(d) $\mathtt{A}_1(\mathtt{x}), \mathtt{A}_2(\mathtt{y}) \rightarrow \mathtt{A}_1(\mathtt{x}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{x}}) \ @ \ \frac{\mathtt{k}_3}{4}$

(by replacing y with x in A_1 and x with y in A_2).

By summing up syntactically equivalent rules, we obtain:

1. $\mathtt{A}_1(\mathtt{x}), \mathtt{A}_2(\mathtt{x}) \rightarrow \mathtt{A}_1(\mathtt{x}^{\mathtt{A}_2@\mathtt{x}}), \mathtt{A}_2(\mathtt{x}^{\mathtt{A}_1@\mathtt{x}}) \ @ \ \frac{\mathtt{k}_1}{4} + \frac{\mathtt{k}_2}{4} + \frac{\mathtt{k}_3}{2} + \frac{\mathtt{k}_4}{2}$

   (given by $1.(a) + 2.(d) + 3.(c) + 4.(b)$)

2. $\mathtt{A}_1(\mathtt{y}), \mathtt{A}_2(\mathtt{y}) \rightarrow \mathtt{A}_1(\mathtt{y}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{y}}) \ @ \ \frac{\mathtt{k}_1}{4} + \frac{\mathtt{k}_2}{4} + \frac{\mathtt{k}_3}{2} + \frac{\mathtt{k}_4}{2}$

   (given by $1.(d) + 2.(a) + 3.(b) + 4.(c)$)

3. $\mathtt{A}_1(\mathtt{x}), \mathtt{A}_2(\mathtt{y}) \rightarrow \mathtt{A}_1(\mathtt{x}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{x}}) \ @ \ \frac{\mathtt{k}_1}{8} + \frac{\mathtt{k}_2}{8} + \frac{\mathtt{k}_3}{4} + \frac{\mathtt{k}_4}{4}$

   (given by $1.(c) + 2.(b) + 3.(a) + 4.(d)$)

4. $\mathtt{A}_1(\mathtt{y}), \mathtt{A}_2(\mathtt{x}) \rightarrow \mathtt{A}_1(\mathtt{y}^{\mathtt{A}_2@\mathtt{x}}), \mathtt{A}_2(\mathtt{x}^{\mathtt{A}_1@\mathtt{y}}) \ @ \ \frac{\mathtt{k}_1}{8} + \frac{\mathtt{k}_2}{8} + \frac{\mathtt{k}_4}{4} + \frac{\mathtt{k}_4}{4}$

   (given by $1.(b) + 2.(c) + 3.(d) + 4.(a)$).

We notice that the rules:

$$\mathtt{A}_1(\mathtt{x}), \mathtt{A}_2(\mathtt{y}) \rightarrow \mathtt{A}_1(\mathtt{x}^{\mathtt{A}_2@\mathtt{y}}), \mathtt{A}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{x}})$$

and:

$$\mathtt{A}_1(\mathtt{y}), \mathtt{A}_2(\mathtt{x}) \rightarrow \mathtt{A}_1(\mathtt{y}^{\mathtt{A}_2@\mathtt{x}}), \mathtt{A}_2(\mathtt{x}^{\mathtt{A}_1@\mathtt{y}})$$

perform exactly the same action. Thus we can gather their contribution. As a result, we can conclude that the initial system and the transformed one will have the same behaviour if, and only if, $k_1 = k_2 = k_3 + k_4$.

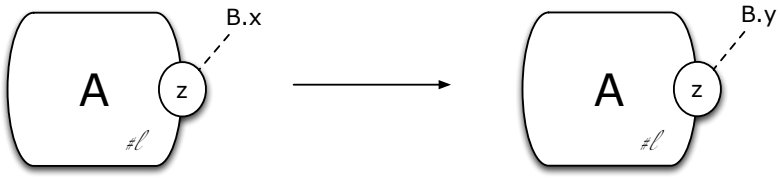


Figure 4.4: Transposition of binding types.

4.2 Permutations of sites in Kappa

4.2.1 Transpositions of sites

In this section, we formalise the action of a transposition of two sites on patterns and rules. This notion will be used to define when two sites are symmetric for a given set of rules.

We consider two kinds of transformation of patterns:

1. transposition of binding types;
2. transposition of states.

The former one consists in replacing a site name with another one in an instance of a binding type. This is defined as follows:

Definition 4.2.1 (Transposition of binding types)**.** *A transposition of binding types is defined as a tuple* $(A, l, z, B, x, y) \in \mathcal{A} \times \mathbb{N} \times \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{S}$ *such that* $z \in \Sigma_\lambda(A)$ *and* $x, y \in \Sigma_\lambda(B)$*.*

In Figure 4.4 we can see an agent A with a site z that exhibits a binding type $B@x$ that is replaced with a binding type $B@y$. The second kind of transposition consists in permuting the state of two sites in one agent. This is defined as follows:

Definition 4.2.2 (Transposition of states)**.** *A transposition of states is defined as a tuple* $(A, l, x, y) \in \mathcal{A} \times (\mathbb{N} \cup \bar{\mathbb{N}}) \times \mathcal{S} \times \mathcal{S}$ *such that the following properties are satisfied:*

1. *the site* x *belongs to the set* $\Sigma(A)$*;*
2. *the site* x *belongs to* $\Sigma_\iota(A)$ *if and only if the site* y *belongs to* $\Sigma_\iota(A)$*;*
3. *the site* x *belongs to* $\Sigma_\lambda(A)$ *if and only if the site* y *belongs to the set* $\Sigma_\lambda(A)$*.*

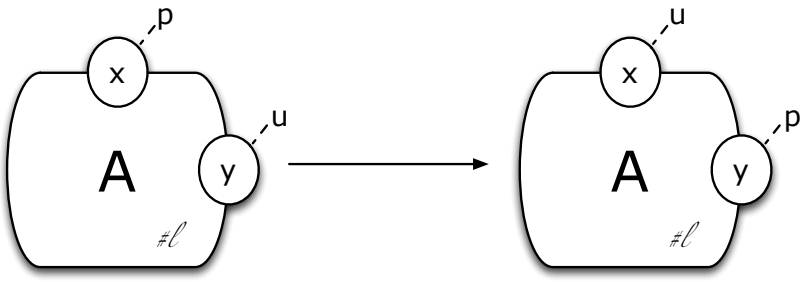


Figure 4.5: Transposition of states.

In Figure 4.5 a transposition of states is applied to an agent A that has the site x phosphorylated and the site y unphosphorylated. The result is an agent A where the states of the two sites are swapped (after the transposition the site x is unphosphorylated and the site y is phosphorylated).

A transposition is either a transposition of binding types or a transposition of states. The set of all transpositions is denoted by $\mathbb{T}$.

4.2.2 Action of a transposition on patterns

We will see how transpositions operate on patterns.

Action of a transposition of binding types on a pattern

Let E be a pattern. A *transposition of binding type* $t := (A, l, z, B, x, y)$ operates on E in the following way (as showed in Figure 4.4):

1. if E contains an agent A with identifier l documenting the site z, then if the binding state of z is the binding type $B@x$, it is replaced with the binding type $B@y$;

2. else if the binding state of z is the binding type $B@y$, then it is replaced with the binding type $B@x$;

3. in any other cases, E is not modified.

Action of a transposition of states on a pattern.

Let E be a pattern. If E exhibits an agent A with identifier l, the transposition of states (A, l, x, y) denotes that we want to permute the internal state

or the binding state of the sites x and y in the agent A with identifier l (as showed in Figure 4.5). The transformation is defined in two steps.

Firstly we define E_1 as the expression which is obtained by replacing any instance of a site address $A_l@x$ with the site address $A_l@y$ and vice versa.

Let us write E_1 as a sequence $a'_1, .., a'_n$ of agents. We know that there exists a unique agent a'_k in E_1 of type A and identifier l. Let us write $a'_k := A_l(\sigma)$. We define the expression E_2 by replacing in E_1 the agent a'_k with the agent $A_l(\sigma')$ where σ' is defined as the interface where:

1. the site x appears in σ' if and only if the site y appears in σ, then the value of the two sites is the same;

2. the site y appears in σ' if and only if the site x appears in σ, then the value of the two sites is the same;

3. any site z in $\Sigma(A)\backslash\{x, y\}$ that appears in σ it appears also in σ' with the same value.

The expression E_2 is called the result of the application of the transposition t on the pattern E.

Whenever there is no agent A with identifier l in the pattern E then the pattern E remains unchanged.

Given a transposition t, we denote by $\mathbf{subs}(t, E)$ the result of the application of the transposition t on the pattern E. We notice that if E is a pattern, then $\mathbf{subs}(t, E)$ is a pattern too. Moreover, if E is a proper pattern (resp. a mixture, resp. a species), then $\mathbf{subs}(t, E)$ is a proper pattern (resp. a mixture, resp. a species) as well.

Example 4.2.3. *We consider the following signature:*

- $\mathcal{A} = \{A, B\}$,
- $\mathcal{S} = \{x, y, z\}$,
- $\mathbb{I} = \varnothing$,
- $\Sigma_\iota(A) = \Sigma_\iota(B) = \varnothing$,
- *and* $\Sigma_\lambda(A) = \Sigma_\lambda(B) = \{x, y, z\}$.

We consider the pattern:

$$E := A_1(x^{A_2@y}), A_2(y^{A_1@x}, z^{B@x}).$$

Applying the transposition of binding types $t_1 := (A, 2, z, B, x, y)$ *to* E *replaces the binding type* $B@x$ *with the binding type* $B@y$ *in the agent* A *with identifier* 2*:*

$$\mathbf{subs}(t_1, E) = A_1(x^{A_2@y}), A_2(y^{A_1@x}, z^{B@y}).$$

Moreover, applying the transposition of states $t_2 := (A, 1, x, y)$ *to* E *is computed in two steps. Firstly we replace the site address* $A_1@x$ *with the site address* $A_1@y$ *, secondly we replace the site name* x *with the site name* y *in the agent* A *with identifier* 1*. Thus, we get:*

$$\mathbf{subs}(t_2, E) = A_1(y^{A_2@y}), A_2(y^{A_1@y}, z^{B@x}).$$

Lastly, applying the transposition of states $t_3 := (A, 2, y, z)$ *is computed in two steps. Firstly we replace the site address* $A_2@y$ *with the site address* $A_2@z$*, secondly we swap the states of the site* y *and of the site* z *in the agent* A *with identifier* 2*. Thus, we get:*

$$\mathbf{subs}(t_3, E) = A_1(x^{A_2@z}), A_2(z^{A_1@x}, y^{B@x}).$$

4.2.3 Action of a transposition on a rule

Given a set of rules $\mathcal{R}$, let us consider a rule $r := E_\ell \xrightarrow{k} E_r$ and a transposition t. The rule:

$$r' := \mathbf{subs}(t, E_\ell) \xrightarrow{k'} \mathbf{subs}(t, E_r),$$

where $k' = k \cdot \frac{\mathbf{card}([\mathbf{subs}(t,E_\ell),\mathbf{subs}(t,E_\ell)])}{\mathbf{card}([E_\ell,E_\ell])}$, is well-defined.

In such a case, the rule r' is called the *action of the transposition* t *on the rule* r, and it is denoted by $\mathbf{subs}_{\mathcal{R}}(t, r)$. Special care has to be taken about the kinetic rates of rules to account for the potential gain/loss of automorphisms in their lhs.

4.2.4 Action of a transposition on a rule application

In this section, we show that the action of a transposition is compatible with the notion of matching between patterns and with the notion of agent replacement. As a consequence we can apply transpositions directly on rule applications.

Proposition 4.2.4 (Transposition of states on agent matching)**.** *Let* $A_\ell(\sigma_\ell)$ *and* $B_i(\sigma_i)$ *be two agents such that* $B_i(\sigma_i) \models A_\ell(\sigma_\ell)$ *and let* $t = (B, i, x, y)$ *be a transposition that operates on* $B_i(\sigma_i)$*. Then the transposition* $t' = (A, \ell, x, y)$ *is such that* $\mathbf{subs}(t, B_i(\sigma_i)) \models \mathbf{subs}(t', A_\ell(\sigma_\ell))$*.*

Proof. Since $B_i(\sigma_i) \models A_\ell(\sigma_\ell)$ we know that the type of the two agents is the same. So $\Sigma_\iota(A) = \Sigma_\iota(B)$ and $\Sigma_\lambda(A) = \Sigma_\lambda(B)$ The only sites that can be affected by the transposition are x and y. We face several cases.

1. x and y do not appear in σ_ℓ. We consider $\sigma_i = \sigma_{i_1}, \sigma_{i_2}$, where σ_{i_2} is equal to σ_i without the sites x, y and σ_{i_2} is equal to the site x, y in σ_i. Since x and y do not appear in σ_ℓ we know that $\sigma_{i_1} \models \epsilon$ and $\sigma_{i_2} \models \sigma_\ell$. So $\mathbf{subs}(t, B_i(\sigma_{i_1}, \sigma_{i_2})) = B_i(\sigma'_{i_1}, \sigma_{i_2})$ and $\mathbf{subs}(t, A_\ell) = A_\ell(\sigma_\ell)$. Since $\sigma'_{i_1} \models \epsilon$ and $\sigma_{i_2} \models \sigma_\ell$, we have $\mathbf{subs}(t, B_i(\sigma_i)) \models \mathbf{subs}(t, A_\ell(\sigma_\ell))$.

2. x appears in σ_ℓ and y does not. If x appears in σ_ℓ then it appears also in σ_i. We can write $\sigma_\ell = x_\ell, \sigma'_\ell$ and $\sigma_i = \sigma'_i, x_i, \sigma''_i$ where $\sigma'_i = \epsilon$ or $\sigma'_i = y_i$. We have $\sigma'_i \models \epsilon$, $x_i \models x_\ell$ and $\sigma''_i \models \sigma'_\ell$. We take $t' = (A, \ell, x, y)$. $\mathbf{subs}(t, B_i(\sigma_i)) = \mathbf{subs}(t, B_i(\sigma'_i, x_i, \sigma''_i))$. $\mathbf{subs}(t', A_\ell(\sigma_\ell)) = \mathbf{subs}(t, A_\ell(x_\ell, \sigma'_\ell)) = A_\ell(y_\ell, \sigma'_\ell)$.

   Since the transposition t on B_i assigns the former value of the site x to the site y, we have $y_i \models y_\ell$, $\mathbf{subs}(t, B_i(\sigma'_i)) \models \epsilon$ and $\mathbf{subs}(t, B_i(\sigma''_i)) = B_i(\sigma''_i) \models A_\ell(\sigma'_\ell) = \mathbf{subs}(t', A_\ell(\sigma'_\ell))$.

   So $\mathbf{subs}(t, B_i(\sigma_i)) \models \mathbf{subs}(t', A_\ell(\sigma_\ell))$.

3. y appears in σ_ℓ and x does not. This case is symmetrical to the previous one.

4. x and y appear in σ_ℓ. Since x and y appear in σ_ℓ they appear also in σ_i. Let us write $\sigma_\ell = x_\ell, y_\ell, \sigma'_\ell$ and $\sigma_i = x_i, y_i, \sigma'_\ell$. We have that $x_i \models x_\ell$, $y_i \models y_\ell$ and $\sigma'_i \models \sigma'_\ell$. We take $t' = (A, \ell, x, y)$. Since the transpositions just swaps the value of x and y we have that $\mathbf{subs}(t, B_i(\sigma_i)) \models \mathbf{subs}(t, A_\ell(\sigma_\ell))$.

□

Proposition 4.2.5 (Transposition of binding types on agent matching)**.** *Let $A_\ell(\sigma_\ell)$ and $B_i(\sigma_i)$ be two agents such that $B_i(\sigma_i) \models A_\ell(\sigma_\ell)$ and let $t = (B, i, z, C, x, y)$ be a transposition on binding types that operates on $B_i(\sigma_i)$. Then $\mathbf{subs}(t, B_i(\sigma_i)) \models \mathbf{subs}(t', A_\ell(\sigma_\ell))$, where t' is the transposition (A, ℓ, z, C, x, y).*

Proof. We have the following cases.

1. The site z does not appear in σ_ℓ. We have $\mathbf{subs}(t', A_\ell(\sigma_\ell)) = A_\ell(\sigma_\ell)$. Since the transposition t does not modify any other site in $B_i(\sigma_i)$, $\mathbf{subs}(t, B_i(\sigma_i)) \models A_\ell(\sigma_\ell)$.

2. The site z appears in σ_ℓ and σ_i can be written as $z^{C@x}, \sigma'_i$. We can have $\sigma_\ell = z^{C@x}, \sigma'_\ell$ or $\sigma_\ell = z^?, \sigma'_\ell$. Since $B_i(\sigma_i) \models A_\ell(\sigma_\ell)$, we know that $\sigma'_i \models \sigma'_\ell$ and z in σ_ℓ is matched by $z^{C@x}$. So the state of z in σ_ℓ can be $z^{C@x}$ or ?. In the former case we have $\mathbf{subs}(t, B_i(z^{C@x}, \sigma'_i)) = B_i(z^{C@y}, \sigma'_i) \models A_\ell(z^{C@y}, \sigma'_\ell) = \mathbf{subs}(t', A_\ell(\sigma_\ell))$; in the latter $\mathbf{subs}(t, B_i(z^{C@x}, \sigma'_i)) = B_i(z^{C@y}, \sigma'_i) \models A_\ell(z^?, \sigma'_\ell) = \mathbf{subs}(t', A_\ell(\sigma_\ell))$.

□

Proposition 4.2.6 (Transposition of states on patterns embedding)**.** *Let E_ℓ, E be two patterns and $t = (A, i, x, y)$ be a transposition on states that operates on E. Let ϕ be an injective substitution. Then, if ϕ induces an embedding from E_ℓ to E, then there exists a transposition t' such that ϕ induces an embedding from $\mathbf{subs}(t', E_\ell)$ to $\mathbf{subs}(t, E)$.*

Proof. Since ϕ induces an embedding we know that for every agent (A, l) such that $A \in \mathcal{A}$ and $l \in \mathbf{agents}(E_\ell, A)$, there exists an agent a' such that $a \equiv a'$ and $a' \models \bar{\phi}(a_\ell)$, where a_ℓ is the unique agent in E_ℓ of type with identifier l and a is the unique agent in E of type A with identifier $\phi(A, l)$. So if we apply the transposition $t = (A, i, x, y)$ to E, we face several cases.

1. There is $j \in \mathbf{agents}(E_\ell, A)$ such that $i = \phi(A, j)$. Let a' be such that $a_i \equiv a'$ and $a' \models a_j$. Then, by Proposition 4.2.4, it exists t' such that $\mathbf{subs}(t, a_i) \models \mathbf{subs}(t', a_j)$. So ϕ induces an embedding from $\mathbf{subs}(t', E_\ell)$ to $\mathbf{subs}(t, E)$.

2. There is no $j \in \mathbf{agents}(E_\ell, A)$ such that $i = \phi(A, j)$. So, we face two possibilities:

   (a) The site x of the agent A_i denotes an address $C_k@z$ and there is an agent C_l in E_ℓ such that $(C, k) = \phi(C, l)$ and the site z of C_l denotes the binding type $A@x$. Then we take $t' = (C, l, A, x, y)$. We can write E_ℓ as $C_l(\sigma_l), E'_\ell = C_l(z^{A@x}\sigma'_l), E'_\ell$. $\mathbf{subs}(t', E_\ell) = C_l(z^{A@y}\sigma'_l)$.

   We have $E = A_i(\sigma_i), E' = A_i(x^{C_k@z}\sigma_i), E'$ and $\mathbf{subs}(t, E) = C_l(z^{A@x}\sigma''_l)$. So ϕ induces an embedding from $\mathbf{subs}(t', E_\ell)$ to $\mathbf{subs}(t, E)$.

   (b) Otherwise we take $t' = (C, k, A, x, y)$, where k is an index that does not appear in E'_ℓ, and we have that ϕ induces an embedding from $\mathbf{subs}(t', E_\ell) = E_\ell$ to $\mathbf{subs}(t, E)$.

□

Proposition 4.2.7 (Transposition of binding types on patterns embedding)**.** *Let E_ℓ, E be two patterns and $t = (A, i, z, B, x, y)$ be a transposition on binding types that operates on E. Let ϕ be an injective substitution. Then if ϕ induces an embedding from E_ℓ to E, then there exists a transposition t' such that ϕ induces an embedding from* $\mathbf{subs}(t', E_\ell)$ *to* $\mathbf{subs}(t, E)$.

Proof. Since ϕ is an embedding we know that for any agent (A, l) such that $A \in \mathcal{A}$ and $l \in \mathbf{agents}(E_\ell, A)$, there exists an agent a' such that $a \equiv a'$ and $a' \models \bar{\phi}(a_\ell)$, where a_ℓ is the unique agent in E_ℓ of type with identifier l and a is the unique agent in E of type A with identifier $\phi(A, l)$. So if we apply the transposition $t = (A, i, z, B, x, y)$ to E, we face two cases.

1. There is $j \in \mathbf{agents}(E_\ell, A)$ such that $i = \phi(A, j)$. Let a' be such that $a_i \equiv a'$ and $a' \models a_j$. Then, by Proposition 4.2.5, it exists t' such that $\mathbf{subs}(t', a_i) \models \mathbf{subs}(t, a_j)$. So ϕ induces an embedding from $\mathbf{subs}(t', E_\ell)$ to $\mathbf{subs}(t, E)$.

2. There is no $j \in \mathbf{agents}(E_\ell, A)$ such that $i\phi(A, j)$. In this case we take $t' = (C, k, A, x, y)$, where k is an index that does not appear in E'_ℓ, and we have that ϕ induces an embedding from $\mathbf{subs}(t', E_\ell)$ to $\mathbf{subs}(t, E)$.

□

Proposition 4.2.8 (Agent replacement)**.** *Let $A_i(\sigma_r)$ and $A_i(\sigma_i)$ be two agents and $t = (A, i, x, y)$ be a transposition of states that applies to $A_i(\sigma_i)[A_i(\sigma_r)]$. Then it exists a transposition t' such that:*

$$\mathbf{subs}(t, A_i(\sigma_i)[A_i(\sigma_r)]) = \mathbf{subs}(t, A_i(\sigma_i))[\mathbf{subs}(t', A_i(\sigma_r))].$$

Proof. We face the following cases.

1. The sites x and y do not appear in σ_r. There we can take $t' = t$, then $[\mathbf{subs}(t', A_i(\sigma_r))] = A_i(\sigma_r)$ and we have $\mathbf{subs}(t, A_i(\sigma_i)[A_i(\sigma_r)]) = \mathbf{subs}(t, A_i(\sigma_i))[\mathbf{subs}(t', A_i(\sigma_r))]$.

2. The site x appears in σ_i and the site y does not.

   We can write $\sigma_i = x_i, \sigma'_i$ and $\sigma_r = x_r, \sigma'_r$. We have:

   $$\begin{aligned} \mathbf{subs}(t, A_i(x_i, \sigma'_i)[A_i(x_r, \sigma'_r)]) &= \mathbf{subs}(t, A_i(x_i, \sigma'_i)[(x_r, \sigma'_r)]) \\ &= \mathbf{subs}(t, A_i(x_i[x_r], [\sigma'_i, \sigma'_r]) \end{aligned}$$

   $x_i[x_r] = x_r$ so:

   $$\begin{aligned} \mathbf{subs}(t, A_i(x_i[x_r], [\sigma'_i, \sigma'_r])) &= \mathbf{subs}(t, A_i(x_r, [\sigma'_i, \sigma'_r])) \\ &= A_i(y, [\sigma'_i, \sigma'_r]) \end{aligned}$$

where y has the old value of x_r.

On the other side we have:

$$\begin{aligned}\mathbf{subs}(t, A_i(\sigma_i))[\mathbf{subs}(t, A_i(\sigma_r))] &= \mathbf{subs}(t, A_i(x_i, \sigma'_i))[\mathbf{subs}(t, A_i(x_r\sigma'_r))] \\ &= A_i(y_i, \sigma'_i)[A_i(y_r, \sigma'_r)] \\ &= A_i(y_i, \sigma'_i)[y_r, \sigma'_r]) \\ &= A_i(y_i[y_r], \sigma'_i)[\sigma'_r]) \\ &= A_i(y_r, \sigma'_i)[\sigma'_r])\end{aligned}$$

$y_i[y_r] = y_r$, where y_r has the old value of x_r.

3. The site y appears in σ_i and the site x does not.

   This case is symmetric to the previous one.

4. The sites x and y appear both in σ_i and in σ_n. So we can write σ_i as x_i, y_i, σ'_i and σ_r as x_r, y_r, σ'_r.

$$\begin{aligned}\mathbf{subs}(t, A_i(x_i, y_i\sigma'_i)[A_i(x_r, y_r\sigma'_r)]) &= \mathbf{subs}(t, A_i(x_i, y_i\sigma'_i)[(x_r, y_r\sigma'_r)]) \\ &= \mathbf{subs}(t, A_i(x_i[x_r], y_i[y_r][\sigma'_i, \sigma'_r])) \\ &= A_i(y_i, x_i, \sigma'_i)[A_i(y_r, x_r, \sigma'_r)] \\ &= \mathbf{subs}(t, A_i(\sigma_i))[\mathbf{subs}(t, A_i(\sigma_y))]\end{aligned}$$

   As we wanted to show.

□

Proposition 4.2.9. *Let $\lambda := (r, E, \phi)$ be a transition label and $t := (A, i, x, y)$ be a transposition on states that operates on E. Then:*

1. *if there is no integer j such as $\phi(A, j) = i$ we have that ϕ is an embedding between the lhs of r and $\mathbf{subs}(t, E)$;*

2. *otherwise it exists a transposition t' such that if we consider $r' := \mathbf{subs}_{\mathcal{R}}(t', r)$, ϕ is also an embedding between the lhs of r' and $\mathbf{subs}(t, E)$.*

Proof. Let us consider the first case. Since a transposition does not change the index of agents, if there is no j such that $\phi(A, j) = i$, ϕ is still an embedding between the lhs of r and $\mathbf{subs}(t, E)$.

Otherwise, let j be such that $\phi(A, j) = (A, i)$. By definition of embedding $A_i \models A_j$. By Proposition 4.2.4, it exists t' such that $\mathbf{subs}(t, A_i) \models \mathbf{subs}(t', A_j)$. So if we take $r' := \mathbf{subs}_{\mathcal{R}}(t', r)$, we have that ϕ is still an embedding between the lhs of r' and $\mathbf{subs}(t, E)$. □

Definition 4.2.10 (Transposition of states on a transition label)**.** *Let $\lambda := (r, E, \phi)$ be a transition label and $t = (A, i, x, y)$ be a transposition on states that operates on E. Let r' be the rule that is defined as r whenever there is no integer j such as $\phi(A, j) = i$ or $r' := \mathbf{subs}_{\mathcal{R}}(t', r)$ (where t' is given by Definition 4.2.9). Let E' be the mixture $\mathbf{subs}(t, E)$. We call the triple (r', E', ϕ) the action of the transposition (A, i, x, y) on the transition label λ and we denote it by $\mathbf{subs}_{\mathcal{L}}(t, \lambda)$.*

Proposition 4.2.11. *Let $\lambda := (r, E, \phi)$ be a transition label and $t := (A, i, z, C, x, y)$ be a transposition on binding types that operates on E. Then:*

1. *if there is no integer j such as $\phi(A, j) = i$ we have that ϕ is an embedding between the lhs of r and $\mathbf{subs}(t, E)$;*

2. *otherwise it exists a transposition t' such that if we consider $r' := \mathbf{subs}_{\mathcal{R}}(t', r)$, ϕ is also an embedding between the lhs of r' and $\mathbf{subs}(t, E)$.*

Proof. The proof is analogous to the one for Proposition 4.2.9. □

Definition 4.2.12 (Transposition of binding types on a transition label)**.** *Let $\lambda := (r, E, \phi)$ be a transition label and $t = (A, i, z, C, x, y)$ be a transposition on binding states that operates on E. Let r' be the rule that is defined as r whenever there is no integer j such as $\phi(A, j) = i$ or $r' := \mathbf{subs}_{\mathcal{R}}(t', r)$ (where t' is given by Proposition 4.2.11). Let E' be the mixture $\mathbf{subs}(t, E)$. We call the triple (r', E', ϕ) the action of the transposition t on the transition label λ and we denote it by $\mathbf{subs}_{\mathcal{L}}(t, \lambda)$.*

We recall that if $r = E_\ell \rightarrow E_r@k$ $(E_r = b_1, ..., b_n)$ is a rule, if $E = a_1, ..., a_m$ is a pattern and if ϕ is an injective substitution that induces an embedding from E_ℓ to E, then the result of the application of r on E along ϕ, which is denoted by $E[E_r]_\phi$ is defined as the mixture $a'_1, ..., a'_m$, where for any integer i such that $1 \leqslant i \leqslant m$, a'_i is defined as $a_i[b_j]$ whenever there exists an integer j such that $\phi(A, j) = i$, and as a_i otherwise.

Proposition 4.2.13. *Let r be a rule $E_\ell \rightarrow E_r@k$, $E = a_1, ..., a_m$ be a pattern and ϕ be an injective substitution that induces an embedding from E_ℓ to E. We have $\mathbf{subs}(t, E)[\mathbf{subs}(t, E_r)]_\phi = \mathbf{subs}(t, E[E_r]_\phi)$, where t denotes the transposition (A, l, x, y).*

Proof. $\mathbf{subs}(t, E[E_r]_\phi)$ is equal to $\mathbf{subs}(t, (a'_1, ..., a'_m))$ where for any integer i such that $1 \leqslant i \leqslant m$, a'_i is defined as $a_i[b_j]$ whenever there exists an integer j such that $\phi(A, j) = i$, and as a_i otherwise. We consider $(\mathbf{subs}(t, a'_1), ..., \mathbf{subs}(t, a'_m))$. Then the only a'_i that is affected by the transposition $t = (A, l, x, y)$ is the one such that $i = l$.

There we face two cases:

1. a'_i is equal to $a_i[b_j]$ (where j is such that $\phi(A, j) = i$),

2. or a'_i is equal to a_i.

In the latter case, $a'_i = a_i$, we have $\mathbf{subs}(t, a_i) = \mathbf{subs}(t, a_i)$. Otherwise we can apply Proposition 4.2.8, so $\mathbf{subs}(t, a_i)[\mathbf{subs}(b_j)] = \mathbf{subs}(t, a_i[b_j])$.

□

Proposition 4.2.14. *Let E be a species, $\lambda := (r, E, \phi)$ be a transition label and $t = (A, i, x, y)$ be a transposition that operates on E.*

We have:

$$E \xrightarrow{\lambda} E' \iff \mathbf{subs}(t, E) \xrightarrow{\mathbf{subs}_{\mathcal{L}}(t,\lambda)} \mathbf{subs}(t, E')$$

Proof. By Proposition 4.2.6 we know that if $t = (A, i, x, y)$ is a transposition that operates on E, if ϕ is an injective substitution that induces an embedding from E_ℓ to E, then there exists t_ℓ such that ϕ induces an embedding from $\mathbf{subs}(t_\ell, E_\ell)$ to $\mathbf{subs}(t, E)$.

$\mathbf{subs}_{\mathcal{L}}(t, \lambda)$ is equal to $(r', \mathbf{subs}(t, E), \phi)$, where r' is equal to $\mathbf{subs}(t', E_\ell) \to \mathbf{subs}(t', E_r)$.

If we take $t' = t_\ell$ we have that $\mathbf{subs}(t', E_\ell) \lhd_\phi \mathbf{subs}(t, E)$, so the rule r' applies to $\mathbf{subs}(t, E)$.

If we apply r' to $\mathbf{subs}(t, E)$ we obtain $\mathbf{subs}(t, E)[\mathbf{subs}(t, E_r)]_\phi$.

$\mathbf{subs}(t, E)[\mathbf{subs}(t, E_r)]_\phi = \mathbf{subs}(t, E[E_r]_\phi)$ that is equal to $\mathbf{subs}(t, E')$.

The converse application comes from the fact that t is an involution. □

4.2.5 Symmetric sites

Transpositions will be used in order to identify sites having the same capabilities of interaction.

In the following, we assume that no syntactic rule occurs twice in a model. This does not restrict our framework, since multiple instances of a syntactic rule can be merged, by summing up their rate constants.

By syntactic rules, we mean that we consider rules up to structural congruence $\equiv$, but not up to rule isomorphism.

Definition 4.2.15 (Strong (syntactical) symmetry)**.** *The sites x and y are symmetric in A, for the set of rules $\mathcal{R}$, if for every rule r and every transposition t that operates on r, the rule $\mathbf{subs}_{\mathcal{R}}(t, r)$ belong to the set of rules $\mathcal{R}$ as well and the ratio between its rate constant and the number of automorphisms in its lhs is the same.*

This is a very strong requirement based on a syntactical equivalence. Later we will see a less demanding notion, when rules can be considered up to isomorphisms (e.g. see Definition 4.2.16).

What is interesting for us it is to understand if a given system and the same one, after a transposition on its sites, perform in the same way (i.e. the two systems are equivalent). In order to show that two rules sets are equivalent it can be necessary to reorder interfaces and reindex agents in rules (by applying a same into substitution to both sides of a given rule) and gather some rules having the same lhs and the same rhs (summing up the rates).

Given a rule r and a non negative real number $k \in \mathbb{R}^+$, we define $\mathbf{scale}(r, k)$ as the rule that is obtained by multiplying the rate constant of r by k. Moreover, we define the orbit of the rule r, which is written $\mathbf{orbit}(r)$, as the set of rules which can be obtained by applying zero, one, or several transpositions of states to the rule r. Since the lhs and the rhs of a rule are finite expressions, the orbit of a rule is always a finite set.

Our model transformation is formalised by the binary relation $\Rightarrow$ over sets of rules, which is defined as follows:

$$\mathcal{R} \Rightarrow \{\mathbf{scale}(r', \frac{1}{\mathbf{card}(\mathbf{orbit}(r))}) | r \in \mathcal{R}, r' \in \mathbf{orbit}(r)\}.$$

for any set of rules $\mathcal{R}$.

Agents and sites in rules can be reordered using the congruence relation over both hand sides. Moreover, agents can be reindexed using substitutions.

A *generalised* substitution is a mapping ϕ from $\mathcal{A} \times \mathbb{N} \cup \bar{\mathbb{N}}$ to $\mathbb{N} \cup \bar{\mathbb{N}}$ such that:

1. for any proper identifier $l \in \mathbb{N}$ we have $\phi(l) \in \mathbb{N}$;
2. for any temporary identifier $\bar{l} \in \bar{\mathbb{N}}$ we have $\phi(\bar{l}) \in \bar{\mathbb{N}}$.

A generalised substitution is into if, and only if, for any identifier $l, l' \in \mathbb{N} \cup \bar{\mathbb{N}}$, $\phi(l) = \phi(l')$ implies $l = l'$. The extension $\bar{\phi}$ of a generalised substitution ϕ to agents is defined as in the case of substitution (see Section 3.3).

Definition 4.2.16 ($\approx$-equivalence over rules)**.** *Given two rules $r := E_\ell \xrightarrow{k} E_r$ and $r' := E'_\ell \xrightarrow{k'} E'_r$, we say that r and r' are isomorphic, and we write $r \approx r'$, if and only if:*

1. *$k = k'$,*
2. *and there exists a generalised substitution ϕ such that $\bar{\phi}(E_\ell) \equiv E'_\ell$ and $\bar{\phi}(E_r) \equiv E'_r$.*

To understand if two sets of rules are equivalent we need to check if they can be made equal by replacing their rules with $\approx$-equivalent ones and by gathering the rules having the same lhs and the same rhs (in such a case, their rates are summed up).

The idea is the following: let us fix an agent type A and two sites x and y in $\Sigma(A)$ such that $x \in \Sigma_\lambda(A) \Leftrightarrow y \in \Sigma_\lambda(A)$ and $x \in \Sigma_\iota(A) \Leftrightarrow y \in \Sigma_\iota(A)$. So, in order to detect whether x and y have the same capabilities of interaction, we will replace each rule with the combination of rules which can be obtained by substituting zero, one or several occurrences of x with y, and zero, one or several occurrences of y with x.

If the obtained system of rules is equivalent to the initial one, then the sites x and y have the same capabilities of interaction.

Example 4.2.17. *Let us have a look to some example. We start by considering the following set of rules:*

$$\mathtt{A_1(x_u)} \xrightarrow{\mathtt{k_1}} \mathtt{A_1(x_p)}$$

$$\mathtt{A_1(y_u)} \xrightarrow{\mathtt{k_2}} \mathtt{A_1(y_p)}$$

$$\mathtt{A_1(x_p, y_p)} \xrightarrow{\mathtt{k_3}} \varepsilon$$

The signature is: $\mathcal{A} = A$, $\mathcal{S} = \{x, y\}$, $\mathbb{I} = \{u, p\}$, $\Sigma_\iota = \{x, y\}$ *and* $\Sigma_\lambda = \varnothing$.

We can notice that whenever the rates k_1 *and* k_2 *are equal, the sites* x *and* y *are symmetric in* A*. So, the system is equivalent to the following one:*

$$\mathtt{A_1(x_u)} \xrightarrow{\mathtt{k'}} \mathtt{A_1(x_p)}$$

$$\mathtt{A_1(y_u)} \xrightarrow{\mathtt{k'}} \mathtt{A_1(y_p)}$$

$$\mathtt{A_1(x_p, y_p)} \xrightarrow{\mathtt{k_3}} \varepsilon$$

where $k' = \frac{k_1+k_2}{2}$.

Let us detail the transformation. The first two rules are replaced with two rules each, with a half rate, one applying on the site x*,and the other applying on the site* y*. The four obtained rules are pairwise equal, so we gather them pairwise. When transforming the third rule, we obtain two equivalent rules (with half rate), that we can gather to recover the initial rule.*

Example 4.2.18. *Let us have a look to a more subtle example and let us consider the following set of rules:*

$$\mathtt{A_1(x), A_2(x)} \xrightarrow{\mathtt{k_1}} \mathtt{A_1(x^{A_2@x}), A_2(x^{A_1@x})}$$

$$\mathtt{A_1(x)}, \mathtt{A_2(y)} \xrightarrow{\mathtt{k_2}} \mathtt{A_1(x^{A_2@y})}, \mathtt{A_2(y^{A_1@x})}$$

$$\mathtt{A_1(y)}, \mathtt{A_2(y)} \xrightarrow{\mathtt{k_3}} \mathtt{A_1(y^{A_2@y})}, \mathtt{A_2(y^{A_1@y})}$$

The signature is: $\mathcal{A} = A$, $\mathcal{S} = \{x, y\}$, $\mathbb{I} = \varnothing$, $\Sigma_\iota = \varnothing$ *and* $\Sigma_\lambda = \{x, y\}$.

Here the sites x *and* y *are symmetric whenever* $k_1 = k_2 = k_3$. *This case is more subtle than the previous one, because some rules gain or loose some symmetries during the transformation. For instance, the transformation of the first rule gives four rules. One of them binds the sites* x *of two agents* A, *another binds the sites* y *of two agents* A. *The lhss of these two rules have the same number of symmetries as the one of the initial rule. Thus, their rate constant are divided by* 4 *(since the rule is rewritten into* 4 *rules). Another rule binds the site* x *of the agent* A *with identifier* 1 *and the site* y *of the agent* A *with identifier* 2, *and the last one binds the site* y *of the agent* A *with identifier* 1 *and the site* y *of the agent* A *with identifier* 2. *The number of symmetries in these rules is twice less as the number of symmetries in the initial rule. Thus the rate constant are divided by* 8 *(*4 *since the rule is rewritten into* 4 *rules, and* 2 *due to the loss of symmetries). But the two obtained rules are equivalent up to reordering and reindexing, thus, we obtain a single rule, the rate constant of which had been divided by* 4. *The transformation of the remaining rules works the same way, except that the rule which binds the sites* x *of two agents* A *and the rule which binds the sites* y *of two agents* A *both gain symmetries (the rate constant is divided by* 4 *and multiplied by* 2*), and the rule which binds the site* x *and the site* y *of two agents* A *keeps the same number of symmetries (the rate constant is divided by* 4*).*

So as to avoid testing for all triples (A, x, y) such that $A \in \mathcal{A}$ and $x, y \in \Sigma(A)$, whether the condition in 4.2.15 is satisfied or not, we use a weaker property to preselect the potential triples. Two symmetric sites have necessarily the same set of potential partners, and there internal states range among the same set of values. This can be detected by the static analyses that are proposed in [27, 31].

4.3 Symmetries and rule applications

In this section we want to show that a pair of symmetric sites induces a bisimulation and how this fact can be exploited to define a model reduction. First of all, we recall the definition of bismulation given in Chapter 2.

Definition 4.3.1. *Given an autonomous system $(\mathcal{V}, \mathbb{F})$ and a relation r, we say that that r induces a bisimulation over $(\mathcal{V}, \mathbb{F})$ if and only if for any pair (ρ, ρ') of states over $\mathcal{V}$, if $P_r(\rho) = P_r(\rho')$, then $P_r(\mathbb{F}(\rho)) = P_r(\mathbb{F}(\rho'))$.*

Indeed, a relation r induces a bisimulation over an autonomous system $(\mathcal{V}, \mathbb{F})$ if, and only if, $P_r \circ \mathbb{F} = P_r \circ \mathbb{F} \circ P_r$.

Definition 4.3.2 (Stoichiometric vector)**.** *Given a reaction*

$$React : R_1, \ldots, R_n \to P_1, \ldots, P_m,$$

we define the stoichiometric vector $V(React)$ as:

$$V(React)(v) = \sum_{i=1}^{m} \begin{cases} 1 \ if \ P_i = v \\ 0 \ otherwise \end{cases} - \sum_{i=1}^{n} \begin{cases} 1 \ if \ R_i = v \\ 0 \ otherwise, \end{cases}$$

for every species v.

For any species v, we denote as $\{S_{v,1}, \ldots, S_{v,v_R}\}$ its orbit. The orbit of a species is obtained by applying every sequence on transpositions on the species v.

We consider a reaction *React* induced by a rule *rule*. We denote as $\{React_1, \ldots, React_k\}$ its orbit. The orbit of a reaction is obtained by applying every sequence on transpositions on the lhs of the reaction *React*.

For each natural number j among 1 and k, we write:

$$React_j \ : \ R_{j,1}, \ldots, R_{j,n} \to P_{j,1}, \ldots, P_{j,m}$$

and we define γ_j as the quotient between the rate constant of the underlying rule and the number of automorphisms in the lhs of this rule.

We notice that for every two natural numbers j, j' among 1 and k, we have:

$$P_r(V(React_j)) = P_r(V(React_{j'})),$$

and:

$$\gamma_j = \gamma_{j'}.$$

Given a state ρ, the overall contribution of the orbit $[React]_{\approx}$ to the function $\mathbb{F}$ can be expressed as:

$$\sum_{j=1}^{k} \left(\gamma_j \times \prod_{i=1}^{n} \rho(R_{j,i}) \times V(React_j) \right).$$

Then, we have:

$$\begin{aligned}
&P_r\left(\textstyle\sum_{j=1}^{k}\left(\gamma_j \times \prod_{i=1}^{n}\rho(R_{j,i}) \times V(React_j)\right)\right)\\
&= \left(\textstyle\sum_{j=1}^{k}\left(\gamma_j \times \prod_{i=1}^{n}\rho(R_{j,i}) \times P_r(V(React_j))\right)\right)\\
&= \left(\textstyle\sum_{j=1}^{k}\left(\gamma_1 \times \prod_{i=1}^{n}\rho(R_{j,i}) \times P_r(V(React_1))\right)\right)\\
&= \gamma_1 \times \textstyle\sum_{j=1}^{k}\prod_{i=1}^{n}\rho(R_{j,i}) \times P_r(V(React_1))\\
&= \gamma_1 \times \textstyle\prod_{i=1}^{n}\sum_{j=1}^{k_{R_{j,1}}}\rho(S_{R,j}) \times P_r(V(React_1)).
\end{aligned}$$

But, for any i between 1 and n, we have, by definition of P_r:

$$\sum_{j=1}^{k_{R_{j,1}}}\rho(S_{R,j}) = \sum_{j=1}^{k_{R_{j,1}}}P_r(\rho(S_{R,j})).$$

It follows that:

$$\begin{aligned}
&P_r\left(\textstyle\sum_{j=1}^{k}\left(\gamma_j \times \prod_{i=1}^{n}\rho(R_{j,i}) \times V(React_j)\right)\right)\\
&= P_r\left(\textstyle\sum_{j=1}^{k}\left(\gamma_j \times \prod_{i=1}^{n}P_r(\rho(R_{j,i})) \times V(React_j)\right)\right)
\end{aligned}$$

Summing up the contribution for every orbit of reactions, we get that the relation r induces a bisimulation, as stated in the following theorem:

Theorem 4.3.3. *We have:*

$$P_r \circ \mathbb{F} = P_r \circ \mathbb{F} \circ P_r.$$

4.4 Conclusion

We have introduced a notion of symmetry among sites for Kappa models. Intuitively, two sites are symmetric in a model, if they play the same role in each instance of protein.

Formally, this is not so easy, because several syntactically different rules may have the same semantics, and we have to take this into account. We defined two notions of symmetry. The first one, called strong symmetry, is purely syntactic (it makes the distinction between two rules having the same semantics). With this notion, two sites are symmetric, if whenever we swap every instance of them in every rule, and we collect the rules that are identical (up to structural congruence), we get back the initial model. The second notion of symmetry considers rules up to their semantics. With this notion, two sites are symmetric, whenever the rule set is semantically equivalent to one that is strongly symmetric.

We show, by induction over the definition of the semantics, that each symmetry between sites induce a bisimulation. Thus symmetries can be used to reduce the dimension of models.

Chapter 5

Production/consumption of a pattern

In this chapter, we express the derivative of the amount of each pattern in a model as an expression of the amount of other patterns.

5.1 Consumption/production of a species

As a preliminary, we reformulate the consumption and the production of a given species in Kappa models.

Let us consider the following set of rules $\mathcal{R}$:

$$\begin{array}{lcll} r_1 & := & c_{1,1}, \ldots, c_{\omega(1),1} \rightarrow p_{1,1}, \ldots, p_{\theta(1),1} & k_1 \\ r_2 & := & c_{1,2}, \ldots, c_{\omega(2),2} \rightarrow p_{1,2}, \ldots, p_{\theta(2),2} & k_2 \\ \ldots & := & \ldots & \\ r_h & := & c_{1,h}, \ldots, c_{\omega(h),h} \rightarrow p_{1,h}, \ldots, p_{\theta(h),h} & k_h \end{array}$$

where for every rule r_ι, $\omega(\iota)$ returns the number of components in the left hand side and $\theta(\iota)$ returns the number of components in the right hand side.

For each rule r_ι, we denote as γ_ι the quotient between the rate constant k_ι and the number of automorphisms in the left hand side of the rule r_ι.

We consider all the ground refinements of $\mathcal{R}$, where every rule r_ι has $\nu(\iota)$ refinements (one for each composite embedding between the lhs of the rule

into a tuple of species in $\mathcal{V}$)

$$
\begin{array}{lcll}
react_{(1,1)} & := & R_{(1,1),1}, \ldots, R_{(1,1),\omega(1)} & \rightarrow \; P_{(1,1),1}, \ldots, P_{(1,1),\theta(1,1)} \\
\ldots & := & \ldots & \\
react_{(1,\nu(1))} & := & R_{(1,\nu(1)),1}, \ldots, R_{(1,\nu(1)),\omega(1)} & \rightarrow \; P_{(1,\nu(1)),1}, \ldots, P_{(1,\nu(1)),\theta(1,\nu(1))} \\
\ldots & := & \ldots & \\
\ldots & := & \ldots & \\
\ldots & := & \ldots & \\
react_{(h,1)} & := & R_{(h,1),1}, \ldots, R_{(h,1),\omega(h)} & \rightarrow \; P_{(h,1),1}, \ldots, P_{(h,1),\theta(h,1)} \\
\ldots & := & \ldots & \\
react_{(h,\nu(h))} & := & R_{(h,\nu(h)),1}, \ldots, R_{(h,\nu(h)),\omega(h)} & \rightarrow \; P_{(h,\nu(h)),1}, \ldots, P_{(h,\nu(h)),\theta(h,\nu(h))}
\end{array}
$$

For every reaction $react_{\iota,\tau}$, $\theta(\iota,\tau)$ denotes the number of products of the reaction $react_{\iota,\tau}$.

Moreover, for every reaction $react_{\iota,\tau}$, we denote as $\phi_{\iota,\tau}$ the corresponding embedding between the lhs of the rule $rule_\iota$ and the lhs of the reaction $react_{\iota,\tau}$ and as $\phi'_{\iota,\tau}$ the corresponding embedding between the rhs of the rule $rule_\iota$ and the rhs of the reaction $react_{\iota,\tau}$.

5.1.1 Consumption

Given a species $v \in \mathcal{V}$, we denote as $\delta^-(v)$ the quantity of v that is consumed as a reactant of a reaction.

This quantity is:

$$\delta^-(v) = \sum_{\iota=1}^{h} \sum_{\tau=1}^{\nu(\iota)} \gamma_\iota \sum_{j=1}^{\omega(\iota)} f^-(\iota,\tau,j,v) \prod_{l=1}^{\omega(\iota)} \rho(R_{(\iota,\tau),l})$$

where:

$$f^-(\iota,\tau,j,v) = \begin{cases} 1 & \text{if } R_{(\iota,\tau),j} = v \\ 0 & \text{otherwise.} \end{cases}$$

5.1.2 Production

Given a species $v \in \mathcal{V}$, we denote as $\delta^+(v)$ the quantity of v that is added as a product of a reaction.

This quantity is:

$$\delta^+(v) = \sum_{\iota=1}^{h} \sum_{\tau=1}^{\nu(\iota)} \gamma_\iota \sum_{j=1}^{\theta(\iota,\tau)} f^+(\iota,\tau,j,v) \prod_{l=1}^{\omega(\iota)} \rho(R_{(\iota,\tau),l})$$

where:

$$f^+(\iota,\tau,j,v) = \begin{cases} 1 & \text{if } P_{(\iota,\tau),j} = R \\ 0 & \text{otherwise.} \end{cases}$$

5.1.3 Balance

We have expressed the consumption and the production of each chemical species directly at the level of rules.

Proposition 5.1.1. *Let R be a chemical species.*
We have:

$$\mathbb{F}(\rho)(v) = \delta^+(v) - \delta^-(v).$$

5.2 Consumption/production of a pattern

Now we express, for every pattern component P, the difference between its production and its consumption.

Given a pattern component P and a state ρ over $\mathcal{V}$, we define the concentration $[P]_\rho$ of the pattern P in the state ρ, as:

$$[P]_\rho = \sum_{v \in \mathcal{V}} \frac{\mathbf{card}([P, v])}{\mathbf{card}([P, P])} \rho(v).$$

We usually write $[P]$ instead of $[P]_\rho$. We notice that $\rho(P) = [P]$ whenever the pattern component is a species in the set $\mathcal{V}$. We remind the reader that $[P, P']$ denotes the set of the embeddings from the pattern P into the pattern P'. This way, $[P, P]$ denotes the set of automorphisms of the pattern P.

We also define the corrected concentration $(\!|P|\!)_\rho$ of a pattern P as follows:

$$(\!|P|\!)_\rho = [P]_\rho \times \mathbf{card}([P, P]).$$

We differentiate the expression $[P]_\rho$ and we obtain that:

$$[P]'_\rho = \sum_{v \in \mathcal{V}} \frac{\mathbf{card}([P, v])}{\mathbf{card}([P, P])} \mathbb{F}(\rho)(v).$$

By linearity, we get that:

$$[P]'_\rho = \frac{\sum_{v \in \mathcal{V}} \mathbf{card}([P, v]) \mathbb{F}(\rho)(v)}{\mathbf{card}([P, P])}.$$

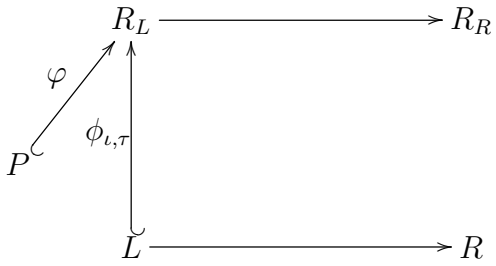
Moreover we know, by Proposition 5.1.1, that for every species $v \in \mathcal{V}$:

$$\begin{aligned} \mathbb{F}(\rho)(v) = &\textstyle\sum_{\iota=1}^{h} \sum_{\tau=1}^{\nu(\iota)} \gamma_\iota \sum_{j=1}^{\theta(\iota,\tau)} f^+(\iota, \tau, j, v) \prod_{l=1}^{\omega(\iota)} \rho(R_{(\iota,\tau),l}) \\ &- \textstyle\sum_{\iota=1}^{h} \sum_{\tau=1}^{\nu(\iota)} \gamma_\iota \sum_{j=1}^{\omega(\iota)} f^-(\iota, \tau, j, v) \prod_{l=1}^{\omega(\iota)} \rho(R_{(\iota,\tau),l}) \end{aligned}$$

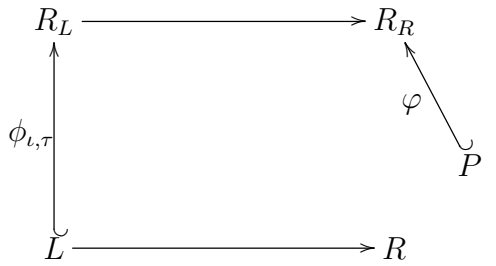
The pattern component P is consumed for each occurrence of the pattern P in a reactant R of a reaction $react_{\iota,\tau}$. It is worth noting that, whenever

the pattern P has n automorphisms, each occurrence of the pattern P corresponds to n embeddings from the pattern component P to the reactant R.

We recall that each ground refinement $R_L \to R_R$ is defined by a rule $L \to R$ and a straight-epi from the lhs L of the rule and the lhs R_L of the ground refinement. For each ground refinement, we consider every embedding between the pattern component P and the lhs R_L of the this ground refinement.



Conversely, the pattern component P is produced for each occurrence of the pattern P in a reactant R of a reaction $react_{\iota,\tau}$. Here again, whenever the pattern P has n automorphisms, each occurrence of the pattern P corresponds to n embeddings from the pattern component P to the reactant R.



It may happen that, given a ground refinement $R_L \to R_R$ and an injective substitution φ, the injective substitution φ induces an embedding both from the pattern component P to the mixture R_L and from the pattern component P to the mixture R_R. In such as case, the corresponding consumption and production terms cancel pair-wisely.

Definition 5.2.1 (proper consumption)**.** *We define the proper consumption of the pattern component P as follows.*

$$\delta_\star^-(P) = \frac{\sum_{v\in\mathcal{V}} \sum_{\varphi\in[P,v]} \sum_{\iota=1}^{h} \sum_{\tau=1}^{\nu(\iota)} \gamma_\iota \sum_{j=1}^{\omega(\iota)} f_\star^-(\iota,\tau,j,v,\varphi) \prod_{l=1}^{\omega(\iota)} \rho(R_{(\iota,\tau),l})}{\mathbf{card}([P,P])}$$

where for every rule index ι, for every reaction index τ, every position j, every species v, and every embedding φ between the pattern component P and the species v, the expression $f_\star^-(\iota, \tau, j, v, \varphi)$ is defined as 1 *if the following conditions are satisfied:*

- *the embedding $\phi_{\iota,\tau}$ maps the connected pattern $c_{j,\iota}$ to the species v,*
- *the instance of P that is defined by the embedding φ is modified by the reaction $react_{\iota,\tau}$,*

and as 0 *otherwise.*

Definition 5.2.2 (proper production)**.** *We define the proper production of the pattern component P as follows:*

$$\delta_\star^+(P) = \frac{\sum_{v\in\mathcal{V}} \sum_{\varphi\in[P,v]} \sum_{\iota=1}^{h} \sum_{\tau=1}^{\nu(\iota)} \gamma_\iota \sum_{j=1}^{\theta(\iota,\tau)} f_\star^+(\iota,\tau,j,v,\varphi) \prod_{l=1}^{\omega(\iota)} \rho(R_{(\iota,\tau),l})}{\mathbf{card}([P,P])}$$

where for every rule index ι, every reaction index τ, every position j, every species v, and every embedding φ between the pattern component P and the species v, the expression $f_\star^+(\iota, \tau, j, v, \varphi)$ is defined as 1 *if the following conditions are satisfied:*

- *the embedding $\phi_{\iota,\tau}$ maps the connected pattern $p_{j,\iota}$ to the species v,*
- *the instance of P that is defined by the embedding φ has been modified by the ground refinement $react_{\iota,\tau}$,*

and as 0 *otherwise.*

Proposition 5.2.3. *Let P be a pattern component.*
We have:

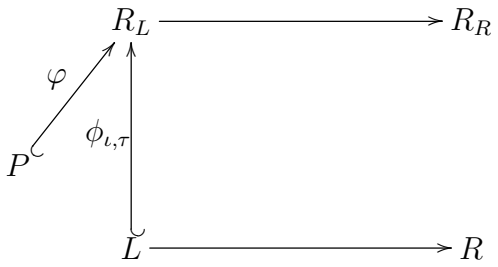
$$\sum_{v\in\mathcal{V}} \sum_{\phi\in[P,v]} \mathbb{F}(v) = [P,P](\delta_\star^+(P) - \delta_\star^-(P)).$$

5.3 Proper consumption of a pattern

Now we express the proper consumption of a pattern as the concentration of other patterns.

5.3.1 Condition

Let us fix a pattern P, a rule index ι, a reaction index τ, a position j, a species v, and an embedding φ between the pattern component P and the species v. We denote the rule $rule_\iota$ as $L \to R$ and the reaction $react_{\iota,\tau}$ as $R_L \to R_R$.



We define the notions of agent instance and site instance as follows:

Definition 5.3.1 (agent instance)**.** *Let us consider a pattern P.*

We call an agent instance *in the pattern P a pair $(A, i) \in \mathcal{A} \times \mathcal{L}$ such that there exists an agent A with identifier i in the pattern P.*

Definition 5.3.2 (site instance)**.** *Let us consider a pattern P.*

We call a site instance *in the pattern P a triple $(A, i, x) \in \mathcal{A} \times \mathcal{L} \times \mathcal{S}$ such that there exists an agent A with identifier i which documents the state of the site $x \in \Sigma(A)$ in the pattern P.*

The instance of P that is defined by the embedding φ is modified by the reaction $react_{\iota,\tau}$ if and only if at least one of both following conditions is satisfied:

- there exist an agent type $A \in \mathcal{A}$, two agent identifiers $i, i' \in \mathbb{N}$, and a site name $x \in \Sigma(A)$, such that:
  - (A, i, x) is a site instance in the pattern P,
  - (A, i', x) is a site instance in the pattern R_L,
  - $(A, \varphi(A, i), x) = (A, \phi_{\iota,\tau}(A, i'), x)$,
  - the site instance (A, i', x) is modified explicitly in the rule $rule_\iota$;
- there exist a site instance (A, i, x) in the pattern P and a site instance $(A', i', x') \in \mathcal{MAY}(rule_\iota) \cup \mathcal{MUST}(rule_\iota)$, such that:
  - the site instances $(A, \varphi(A, i), x)$ and $(A', \phi_{\iota,\tau}(A', i'), x')$ are bound in R_L of the reaction $react_{\iota,\tau}$,
  - $(A, \varphi(A, i), x)$ is a site instance in R_R,

– the site instance $(A, \varphi(A,i), x')$ is free in R_R.

We gather the proper negative contributions for the concentration of the pattern P according to how the pattern P is *glued* with a pattern component in the lhs of the rule $rule_\iota$.

5.3.2 Gluing

Let us explain how we can glue two patterns.

To glue two patterns, we need to define a pairing relation among the agents of each other (these pairs of agents will be considered equal). Moreover it is possible to add links between some site instances, so as to take into account about potential side effects. We do this in two steps. Firstly, we introduce a primitive to gather two patterns while fusing some of their agents. Secondly, we introduce a primitive to add links in patterns.

We start by defining the gluing operation.

Definition 5.3.3 (pairing)**.** *Let Z_1 and Z_2 be two patterns. Let $\mathcal{A}_{Z_1}$ and $\mathcal{A}_{Z_2}$ be respectively the set of agents of Z_1 and the set of agents of Z_2. $\mathscr{A} \in \mathcal{P}(\mathcal{A}_{Z_1} \times \mathcal{A}_{Z_2})$ is a pairing between Z_1 and Z_2 if $\forall i, i' \in \mathcal{A}_{Z_1}$ and $\forall j, j' \in \mathcal{A}_{Z_2}$ we have:*

1. *if $(i,j) \in \mathscr{A}$ and $(i,j') \in \mathscr{A}$ then $j = j'$;*
2. *if $(i,j) \in \mathscr{A}$ and $(i',j) \in \mathscr{A}$ then $i = i'$.*

Now we define the common region between two patterns, according to a pairing. Intuitively, the agents of this common region belongs to both patterns, and thus they have two agent identifiers. In order to properly rename these labels, we assume that we are given a bijection Φ from the set $(\mathbb{N} \cup \overline{\mathbb{N}})^2$ to the set $\mathbb{N} \cup \overline{\mathbb{N}}$.

Definition 5.3.4 (common part)**.** *Let Z_1 and Z_2 be two patterns, and $\mathscr{A}$ be a pairing between them.*

We define $X := Z_1 \cap_{\mathscr{A}} Z_2$ as the pattern that is defined by the following constraints:

1. *$A_{\Phi(i,j)}$ is an agent in X if and only if the pair $((A,i),(A,j))$ belongs to the set $\mathscr{A}$;*
2. *the agent $A_{\Phi(i,j)}$ in X documents the site x if and only if the site instance (A,i,x) is documented in Z_1 and the site instance (A,j,x) is documented in Z_2;*

3. *the site x of the agent $A_{\Phi(i,j)}$ is in the state w in X if and only if:*
   (a) *either (A, i, x) is in the state w in Z_1 and (A, j, x) is in the state w in Z_2,*
   (b) *or $w = \epsilon$ and (A, i, x) in Z_1 and (A, j, x) in Z_2 have a different internal state.*
4. *For every binding state λ in $\{\epsilon, -, ?\} \cup \{N@l \mid N \in \mathcal{A}, n \in \Sigma_\lambda(A)\}$, the site x of the agent $A_{\Phi(i,j)}$ has the binding state λ in X if and only if:*
   (a) *(A, i, x) has the binding state λ in Z_1 and (A, j, x) has the binding state λ in Z_2,*
   (b) *or (A, i, x) in Z_1 and (A, j, x) in Z_2 have a different binding state and $\lambda = ?$;*
5. *two sites x and y of the agents $A_{\Phi(i,j)}$ and $A_{\Phi(i',j')}$ are bounded in X if and only if the sites x and y are bounded in the agents $A_i, A_{i'}$ of Z_1 and in the agents $A_j, A_{j'}$ of Z_2.*

Not every pairing defines a gluing, because two contradicting states could be attached in both patterns to a same site. When it is defined, the gluing between two patterns Z_1 and Z_2 according to a pairing relation $\mathscr{A}$ is defined up to isomorphism, as the site graph that refines the site-graph $Z_1 \cap_{\mathscr{A}} Z_2$, by adding any information that is avaible either in Z_1 and Z_2. Special care has to be taken for dealing with the state of bound sites, because they can be defined at different levels of resolution.

The following definition formalises the notion of gluing.

Definition 5.3.5 (gluing)**.** *Let Z_1 and Z_2 be two patterns, $\mathscr{A}$ be a pairing between them and X be $Z_1 \cap_{\mathscr{A}} Z_2$.*

The gluing of the patterns Z_1 and Z_2 according to the pairing $\mathscr{A}$ is well defined if and only if there exists a pattern Y and three embeddings:

1. $\gamma_1 : Z_1 \to Y$
2. $\gamma_2 : Z_2 \to Y$
3. $\gamma_X : X \to Y$

such that:

1. *for every agent A_i in Y, at least one of both following assertions is satisfied:*
   (a) *there exists an agent A_j belonging to Z_1 such that $i = \gamma_1(A, j)$;*

(b) *there exists an agent* A_j *belonging to* Z_2 *such that* $i = \gamma_2(A, j)$*;*

2. *for every agent* A_k *in* X *(we write* $(i, j) := \Phi^{-1}(k)$*), both following assertions are satisfied:*

   (a) $\gamma_X(A, k) = \gamma_1(A, i)$*;*

   (b) $\gamma_X(A, k) = \gamma_2(A, j)$*;*

3. *for every agent* A_i *in* Y *and any site* $x \in \Sigma(A)$*, the site instance* (A, i, x) *is documented in* Y *if and only if at least one of both following assertions is satisfied:*

   (a) *there exists an agent* A_j *belonging to* Z_1 *such that* $i = \gamma_1(A, j)$ *and the site instance* (j, x) *is documented in the pattern* Z_1*;*

   (b) *there exists an agent* A_j *belonging to* Z_2 *such that* $i = \gamma_2(A, j)$ *and the site instance* (A, j, x) *is documented in the pattern* Z_2*;*

4. *for every site instance* (A, i, x) *in* Y*, the site instance* (A, i, x) *has a non-empty internal state in* Y *if and only if at least one of both following assertions is satisfied:*

   (a) *there exists an agent* A_j *belonging to* Z_1 *such that* $i = \gamma_1(A, j)$ *and the site instance* (A, j, x) *has a non-empty internal state in the pattern* Z_1*;*

   (b) *there exists an agent* A_j *belonging to* Z_2 *such that* $i = \gamma_2(A, j)$ *and the site instance* (A, j, x) *has a non-empty internal state in the pattern* Z_2*;*

5. *for every site instance* (A, i, x) *in* Y*:*

   (a) *the binding state of the site instance* (A, i, x) *belongs to the set:*

$$\{\epsilon\} \times \{A_i@x \mid A \in \mathcal{A}, i \in \mathbb{N} \cup \overline{\mathbb{N}}, x \in \Sigma_\lambda(A)\}$$

   whenever at least one of both following assertions is satisfied:

   i. *there exists an agent* A_j *belonging to* Z_1 *such that* $i = \gamma_1(A, j)$ *and the site instance* (A, j, x) *has has a binding state in the set:*

$$\{\epsilon\} \times \{A_i@x \mid A \in \mathcal{A}, i \in \mathbb{N} \cup \overline{\mathbb{N}}, x \in \Sigma_\lambda(A)\}$$

   in the pattern Z_1*;*

ii. there exists an agent A_j belonging to Z_2 such that $i = \gamma_2(A, j)$ and the site instance (A, j, x) has has a binding state in the set:

$$\{\epsilon\} \times \{A_i@x \mid A \in \mathcal{A}, i \in \mathbb{N} \cup \overline{\mathbb{N}}, x \in \Sigma_\lambda(A)\}$$

in the pattern Z_2;

(b) else the binding state of the site instance (A, i, x) is a binding type whenever at least one of both following assertions is satisfied:

i. there exists an agent A_j belonging to Z_1 such that $i = \gamma_1(A, j)$ and the binding state of the site instance (A, j, x) in the pattern Z_1 is a binding type;

ii. there exists an agent A_j belonging to Z_2 such that $i = \gamma_2(A, j)$ and the binding state of the site instance (A, j, x) in the pattern Z_2 is a binding type;

(c) else the binding state of the site instance (A, i, x) is equal to '$-$' whenever at least one of both following assertions is satisfied:

i. there exists an agent A_j belonging to Z_1 such that $i = \gamma_1(A, j)$ and the site instance (A, j, x) has equal to '$-$' in the pattern Z_1;

ii. there exists an agent A_j belonging to Z_2 such that $i = \gamma_2(A, j)$ and the site instance (A, j, x) has equal to '$-$' in the pattern Z_2;

(d) otherwise it is equal to '?'.

Whenever it is defined (up to isomorphism), the pattern Y is called the gluing between the patterns Z_1 and Z_2 according to the pairing $\mathscr{A}$, and is denoted as $Z_1 \cup_{\mathscr{A}} Z_2$.

Example 5.3.6. *In Figure 5.1 we can see an example of gluing.*
There we have:

$$Z_1 = A_3(x^{B_4@x}, y^{C_5@y}), B_4(x^{A_3@x}), C_5(y^{A_3@y})$$

$$Z_2 = A_6(x^{B_7@x}), B_7(x^{A_2@x}, z^{D_8@z}), D_8(z^{B_7@z}).$$

We may chose, for the pairing $\mathscr{A}$, any non empty subset of the following set:

$$\{((A, 3), (A, 6)), ((B, 4), (B, 7))\}.$$

The pattern X identifying the common part between Z_1 and Z_2 is:

$$A_1(x^{B_2@x}), B_2(x^{A_1@x}).$$

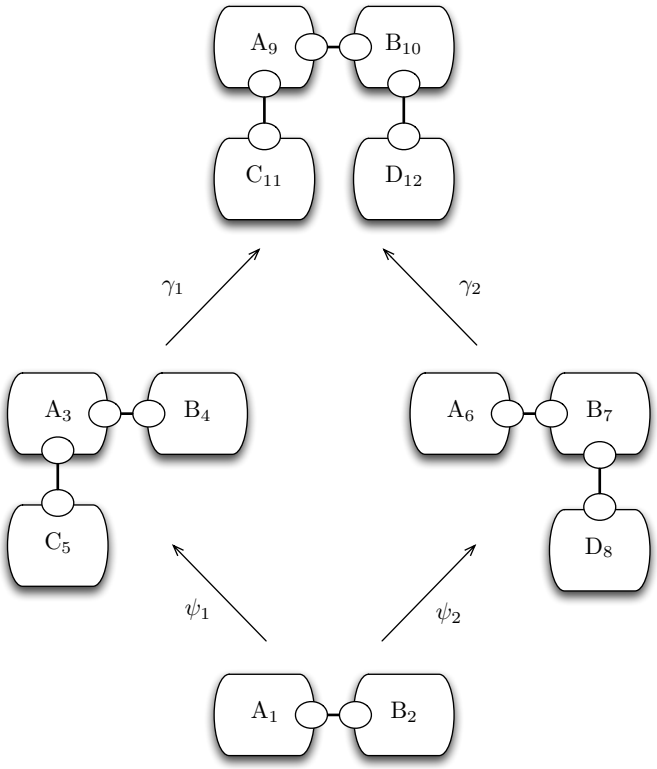


Figure 5.1: A common region and a gluing.

The corresponding gluing is defined as follows:

$$A_{3,6}(x^{B_{4,7}@x}, y^{C_5@y}), B_{4,7}(x^{A_{3,6}@x}, z^{D_8@z}), C_5(y^{A_{3,6}@y}), D_8(z^{B_{4,7}@z}).$$

We observe that in Definition 5.3.5, the fact that the sites that are both defined in Z_1 and in Z_2 have compatible states is ensured by the existence of the three embeddings.

Then, we define the operation to add a link in a pattern.

Definition 5.3.7 (bonds insertion)**.** *Let P be a pattern and B be a set of pairs of site instances, such that:*

1. *for every pair $(s, s') \in B$, $s \neq s'$;*

2. *for every two distinct pairs $(s, s'), (s'', s''') \in B$, we have:*

$$\{s, s'\} \cap \{s'', s'''\} = \varnothing;$$

3. *for every pair $(s, s') \in B$, if the site s (resp. s') is a site instance in the pattern P, then the binding state of the site instance s (resp. s') is not a site address (that is to say that its binding state belongs to the set: $\{\epsilon, -, ?\} \cup \{A@x \mid A \in \mathcal{A}, x \in \Sigma_\lambda(A)\}$).*

We define the pattern $P{\uparrow_B}$ as the pattern that is obtained by performing the following transformation on the pattern P:

1. *for every pair $((A, i, x), (A', i', x')) \in B$, if the pattern P contains the site instance (A, i, x) (resp. (A', i', x')), then the binding state of the site instance (A, i, x) (resp. (A', i', x')) is removed;*

2. *for every pair $((A, i, x), (A', i', x')) \in B$, if the pattern P has no agent A_i (resp. $A_{i'}$), then we add the agent A_i (resp. $(A_{i'})$) in the pattern P with an empty interface;*

3. *for every pair $((A, i, x), (A', i', x')) \in B$, if the pattern P has no site instance (A, i, x) (resp. (A', i', x'') then this site instance is added;*

4. *for every pair $((A, i, x), (A', i', x')) \in B$, a bond is created between the site instance (A, i, x) and the site instance (A', i', x').*

Now we have all the ingredients to define the different positions for a pattern to be consumed.

Definition 5.3.8. *Let ι be the index of a rule. Let j be the index of a connected pattern in the lhs of the rule $rule_\iota$. Let P be a pattern.*

We define the set $\overrightarrow{\mathcal{M}}(\iota, j, P)$ as the set of the pairs $(\mathscr{A}, B)$ where:

- *$\mathscr{A}$ is a pairing between $c_{\iota,j}$ and P,*

  we denote as γ_ℓ and γ_P the corresponding embeddings respectively from $c_{\iota,j}$ to $c_{\iota,j} \cup_{\mathscr{A}} P$ and from P to $c_{\iota,j} \cup_{\mathscr{A}} P$;

- *B is a set of pairs of sites instances.*

such that the following properties are satisfied:

1. *if $\mathscr{A} \neq \varnothing$, then $B = \varnothing$;*

2. *if $\mathscr{A} = \varnothing$, then B is a singleton;*

3. *for every pair $((A, i, x), s') \in B$, there exists a site instance (A', i', x') in $\mathcal{MAY}(rule_\iota) \cup \mathcal{MUST}(rule_\iota)$, such that:*

   - *$A = A'$ and $x = x'$;*

- *the agent* (A', i') *belongs to the pattern component* $c_{\iota,j}$*;*
- $(A, i, x) = (A, \gamma_\ell(A', i'), x')$*;*

4. *for every pair* $(s, (A, i, x)) \in B$*, there exists no site instance* (A', i', x') *in the pattern component* $c_{\iota,j}$ *such that* $(A, i, x) = (A', \gamma_\ell(A', i'), x)$*;*

5. *for every pair* $((A, i, x), (A', i', x')) \in B$*, there exists a site instance* (A'', i'', x'') *in the pattern* P *such that:*

   - $A'' = A'$ *and* $x'' = x'$*;*
   - $x'' \in \Sigma_\lambda(A'')$*;*
   - *the binding state of the site instance* (A'', i'', x'') *in the pattern* P *is either equal to '-', to '?', or to the binding type* $A@x$*;*

6. $\mathscr{A}$ *is equal to the set of the pairs* $((A, i), (A', i'))$ *such that* (A, i) *is an agent instance in* $c_{\iota,j}$*,* (A', i') *is an agent instance in* P*,* $A = A'$ *and* $\gamma_\ell(A, i) = \gamma_P(A', i')$*.*

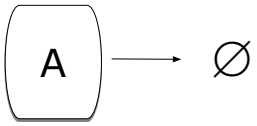
Then for every pair $(\mathscr{A}, B) \in \overrightarrow{\mathcal{M}}(\iota, j, P)$*, we define the extended gluing* $(c_{\iota,j} \cup_{(\mathscr{A},B)} P)$ *as the pattern* $(c_{\iota,j} \cup_A P)\!\uparrow_B$*.*

Note that in Definition 5.3.8 we have used the assumption that species have no cycle. This way, it is not possible to add a link to a pattern component (this justifies the following constraint: if $\mathscr{A} \neq \varnothing$, then $B = \varnothing$), and if the pattern component P is disjoint from the pattern $c_{\iota,j}$ we can create only one bond (this justifies the following constraint: if $\mathscr{A} = \varnothing$, then B is a singleton). The other constraints stipulate that a bond may have been removed between a site that has been made free by side effects and a site that were not in the lhs of the rule. Lastly, we assume that the pairing is maximal, so as to identify uniquely each way of gluing two patterns.

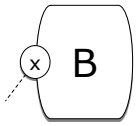
Example 5.3.9. *Let us have a look to the example depicted in Figure 5.2. As we can observe, there is no common part between the lhs of the rule in 5.2(a) and the pattern in 5.2(b). So, if we consider the gluing between them (figure 5.2(c)), this returns just the disjoint union of the two. Indeed, by computing the extendend gluing (figure 5.2(d)) we notice that the pattern is modified by the rule (figure 5.2(e)).*

5.3.3 Contribution

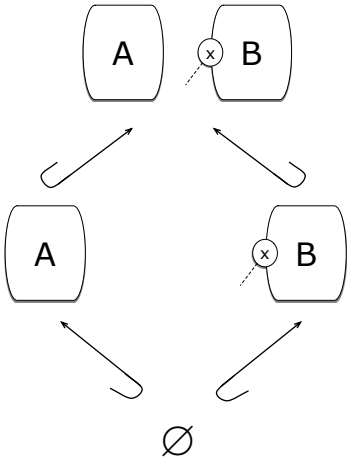
Now we partition the proper consumption of the pattern P, according to the extended gluing that it forms with a pattern component of a rule.



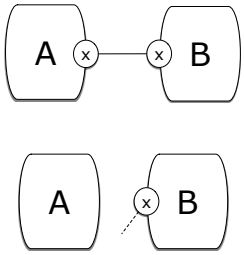
(a) A rule.



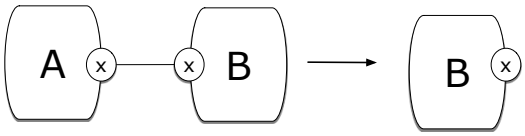
(b) A pattern.



(c) Gluing between the lhs of the rule and the pattern.



(d) Extended gluing.



(e) Refinement of the rule.

Figure 5.2: Consumption of a pattern.

We get that $\delta_\star^-(P)$ is equal to:

$$\frac{\sum_{\iota=1}^{h} \gamma_\iota \sum_{j=1}^{\omega(\iota)} \sum_{v\in\mathcal{V}} \sum_{\varphi\in[P,v]} \sum_{M\in\overrightarrow{\mathcal{M}}(\iota,j,P)} g^-(\iota,\tau,j,v,\varphi,M) \prod_{l=1}^{\omega(\iota)} \rho(R_{(\iota,\tau),l})}{\mathbf{card}([P,P])}$$

where for every rule index ι, every reaction index τ, every position j, every species $v \in \mathcal{V}$, every embedding φ from P to v, and any extended gluing $(\mathscr{A}, B)$, the expression $g^-(\iota,\tau,j,v,\varphi,M)$ is defined as 1 if the following conditions are satisfied:

- the embedding $\phi_{\iota,\tau}$ maps the connected pattern $c_{j,\iota}$ to the species v,
- the instance of P that is defined by the embedding φ is modified by the reaction $react_{\iota,\tau}$,
- $\mathscr{A} = \{((A,i),(A',i')) \mid A = A' \text{ and } \varphi(A,i) = \phi_{\iota,\tau}(A,i')\}$,
- if $\mathscr{A} = \varnothing$, then B is a singleton that describes a link that is removed by side effect;

and as 0 otherwise.

That is to say that $\delta_\star^-(P)$ is equal to:

$$\frac{\sum_{\iota=1}^{h} \gamma_\iota \sum_{j=1}^{\omega(\iota)} \sum_{M\in\overrightarrow{\mathcal{M}}(\iota,j,P)} \sum_{v\in\mathcal{V}} \sum_{\varphi\in[P,v]} g^-(\iota,\tau,j,v,\varphi,M) \prod_{l=1}^{\omega(\iota)} \rho(R_{(\iota,\tau),l})}{\mathbf{card}([P,P])}.$$

This expression can be factorised (e.g. see [25]) as follows.

Proposition 5.3.10.

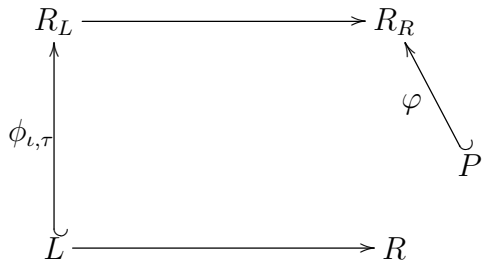
$$\delta_\star^-(P) = \frac{\sum_{\iota=1}^{h} \gamma_\iota \sum_{j=1}^{\omega(\iota)} \sum_{M\in\overrightarrow{\mathcal{M}}(\iota,j,P)} (\!|c_{j,\iota} \cup_M P|\!)_\rho \prod_{\substack{\chi=1 \\ \chi\neq j}}^{\omega(\iota)} (\!|c_{\chi,\iota}|\!)_\rho}{\mathbf{card}([P,P])}.$$

5.4 Proper production of a pattern

Now we express the proper production of a pattern as the concentration of other patterns.

5.4.1 Condition

Let us fix a pattern P, a rule index ι, a reaction index τ, a position j, a species v, and an embedding φ between the pattern component P and the species v. We denote the rule $rule_\iota$ as $L \to R$, and the reaction $react_{\iota,\tau}$ as $R_L \to R_R$.



The instance of P, that is defined by the embedding φ, has been modified by the reaction $react_{\iota,\tau}$ if and only if at least one of both following conditions is satisfied:

- the whole pattern P has been created by the application of the rule $rule_\iota$;
- there exist an agent type $A \in \mathcal{A}$, two agent identifiers $i, i' \in \mathbb{N}$, and a site name $x \in \Sigma(A)$, such that:
  - (A, i, x) is a site instance in the pattern P,
  - (A, i', x) is a site instance in R_R,
  - $(A, \varphi(A, i), x) = (A, \phi_{\iota,\tau}(A, i'), x)$,
  - the site instance (A, i', x) is modified explicitly in the rule $rule_\iota$;
- there exist a site instance (A, i, x) in the pattern P and a site instance $(A', i', x') \in \mathcal{MAY}(rule_\iota) \cup \mathcal{MUST}(rule_\iota)$, such that:
  - the site instances $(A, \varphi(A, i), x)$ and $(A', \phi_{\iota,\tau}(A', i'), x')$ are bound in R_L and the site,
  - $(A, \varphi(A, i), x)$ is a site instance in R_R,
  - the site instance $(A, \varphi(A, i), x')$ is free in R_R.

We gather the proper positive contributions for the concentration of the pattern P according to how the pattern P is *glued* with the rhs of the rule $rule_\iota$ (the pattern P might overlap with zero, one, or several pattern components in the rhs of the rule $rule_\iota$). For each gluing, we have to invert the

$$
\begin{array}{rcl}
[\epsilon;\epsilon]E & = & E \\
[a_\ell, E_\ell; a_r, E_r](a, E) & = & [a_\ell; a_r]a, [E_\ell; E_r]E \\
[a_\ell; \varnothing]\varnothing & = & a_\ell \\
[\varnothing; a_r]a & = & \varnothing \\
[A(\sigma_\ell); A(\sigma_r)]A(\sigma) & = & A([\sigma_\ell; \sigma_r]\sigma) \\
[\epsilon;\epsilon]\sigma & = & \sigma \\
[s_\ell, \sigma_\ell; s_r, \sigma_r]s, \sigma & = & [s_\ell; s_r]s, [\sigma_\ell; \sigma_r]\sigma \\
[x^{\lambda_\ell}; x^{\lambda_r}]x^\lambda & = & x^{\lambda_\ell} \text{ if } \lambda_\ell = \epsilon, i \in \mathbb{N} \\
[x^{A@x}; x^{A@x}]x^\lambda & = & x^\lambda \\
[x^-; x^-]x^\lambda & = & x^\lambda \\
[x^-; x^{\lambda_r}]x^\lambda & = & x^- \text{ if } \lambda_r \neq -
\end{array}
$$

Figure 5.3: Inverse substitution.

effect of applying the rule $rule_\iota$, so as to characterise the most general context that ensures that the rule effectively creates an instance of the pattern P. Because of the potential side effects, applying a rule is not an invertible process. Some links between sites may have been released by side effects. This way, having fixed an embedding from the rhs of a rule and a pattern, there may be zero, one, or several refinements of $rule_\iota$ that may correspond with this embedding. Yet there are at most a finite number of them and they can be enumerated easily. This provides a partitioning of the potential context for producing a given pattern thanks to a given rule.

5.4.2 Inverse substitution

Let us explain how we can invert the application of a rule.

Firstly we define this operation in the case of application of a rule that is free of side effects.

Definition 5.4.1 (side effects free right refinement of a rule)**.** *Let $r : E_\ell \to E_r$ be a rule and let E be a pattern such that $E \models E_r$. We define the* right refinement *of the rule r by the pattern E, written $\{r\}E$, as the rule $[E_\ell; E_r]E \to E$ defined by inverse substitution (see Figure 5.3).*

The side effects free right refinement of a rule is not the only way to specialise the rule r to produce the pattern E. Some bonds may have been released by side effects. These bonds can be added in the lhs of the refinement thanks to the primitive $\uparrow$.

Now we have all the ingredients to define the different ways of producing a given pattern.

Definition 5.4.2 (extended precondition)**.** *Let ι be the index of a rule. We denote the rule $rule_\iota$ as $L_\iota \to R_\iota$. Let P be a pattern.*

We define the set $\overleftarrow{\mathcal{M}}(\iota, P)$ as the set of the pairs $(\mathscr{A}, B)$ where:

- *$\mathscr{A}$ is a pairing between R_ι and P;*

  we denote as γ_r and γ_P the corresponding embeddings respectively from R_ι to $R_\iota \cup_{\mathscr{A}} P$ and from P to $R_\iota \cup_{\mathscr{A}} P$,

  since the gluing $R_\iota \cup_{\mathscr{A}} P$ is defined up to isomorphism, we may assume without any loss of generality that $R_\iota \cup_{\mathscr{A}} P \models R_\iota$ (that is to say that γ_r maps (A, i) to i, for any agent instance (A, i) in R_ι);

- *B is a set of pairs of sites instances.*

such that the following properties are satisfied:

1. *if $\mathscr{A} \neq \varnothing$, then $B = \varnothing$;*
2. *if $\mathscr{A} = \varnothing$, then B is a singleton;*
3. *for every pair $((A, i, x), s') \in B$, there exists a site instance (A', i', x') in $\mathcal{MAY}(rule_\iota) \cup \mathcal{MUST}(rule_\iota)$, such that:*
   (a) *$A = A'$ and $x = x'$;*
   (b) *the agent (A', i') belongs to L_ι;*
   (c) *$(A, i, x) = (A, \gamma_r(A', i'), x')$;*
4. *for every pair $(s, (A, i, x)) \in B$, there exists no site instance (A', i', x') in R_ι such that $(A, i, x) = (A', \gamma_r(A', i'), x)$;*
5. *for every pair $((A, i, x), (A', i', x')) \in B$, there exists a site instance (A'', i'', x'') in the pattern P such that:*
   (a) *$A'' = A'$ and $x'' = x'$;*
   (b) *$x'' \in \Sigma_\lambda(A'')$;*
   (c) *the binding state of the site instance (A'', i'', x'') in the pattern P is either equal to 'ϵ' or '?';*
6. *$\mathscr{A}$ is equal to the set of the pairs $((A, i), (A', i'))$ such that (A, i) is an agent instance in R_ι, (A', i') is an agent instance in P, $A = A'$ and $\gamma_r(A, i) = \gamma_P(A', i')$;*
7. *the pattern $([L_\iota, R_\iota](R_\iota \cup_{\mathscr{A}} P))\!\uparrow_B$ has exactly $\omega(\iota)$ pattern components.*

Then for every pair $(\mathscr{A}, B) \in \overleftarrow{\mathcal{M}}(\iota, P)$*, we define the extended precondition* $pred(\iota, P, (\mathscr{A}, B))$ *as the pattern* $([L_\iota, R_\iota](R_\iota \cup_{\mathscr{A}} P))\uparrow_B$*. Moreover, we define as* $\overline{c}_{1,\iota,P,(\mathscr{A},B)}, \ldots, \overline{c}_{\omega(\iota),\iota,P,(\mathscr{A},B)}$ *the list of its pattern components.*

Note that the inverse substitution may not preserve the connectedness of the pattern P. Let us call the antecedent of P, the pattern in $[L_\iota, R_\iota](R_\iota \cup_{\mathscr{A}} P)$ that contains exactly the agent instances $(A, \gamma_P(A, i))$ and the site instances $(A, \gamma_P(A, i), x)$ for every agent instance (A, i) and every site instance (A, i, x) of P that has not been created by the rule. Whenever the antecedent of P is not connected, then necessarily each of its pattern component contains a site instance that was free before the application of the rule and bound after. As a consequence, any pattern component of the antecedent, if it is disconnected, contains a site instance that has been modified explicitly by the rule. Then, if the antecedent of P overlaps with L_ι, then necessarily, each of its pattern component overlaps with L_ι, and then it is not possible to add a bond without creating a cycle (this justifies the following constraint: if $\mathscr{A} \neq \varnothing$, then $B = \varnothing$). If the antecedent of the pattern P is disjoint from L_ι, then we can create only one bond (this justifies the following constraint: if $\mathscr{A} = \varnothing$, then B is a singleton). Lastly, we assume that the pairing is maximal, so as to identify uniquely each way of gluing two patterns.

Example 5.4.3. *Let us have a look to the example depicted Figure 5.4. As we can observe, there is no common part between the rhs of the rule Figure 5.4(a) and the pattern Figure 5.4(b). So, if we consider the gluing between the rhs of this rule and pattern (Figure 5.4(c)), this returns just the pattern itself. If we consider the right refinement of the gluing by inverse substitution Figure 5.4(d), there is no way to see that the pattern is produced by the rule. Yet, the site of pattern might have been freed by side effect. This is captured by extending the right refinement, as depicted in Figure 5.4(e).*

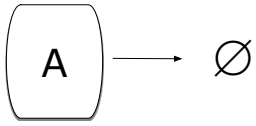
5.4.3 Contribution

We now partition the proper production of the pattern P, according to the gluing it forms with the rhs of a rule.

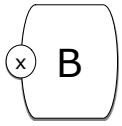
We get that $\delta^+_\star(P)$ is equal to:

$$\frac{\sum_{\iota=1}^{h} \gamma_\iota \sum_{j=1}^{\omega(\iota)} \sum_{v \in \mathcal{V}} \sum_{\varphi \in [P,v]} \sum_{M \in \overleftarrow{\mathcal{M}}(\iota,P)} g^+(\iota, \tau, j, v, \varphi, M) \prod_{l=1}^{\omega(\iota)} \rho(R_{(\iota,\tau),l})}{\mathbf{card}([P, P])}$$

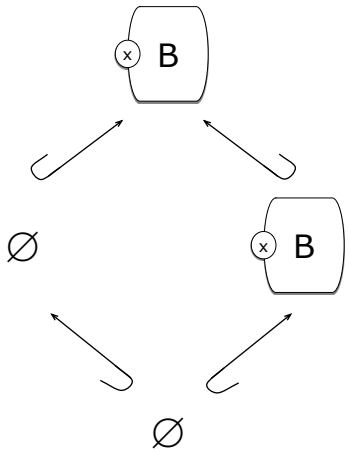
where for every rule index ι, every reaction index τ, every position j, every species $v \in \mathcal{V}$, every embedding φ from P to v, and any extended precondition



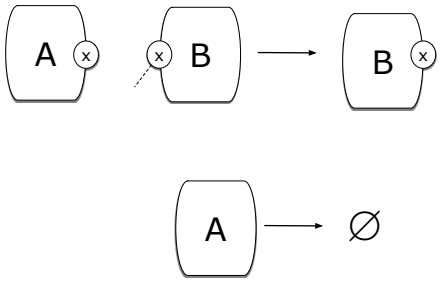
(a) A rule.



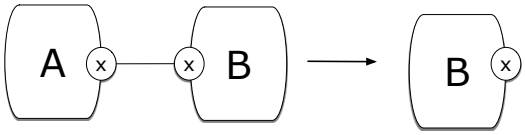
(b) A pattern.



(c) Gluing between the rhs of the rule and the pattern.



(d) Right refinement.



(e) Extended right refinement.

Figure 5.4: Production of a pattern.

$(\mathscr{A}, B)$, the expression $g^{-}(\iota, \tau, j, v, \varphi, M)$ is defined, as 1 if the following conditions are satisfied:

- the embedding $\phi'_{\iota,\tau}$ maps the connected pattern $p_{j,\iota}$ to the species v,
- the instance of P that is defined by the embedding φ is modified by the reaction $react_{\iota,\tau}$,
- $\mathscr{A} = \{((A, i), (A', i')) \mid A = A' \text{ and } \varphi(A, i) = \phi'_{\iota,\tau}(A, i')\}$,
- if $\mathscr{A} = \varnothing$, then B is a singleton that denotes a link that is removed by side effect;

and as 0 otherwise.

That is to say that $\delta^{+}_{\star}(P)$ is equal to:

$$\frac{\sum_{\iota=1}^{h} \gamma_\iota \sum_{M \in \overrightarrow{\mathcal{M}}(\iota,P)} \sum_{v \in \mathcal{V}} \sum_{\varphi \in [P,v]} \sum_{j=1}^{\omega(\iota)} g^{+}(\iota, \tau, j, v, \varphi, M) \prod_{l=1}^{\omega(\iota)} \rho(R_{(\iota,\tau),l})}{\mathbf{card}([P, P])}.$$

This expression can be factorised (e.g. see [25]) as follows:

Proposition 5.4.4.

$$\delta^{+}(P) = \frac{\sum_{\iota=1}^{h} \sum_{M \in \overleftarrow{\mathcal{M}}(\iota,P)} \prod_{\chi=1}^{\omega(\iota)} (\!|\overline{c}_{\chi,\iota,P,M}|\!)_\rho}{\mathbf{card}([P, P])}.$$

5.5 Conclusion

We have reformulated the consumption and the production term of every pattern, as an expression of the quantity of the other patterns.

There are two kinds of consumption terms. The first kind is obtained by gluing the pattern of interest to a connected pattern in the lhs of a rule on a site that is modified by the rule, the second one is obtained by taking the disjoint union between the pattern of interest and a connected pattern in the lhs of a rule while adding a bond to connect them (in such a case this bond has to be released by side effect).

There are also two kinds of production terms. The first kind consists in gluing the pattern of interest to the rhs of a rule on a site that is modified by the rule and then to apply the rule backward. The production term is then obtained as a product of the quantity of the connected pattern in the result of this operation. The second kind consists in taking the disjoint union between the pattern of interest and the lhs of the rule and then to connect them by a bond that will be released by side effect.

We have made this computation under the assumption that there are no cycles in species. This computation can be extended to the general case but, in such a case, the consumption and the production of the pattern of interest have to be expressed with respect to the quantity of more complex patterns.

Chapter 6

Information flow

In this chapter, we show how to construct an abstract/reduced semantics thanks to an over-approximation of the information flow between the different regions of chemical species. The main idea is to detect and prove that some correlations between the states of some sites are useless. Then, we can safely abstract away these correlations by cutting chemical species into some smaller pattern components, that we call fragments.

The initial semantics and the reduced one are formally related by Abstract Interpretation.

6.1 Motivating example

Before introducing formally the framework, we focus on the case study that we have seen in Section 1.2.2. Let us recall it.

We consider a protein having three activation sites r, c, l, each of which can be activated 'p', or deactivated 'u'. Thus a chemical species is denoted as a triple of symbols among 'u' and 'p', the first component denotes the state of the site l, the second one the state of the site c, and the third one the state of the site r. Initially, all the sites are deactivated. The evolution of the state of the proteins is described thanks to some chemical reactions. There is some hierarchical control between the states of the sites. The site c has to be activated first, at rate k_1, thanks to the reaction in the Figure 6.1(a). Once the site c has been activated, the l site can get activated at rate k_2, no matter the state of the site r is (see the Figure 6.1(b)); and the site r can get activated at rate k_3, no matter the state of the site l is (see the Figure 6.1(c)). We describe the flow of information among the states of the sites of a protein in the Figure 6.1(d). Intuitively, the flow of information summarises the fact that the state of the site c may control the behaviour of

$$(u,u,u) \rightarrow (u,p,u) \quad k_1$$

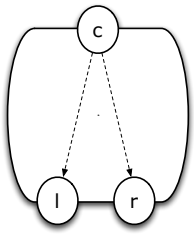
(a) 2nd site activation.

$$(u,p,u) \rightarrow (p,p,u) \quad k_2 \qquad (u,p,p) \rightarrow (p,p,p) \quad k_2$$

(b) 1st site activation.

$$(u,p,u) \rightarrow (u,p,p) \quad k_3 \qquad (p,p,u) \rightarrow (p,p,p) \quad k_3$$

(c) 3rd site activation.



(d) Flow of information.

$$\begin{cases} [u,u,u]' = -k_1[u,u,u] \\ [u,p,u]' = -k_2[u,p,u] + k_1[u,u,u] - k_3[u,p,u] \\ [u,p,p]' = -k_2[u,p,p] + k_3[u,p,u] \\ [p,p,u]' = k_2[u,p,u] - k_3[p,p,u] \\ [p,p,p]' = k_2[u,p,p] + k_3[p,p,u] \end{cases}$$

(e) Initial differential system.

$$\begin{cases} [u,u,u]' = -k_1[u,u,u] \\ ([u,p,u] + [u,p,p])' = k_1[u,u,u] - k_2([u,p,u] + [u,p,p]) \\ ([p,p,u] + [p,p,p])' = k_2([u,p,u] + [u,p,p]) \\ ([u,p,u] + [p,p,u])' = k_1[u,u,u] - k_3([u,p,u] + [p,p,u]) \\ ([u,p,p] + [p,p,p])' = k_3([u,p,u] + [p,p,u]) \end{cases}$$

(f) Reduced differential system.

Figure 6.1: Chemical reactions, flow of information, and ODEs for the protein with hierarchical flow of information.

the states of the sites l and r, but that the states of the sites l and r do not control the behaviour of the states of the other sites.

The differential semantics of this model is the solution of the system of ODEs which is given in the Figure 6.1(e), where the concentration of the protein in the state (x_l, x_c, x_r) is denoted by $[x_l, x_c, x_r]$. Since the state of the site l does not control the evolution of the state of the site r, and conversely, we can abstract away the correlation between the states of the sites l and r. To do this, we cut the chemical species into fragments, each fragment documenting either the sites l and c, or the sites c and r.

Such a fragmentation defines a linear change of variables. Indeed we can define the concentration of a fragment, as the linear combination of the concentration of the chemical species in which this fragment occurs. For instance, the concentration of the fragment which documents the sites l and c, and where both these sites are activated is equal to the sum of the concentrations of the chemical species (p,p,u) and (p,p,p). Applying this change of variables, we get the reduced system which is given in the Figure 6.1(f). We notice that the number of variables in the two systems is the same, because of the simplicity of the example. In practice, abstracting away a correlation

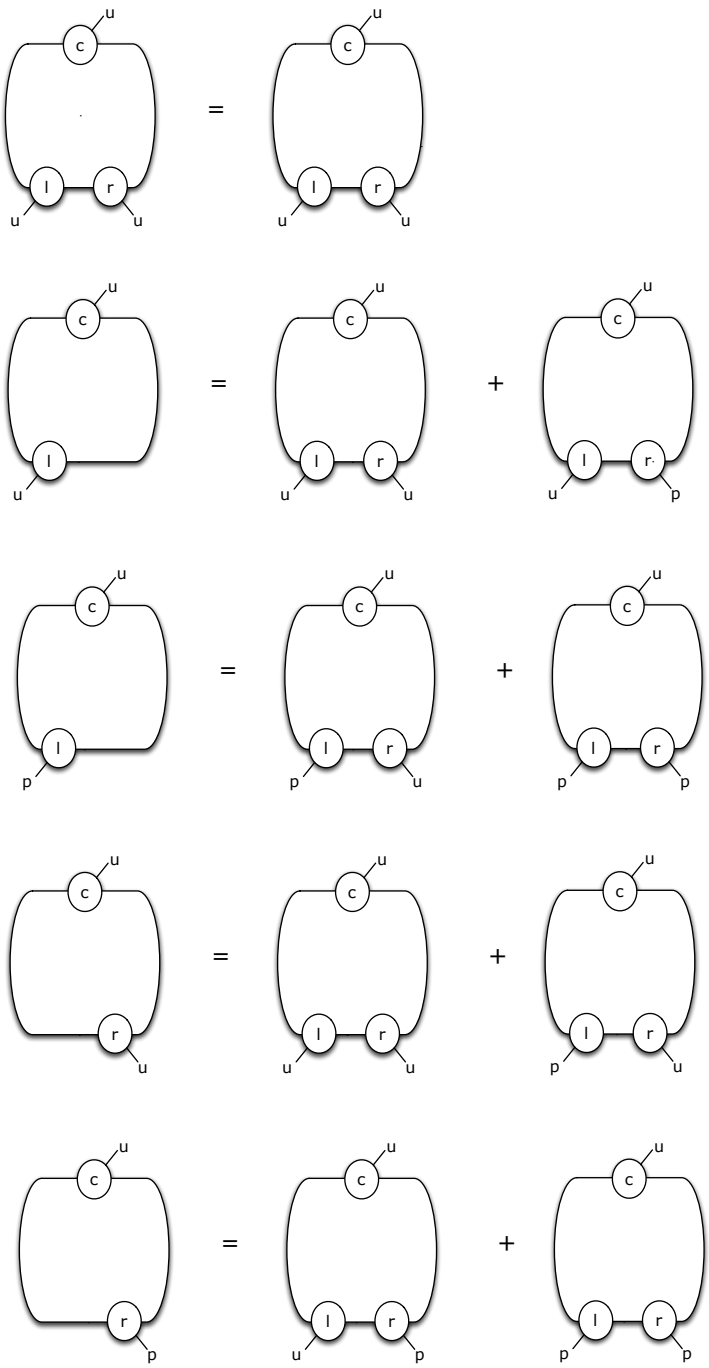


Figure 6.2: Change of variables.

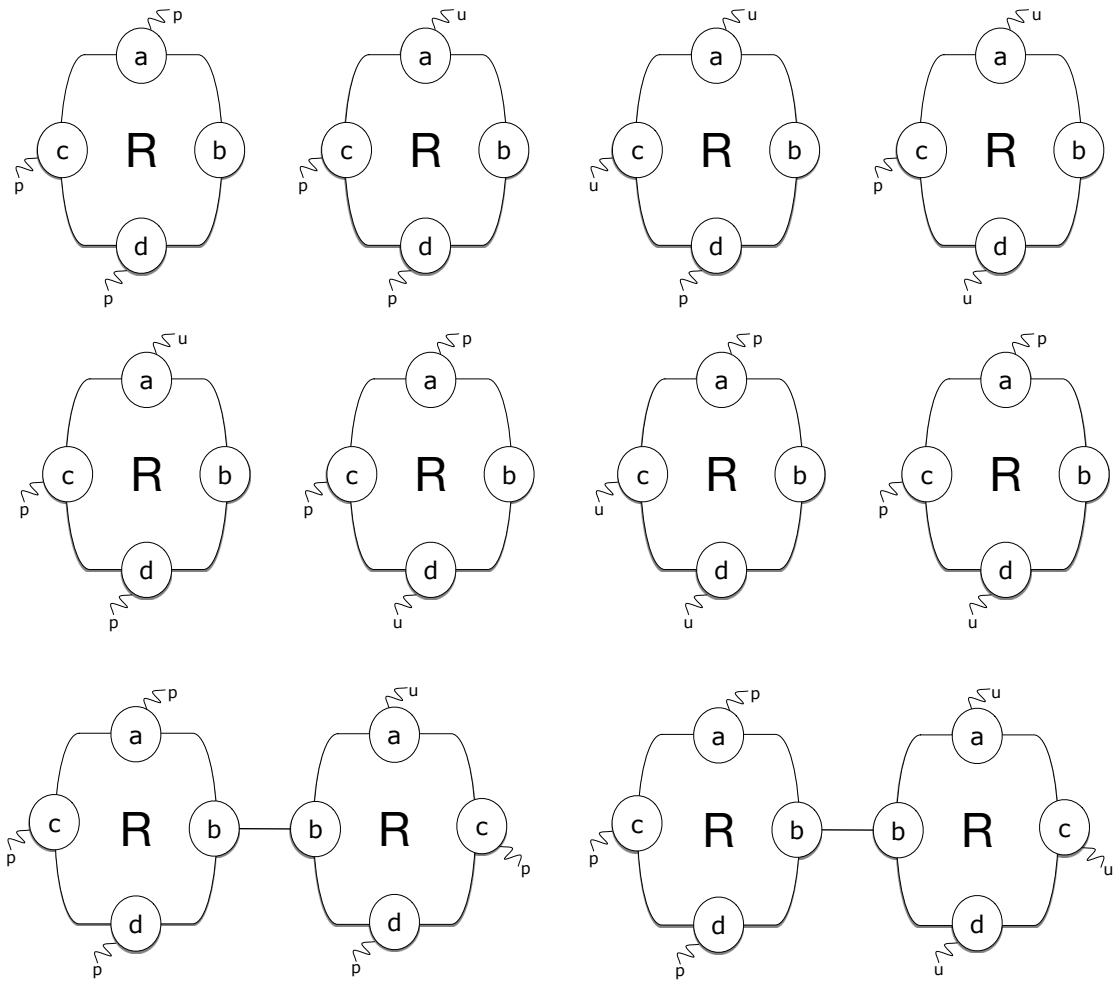


Figure 6.3: A set of species.

reduces a lot the number of variables.

We have seen in this example, that an over-approximation of the flow of information between the sites of chemical species, can be used to identify useless correlations, which can be used to discover appropriate change of variables.

6.2 Contact map and annotated contact map

Given a set of species $\mathcal{V}$, the contact map (CM) associated to the set $\mathcal{V}$ is a summary of all the bindings found in any species in $\mathcal{V}$. We can look at it as a symbolic representation of the set of all the species.

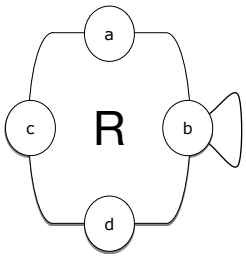


Figure 6.4: Contact map for the set of species in 6.3.

A contact map is formally defined in the following way.

Definition 6.2.1 (Contact map)**.** *Given a signature Σ, an initial set of species $\mathcal{V}$ and a set of rules $\mathcal{R}$, a contact map CM is a non-oriented graph where:*

1. *the nodes are the pairs $(A, x) \in \mathcal{A} \times \mathcal{S}$ such that $x \in \Sigma(A)$;*
2. *there is an edge between two nodes (A, x) and (B, y) if and only if there is a link between two sites (A, i, x) and (B, j, y) in at least one species in the original set $\mathcal{V}$ or in the right hand side of at least one rule in $\mathcal{R}$, for two agent identifiers $i, j \in \mathbb{N} \cup \overline{\mathbb{N}}$.*

Figure 6.4 shows the contact map associated to the set of species Figure 6.3.

We propose to annotate a contact map with an over-approximation of the flow of information between the different regions of chemical species. The main idea is to identify correlations between the states of the sites, which can be safely abstracted away, because they have no influence on the behaviour of the state of the other sites. This way, the so-obtained annotated contact map (aCM) will be used as a symbolic representation of the set of fragments of chemical species, the concentration of which will be the variables of our reduced system.

Definition 6.2.2 (Annotated contact map)**.** *An annotated contact map (aCM) is given by a contact map and a binary (oriented) relation over its nodes. The relation may relate two pairs (A, x) and (B, y) if:*

1. *$A = B$ and $x \neq y$,*
2. *or there is an edge between (A, x) and (B, y) in the contact map.*

In such case, we say that there is an arc in the aCM from the site x of A to the site y of B.

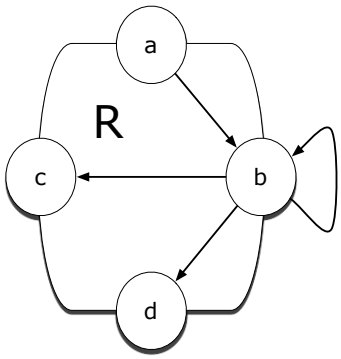


Figure 6.5: An annotated contact map.

6.3 Fragments

Fragments are well chosen pattern components, which can be derived from an annotated contact map.

Definition 6.3.1 (Path in a pattern)**.** *A* path *in a pattern E is a finite sequence of site instances $p := (A_k, i_k, x_k)_{1 \leqslant k \leqslant n}$ such that:*

1. *for any k between 1 and $n-2$, $(A_k, i_k, x_k) \neq (A_{k+2}, i_{k+2}, x_{k+2})$;*

2. *for any k between 1 and $n-1$, either $(A_k, i_k) = (A_{k+1}, i_{k+1})$ and $x_k \neq x_{k+1}$, or the instances of the site (A_k, i_k, x_k) and $(A_{k+1}, i_{k+1}, x_{k+1})$ are bound together in the pattern E.*

In such a case, n is called the length of the path p, and for any k between 1 and $n-1$, $((A_k, i_k, x_k), (A_{k+1}, i_{k+1}, x_{k+1}))$ is called an arc in p.

Definition 6.3.2 (Compatible path)**.** *A path p in a pattern E is* compatible *with the aCM, if and only if, for any arc $((A, i, x), (A', i', x'))$ in p, there is an arc from the node (A, x) and the (A', x') in the aCM.*

As stated by the two following lemmas, compatible paths can be composed and the image of a compatible path by an embedding is a compatible path.

Lemma 6.3.3 (Path composition)**.** *If there exists two paths p_1 and p_2 in E, both compatible with the aCM and, respectively from a site instance (A, i, x) to a site instance (A', i', x'), and from the site instance (A', i', x') and a site instance $(A", i", x")$, then there exists a path in E, compatible with the aCM, from the site instance (A, i, x) to the site instance $(A", i", x")$.*

Proof. We prove the Lemma 6.3.3 by induction over the length of p_1. Let us write $p_1 = (A_k, i_k, x_k)_{1 \leqslant k \leqslant n}$ and $p_2 = (A'_k, i'_k, x'_k)_{1 \leqslant k \leqslant n'}$. If $n = 0$ or $n' = 0$, then p_1 or p_2 is a path in E compatible with the aCM from the site instance (A, i, x) to the site instance $(A", i", x")$; else if $(A_{n-1}, i_{n-1}, x_{n-1}) \neq (A'_2, i'_2, x'_2)$, then $(A_1, i_1, x_1), \ldots, (A_n, i_n, x_n), (A'_2, i'_2, x'_2), \ldots, (A'_{n'}, i'_{n'}, x'_{n'})$ is a path in E, compatible with the aCM; otherwise we apply the induction hypothesis with the paths $(A_k, i_k, x_k)_{1 \leqslant k < n}$ and $(A'_k, i'_k, x'_k)_{1 < k \leqslant n'}$. □

Lemma 6.3.4 (Path image)**.** *Let ϕ be an embedding between two patterns E and E' and $(A_k, i_k, x_k)_{1 \leqslant k \leqslant n}$ be a path in E, compatible with the aCM. Then $(A_k, \phi(A_k, i_k), x_k)_{1 \leqslant k \leqslant n}$ is a path in E' which is compatible with the aCM.*

Proof. Let $(A_k, i_k, x_k)_{1 \leqslant k \leqslant n}$ be a path in E that is compatible with the aCM and let us consider $(A_k, \phi(A_k, i_k), x_k)_{1 \leqslant k \leqslant n}$. We need to show that for every k between 1 and $n - 1$, there is an arc from the site x_k of the agent A_k to the site x_{k+1} of the agent A_{k+1}.

Since $(A_k, i_k, x_k)_{1 \leqslant k \leqslant n}$ is a path in E that is compatible with the aCM. For every k between 1 and $n-1$, there is an arc in the aCM between the site (A_k, x_k) and the site (A_{k+1}, x_{k+1}). □

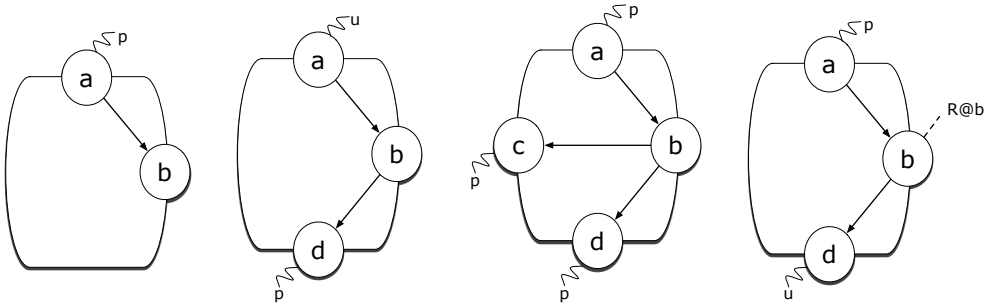
Now we are ready to define the pattern components we were looking for: the fragments. We do it in two steps. Firstly, we define prefragments, as the patterns such that the arcs that are compatible with the aCM form a directed relation over their site instances. Then, we define fragments as the prefragments which are maximal with respect to the embedding ordering.

Definition 6.3.5 (Prefragment)**.** *A* prefragment *is a pattern component E such that there is a site instance (A, i, x) in E, such that, for any site instance (A', i', x') there is a path in E, compatible with the aCM, from (A', i', x') to (A, i, x). In such a case, the site instance (A, i, x) is called a* target *of the prefragment E.*

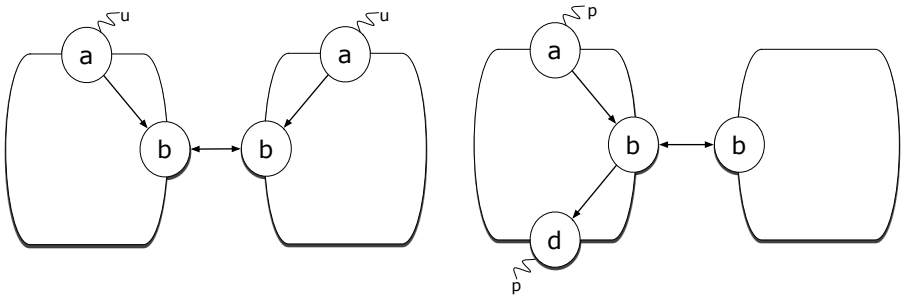
Definition 6.3.6 (Fragment)**.** *A* fragment *is a prefragment which is maximal for the embedding ordering. So a prefragment F is a fragment whenever for any prefragment F' such that there exists an embedding between F and F', we have $F = F'$.*

As we did for the set of species, we assume that there are no cycle in prefragments.

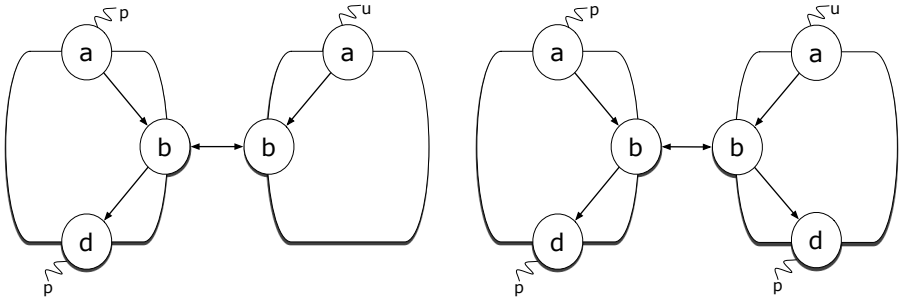
Example 6.3.7. *We consider the aCM in Figure 6.5 and the pattern components that are given in Figure 6.6. We have annotated each pattern component with every path that is compatible with the aCM. Among these pattern*



(a) F_1. (b) F_2. (c) F_3. (d) F_4.



(e) F_5. (f) F_6.



(g) F_7. (h) F_8.

Figure 6.6: Some annotated pattern components.

components, only F_1, F_2, F_4, F_5, F_6 and F_7 are prefragments. Yet, we can notice that:

- *F_1 can be embedded into F_2;*
- *F_4 can be embedded into F_7;*
- *F_6 can be embedded into F_7.*

So, F_1, F_4 and F_6 are not fragments.

On the other site, the only way to refine F_2 is by adding the site c but what we would obtain would be F_3, which is not a fragment (because it has not a unique target), thus F_2 is a fragment. The same way, the only manner to refine F_7 is by adding the site d, but also in this case the result, F_8, is not a fragment (it does not have a target). So we can conclude that both F_2 and F_7 are fragments.

Remark 6.3.8. *The previous example illustrates precisely the main difference between the framework that is proposed in [25, 28], and the one that is presented in this chapter. In this chapter, each annotated contact map generates an heterogenous set of fragments, in the sense that the agents of a fragment do not have to be cut in the same way. With the present framework, exactly one agent in the fragments that embed into a dimer, documents the state of one of the site among c and d, whereas in the framework proposed in [25, 28], both agents document the state of one of the site among c and d. As a consequence, the reduced model that we obtain with the present framework is always smaller than the one that we obtain with the framework described in [25, 28].*

Given a pattern component E and a state ρ over $\mathcal{V}$, we recall that the concentration $[E]_\rho$ of the pattern E in the state ρ, is defined as:

$$[E]_\rho = \sum_{v \in \mathcal{V}} \frac{\mathbf{card}([E, v])}{\mathbf{card}([E, E])} \rho(v).$$

and that the corrected concentration $(\!|E|\!)_\rho$ of a pattern E is defined as $[E]_\rho \times \mathbf{card}([E, E])$.

We define the set of abstract variables as the set of fragments of chemical species modulo isomorphisms. Formally, $\mathcal{V}^\sharp$ is a set of fragments, such that:

1. for each fragments F in $\mathcal{V}^\sharp$, there exists $v \in \mathcal{V}$ such that F embeds in v, and
2. for any pair (F_1, F_2) of fragments in $\mathcal{V}^\sharp$, if F_1 embeds into F_2 then $F_1 = F_2$.

The set of concrete states $\mathcal{V} \to \mathbb{R}$ over $\mathcal{V}$ and the set of (abstract) states $\mathcal{V}^\sharp \to \mathbb{R}$ over $\mathcal{V}^\sharp$ are related via the abstraction function ψ which is defined as:

$$\psi(\rho)(v^\sharp) = [v^\sharp]_\rho.$$

The (pre)fragments and the abstraction function ψ enjoy the following properties.

Proposition 6.3.9 (Orthogonal decomposition.)**.** *Let F be a prefragment. The concentration of a prefragment F can be expressed as a linear combination with positive coefficients of the concentration of some fragments.*

Proof. Let us prove the following equivalent property: $(\!|E|\!)_\rho$ can be expressed as a linear combination with positive coefficients of the corrected concentration of $(\!|F_i|\!)_\rho$ of some fragments.

The proof works iteratively. At each step, a prefragment E' will be replaced with a multi-set of more refined prefragments, while preserving the overall corrected concentration, until we obtain a multi-set of fragments. If E' does not embed into a species in $\mathcal{V}$, we remove it. Otherwise E' has to be refined.

A pattern can be refined into a multi-set of patterns while preserving the overall correcting concentration, by using the following rewrite steps. We can refine the internal state of a site which misses one with any internal state in $\mathbb{I}$, or refine a binding state '$-$' with each potential binding type (according to the CM). Moreover, a fresh site can be added in the interface of an agent A_i with the binding state '?' if it belongs to $\Sigma_{\lambda(A)}$, and with no binding state otherwise. Lastly, if E' constains a site instance annotated with a binding type, one could replace it with a bond to an existing site (if its binding state allows it), to fresh site in existing agents, or to a site in a fresh agent.

We are left to show that, whenever a prefragment E' is not a fragment, then there always exists a rewrite step which replaces E' into a multi-set of prefragments. Only the steps which add a fresh site instance raise an issue. Thus, let us assume that no other rewrite step can apply. We consider an embedding ϕ between E' and a fragment F. We assume that there exists a target (A, i, x) in F, which has no antecedent by ϕ. Let us consider (A', i', x') a target in E'. Then we can consider a path in F compatible with the aCM from the site instance $(A', \phi(A', i'), x')$ to the site instance (A, i, x). The first site in this path which has no antecedent by ϕ can be aded to E' and, by construction, it is a target of the result. Otherwise, there exists a site instance (A, i, x) in E' such that (A, i, x) is a target in E' and $(A, \phi(A, i), x)$ a target in F and there exists a path in F compatible with the aCM from the site $(A, \phi(A, i), x)$ and a site having no antecedent by ϕ. The first site

having no antecedent can be added to E', and (A, i, x) is still a target of the result.

Thus we have a rewriting strategies, where all intermediar steps are multi-set of prefragments. The set of prefragments which can be embedded into chemical species in $\mathcal{V}$ is finite (since $\mathcal{V}$ is finite), which ensures the termination of our iteration. □

Proposition 6.3.10 (Divergence preservation)**.** *We have:*

$$(\mathcal{V} \to \mathbb{R}) \xrightarrow{\psi} (\mathcal{V}^\sharp \to \mathbb{R}).$$

Proof. By construction, ψ is a linear function which maps any state over $\mathcal{V}$ to a state over $\mathcal{V}^\sharp$. Let us prove that ϕ preserves the divergence of sequences. Let us consider a sequence $(\rho_n)_{n\in\mathbb{N}}$ of states over $\mathcal{V}$ such that the sequence $(||\rho_n||)_{n\in\mathbb{N}}$ diverges. Since $\mathcal{V}$ is a finite set, there exists a variable $v \in \mathcal{V}$ such that the sequence $(\rho_n(v))_{n\in\mathbb{N}}$ diverges toward $+\infty$ (by definition, in a state ρ, $\rho(v) \geqslant 0$). Let us take an instance (A, i, x) of a site in v, we define ι as its internal state and λ as 'ϵ' if the site is free, or as its binding type $B@y$ otherwise. The pattern $E := A_1(x_\iota^\lambda)$ is a prefragment. In given a mixture, the number of embeddings of E is greater than the number of embeddings of v. Moreover, by the Proposition 6.3.9, the number of embeddings of E in a mixture E' can be expressed as a linear combination with positive coefficients of the number of embeddings of some fragments $F_1, \ldots, F_k$ in E'. As a consequence, for at least one of the fragments, let us say F_j, the sequence $(\rho_n(F_j))_{n\in\mathbb{N}}$ diverges as well. Thus the sequence $(||\psi(\rho_n)||^\sharp)_{n\in\mathbb{N}}$ diverges as well. □

6.4 Flow analysis

In this section we define some criteria to ensure that the aCM is a sound over-approximation of the information flow.

6.4.1 Trivial rules

Some specific rules induce no flow of information. We say that a rule is *trivial*, if it has the following form:

$$\mathtt{A_1}(\mathtt{a}^{\mathtt{B_2@b}}), \mathtt{B_2}(\mathtt{b}^{\mathtt{A_1@a}}) \xrightarrow{\mathtt{k}} \mathtt{A_1}(\mathtt{a}), \mathtt{B_2}(\mathtt{b}).$$

Thus a trivial rule releases a bond without testing any other information. We could extend the class of trivial rules, but we do not do it for the sake of simplicity.

6.4.2 Side effects

We remind that a system of rules has side effects if it is possible to modify the state of the sites that are not documented in the left hand side of the rules. More precisely, in the two following cases:

1. when we remove an agent without checking that all its sites are free;

2. when we remove a bound denoted by a binding type or the symbol '−'.

In the first case a side effect is possible (may semantics), in the latter one a side effect is sure (must semantics).

So, given a rule r, $\mathcal{MAY}(r)$ denotes the set of sites that may raise side effects when the rule r is applied whereas $\mathcal{MUST}(r)$ denotes the set of sites that raise side effects whenever the rule r is applied.

Definition 6.4.1 (Partner of a site)**.** *Let (A, i, x) be a site instance. We say that (A', x') is a partner of (A, i, x) if there is an edge between the nodes (A, x) and (A', x') in the contact map.*

6.4.3 Valid annotation

In the following, we say that a rule r modifies a site instance s, if s is a site instance in the lhs of the rule r, and if either s does not belong to the rhs of the rule r (that is to say that its agents have been removed by the rule r), or if it occurs in the rhs of the rule r but with a different state.

Definition 6.4.2 (aCM)**.** *The annotated contact map (aCM) is valid with respect to a rule set $\mathcal{R}$ if it satisfies the following constraints.*

1. *Direct flow:*

   (a) *every path in the lhs of a (non trivial) rule r that ends in a site instance which is modified by the rule r is compatible with the aCM;*

   (b) *for every rule r and every site $(A, i, x) \in \mathcal{MAY}(r)$, there is an arc in the aCM from the node (A, x') to the node (A, x), for any site $x' \in \mathcal{S}$ such that the site instance (A, i, x') occurs in the lhs of the rule r;*

   (c) *for every rule r, every site $(A, i, x) \in \mathcal{MAY}(r) \cup \mathcal{MUST}(r)$, and any potential partner (A', x') of the site (A, x) in the CM, there is an arc from the site (A, x) to the site (A', x') in the aCM.*

2. *Indirect flow: for any pattern component c in the lhs of a non trivial rule r, there exists a site instance (A, i, x) such that for any site instance (A', i', x') in c, there is a path from x to (A, i, x) that is compatible with the aCM.*

Intuitively, *direct flows* describe the flow of information between the sites that are tested (because they occur in the lhs of a rule) and the sites that are modified. *Indirect flows* handle with the pattern components which are not modified: the concentration of these patterns regulates the speed of rule application.

6.5 Reduced model

We recall the notations of Chapter 5.

We assume that the model is made of the following set of rules $\mathcal{R}$:

$$\begin{array}{lcll} r_1 & := & c_{1,1}, \ldots, c_{\omega(1),1} \rightarrow p_{1,1}, \ldots, p_{\theta(1),1} & k_1 \\ r_2 & := & c_{1,2}, \ldots, c_{\omega(2),2} \rightarrow p_{1,2}, \ldots, p_{\theta(2),2} & k_2 \\ \ldots & := & \ldots & \\ r_h & := & c_{1,h}, \ldots, c_{\omega(h),h} \rightarrow p_{1,h}, \ldots, p_{\theta(h),h} & k_h \end{array}$$

where for every rule r_ι, $\omega(\iota)$ returns the number of components in the left hand side and $\theta(\iota)$ returns the number of component in the right hand side.

We denote by $\mathbb{F}$ the semantics function, which maps each state $\rho : \mathcal{V} \rightarrow \mathbb{R}$ to the function of the concentration derivatives $\mathcal{V} \rightarrow \mathbb{R}$

$$\begin{aligned} \mathbf{card}([P,P]) \times \mathbb{F}(\rho)(P) = & \sum_{\iota=1}^{h} \sum_{M \in \overleftarrow{\mathcal{M}}(\iota,P)} \prod_{\chi=1}^{\omega(\iota)} (\!|\overline{c}_{\chi,\iota,F,M}|\!)_\rho \\ & - \sum_{\iota=1}^{h} \gamma_\iota \sum_{j=1}^{\omega(\iota)} \sum_{M \in \overrightarrow{\mathcal{M}}(\iota,j,P)} (\!|c_{j,\iota} \cup_M P|\!)_\rho \prod_{\substack{\chi=1 \\ \chi \neq j}}^{\omega(\iota)} (\!|c_{\chi,\iota}|\!)_\rho \end{aligned}$$

where for every rule index ι, every position j,

- $\overrightarrow{\mathcal{M}}(\iota, j, P)$ denotes the set of the potential extended gluings (e.g. see Definition 5.3.8) between the pattern component $c_{j,\iota}$ and the pattern P;
- $\overleftarrow{\mathcal{M}}(\iota, P)$ is the set of the extended preconditions of the refinements of the rule $rule_\iota$ according to the overlap between the rhs of the rule $rule_\iota$ and the pattern P;
- and for every $M \in \overleftarrow{\mathcal{M}}(\iota, P)$, $(\overline{c}_{\chi,\iota,F,M})_{1 \leqslant \chi \leqslant \omega(\iota)}$ is the list of the pattern components in the extended precondition $pred(\iota, P, M)$.

6.5.1 Contribution of proper consumption

We already know, that the pattern components that occur in the lhs of a rule are all prefragments, thus we can express their concentration (and their corrected concentration) as a linear combination of the concentration of the fragments. Let us prove that if F is a fragment, ι a rule index, and j a position, and any way $M \in \overrightarrow{\mathcal{M}}(\iota, j, F)$ to glue the fragment F on the j-th pattern component of the rule $rule_\iota$, then the (corrected) concentration $(\!|c_{j,\iota} \cup_M F|\!)$ of pattern component $(c_{j,\iota} \cup_M F)$ can also be expressed as the linear combination of the concentration of the other fragments.

Trivial rules

We assume that ι is the index of a trivial rule. By definition, the rule $rule_\iota$ has the following form:

$$\mathtt{A}_1(\mathtt{x}^{\mathtt{B}_2@\mathtt{y}}), \mathtt{B}_2(\mathtt{y}^{\mathtt{A}_1@\mathtt{x}}) \to \mathtt{A}_1(\mathtt{x}), \mathtt{B}_2(\mathtt{y}).$$

Hence, the number $\omega(\iota)$ of connected components in the lhs of the rule $rule_\iota$ is equal to 1, and j is equal to 1 as well.

We consider $(\mathscr{A}, B) \in \overrightarrow{\mathcal{M}}(\iota, j, F)$ an extended gluing between the pattern component $c_{1,\iota}$ and the fragment F.

The rule $rule_\iota$ has no side effect.

As a consequence, we have $B = \varnothing$.

By Definition 5.3.8, it follows that $\mathscr{A} \neq \varnothing$.

We consider two cases according to whether or not the set $\mathscr{A}$ is a singleton.

1. Let us assume that the set $\mathscr{A}$ is a singleton.

   It follows that the (extended) gluing between c_1 and F is obtained (up to isomorphism) by replacing a binding type carried by a site instance s, by a link to a site in a fresh agent.

   We wonder what are the different possibilities to refine the binding type of the site instance s in the fragment F. The site s might be bound to a site instance in F, but the so-refined pattern would contain a cycle, that is to say, since we have assumed that no species contains cycles, that its (corrected) concentration would be equal to 0. The only remaining case is the one where the site s is bound to a site in a fresh agent. In this case, the refined pattern is isomorphic to the pattern $c_{1,\iota} \cup_{\mathscr{A}} F$.

   We conclude that:

   $$(\!|c_{1,\iota} \cup_{(\mathscr{A},B)} F|\!) = (\!|F|\!).$$

2. Let us assume that the set $\mathscr{A}$ contains two elements.

   Let us write $\mathscr{A} = \{(1, i_1), (2, i_2)\}$.

   In that case, either there is a link between the site instances (A, i_1, x) and (B, i_2, y) in the fragment F, or not.

   (a) If there is a link between the site instances (A, i_1, x) and (B, i_2, y) in the fragment F, then $c_{1,\iota} \cup_{(\mathscr{A},B)} F$ is isomorphic to the pattern component F.

   (b) Otherwise, the pattern component $c_{1,\iota} \cup_{(\mathscr{A},B)} F$ contains a cycle, that is to say, since we have assumed that no species contains cycles, that its (corrected) concentration is equal to 0.

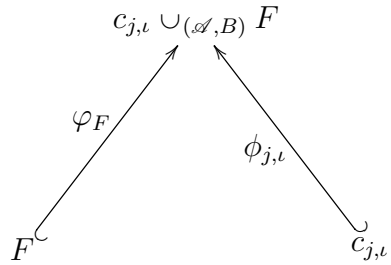
In all case, we have been able to express the corrected concentration $(\![c_{1,\iota} \cup_{(\mathscr{A},B)} F]\!)$ as a linear combination of the corrected concentration of some fragments.

Non-trivial rules

We assume that ι is the index of a non trivial rule. We consider an extended gluing $(\mathscr{A}, B) \in \overrightarrow{\mathcal{M}}(\iota, j, F)$, between the pattern component $c_{j,\iota}$ and a prefragment F.

Thanks to Constraint 2 of Definition 6.4.2, the pattern component $c_{\chi,\iota}$ is a prefragment, for every position χ such that $j \neq \chi$. We are left to express the (corrected) concentration of the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$.

We consider the embedding φ_F from the fragment F to the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ and $\phi_{j,\iota}$ the embedding from the pattern component $c_{j,\iota}$ and the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$, that have been used to define the extended gluing $c_{j,\iota} \cup_{(\mathscr{A},B)} F$.



The pattern component F is a fragment, thus there exists a site instance (A_t, i_t, x_t) in F that is a target. We consider two cases according to whether the set $\mathscr{A}$ is empty, or not.

1. We assume that the set $\mathscr{A}$ is not empty.

   By Definition 5.3.8, the set B is empty.

   That is to say that: $c_{j,\iota} \cup_{(\mathscr{A},B)} F = c_{j,\iota} \cup_{\mathscr{A}} F$.

   By Definition 5.3.5, for every site instance (A, i, x) in $c_{j,\iota} \cup_{\mathscr{A}} F$, we know that either there exists an agent identifier i' such that (A, i', x) is a site instance in F and $i = \varphi_F(A, i')$, or that there exists an agent identifier i' satisfying (A, i', x) is a site instance in $c_{j,\iota}$ and $i = \phi_{j,\iota}(A, i')$.

   We propose to show that the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ is a target in the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$.

   Let (A_s, i_s, x_s) be a site instance in the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$.

   (a) We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in F and $i_s = \varphi_F(A, i')$.

   Since (A_t, i_t, x_t) is a target in F, there is a path in the pattern component F from the site instance (A_s, i'_s, x_s) to the site instance (A_t, i_t, x_t) that is compatible with the aCM.

   By Lemma 6.3.4, there exists a path in the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

   (b) We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in $c_{j,\iota}$ and $i_s = \phi_{j,\iota}(A, i')$.

   By Definition 5.3.8, there exists a site instance (A_m, i_m, x_m) in $c_{j,\iota}$ and an agent identifier i'_m such that:

   - (A_m, i'_m, x_m) is a site instance in F;
   - $\varphi_F(A_m, i'_m) = \phi_{j,\iota}(A_m, i_m)$;
   - the site instance (A_m, i_m, x_m) is modified by the rule $rule_\iota$.

   By Constraint 1a of Definition 6.4.2, there is a path in the pattern component $c_{j,\iota}$ from the site instance (A_s, i', x_s) to the site instance (A_m, i_m, x_m) that is compatible with the aCM.

   By Lemma 6.3.4, there exists a path in the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$ from the site instance (A_s, i_s, x_s) to the site instance $(A_m, \phi_{j,\iota}(A_m, i_m), x_m)$ that is compatible with the aCM.

   By the previous case, we know that there is a path in the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$ from the site instance $(A_m, \varphi_F(A_m, i'_m), x_m)$ to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

Since $\varphi_F(A_m, i'_m) = \phi_{j,\iota}(A_m, i_m)$, by Lemma 6.3.3, there exists a path in the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

2. We assume that the set $\mathscr{A}$ is empty.

   The set B is a singleton that we write: $((A_{B_1}, i_{B_1}, x_{B_1}), (A_{B_2}, i_{B_2}, x_{B_2}))$.

   By Definition 5.3.5, we know that for any site instance (A, i, x) in $c_{j,\iota} \cup_{(\mathscr{A},B)} F$, at least one of the following condition is satisfied:

   - either there exists an agent identifier i' such that (A, i', x) is a site instance in F and $i = \varphi_F(A, i')$,
   - or there exists an agent identifier i' such that (A, i', x) is a site instance in $c_{j,\iota}$ and $i = \phi_{j,\iota}(A, i')$,
   - or $(A, i, x) = (A_{B_1}, \phi_{j,\iota}(A_{B_1}, i_{B_1}), x_{B_1})$.

   Let us show that the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ is a target in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$.

   Let (A_s, i_s, x_s) be a site instance in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$.

   (a) We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in F and $i_s = \varphi_F(A, i')$.

   Since (A_t, i_t, x_t) is a target in F, there is a path in the pattern component F from the site instance (A_s, i'_s, x_s) to the site instance (A_t, i_t, x_t) that is compatible with the aCM.

   By Lemma 6.3.4, there exists a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

   (b) We assume that $(A_s, i_s, x_s) = (A_{B_1}, \phi_{j,\iota}(A_{B_1}, i_{B_1}), x_{B_1})$.

   By Constraint 1c of Definition 6.4.2, there is an arc in the aCM between the site (A_s, x_s) and the site (A_{B_2}, x_{B_2}).

   As a consequence, there exists a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance (A_s, i_s, x_s) to the site instance $(A_{B_2}, \varphi_F(A_{B_2}, i_{B_2}), x_{B_2})$ that is compatible with the aCM.

   By the previous case, we already know that there exists a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance $(A_{B_2}, \varphi_F(A_{B_2}, i'_{B_2}), x_{B_2})$ to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

By Lemma 6.3.3, we can conclude that there exists a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

(c) We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in $c_{j,\iota}$ and $i_s = \phi_{j,\iota}(A, i')$.

- If $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MAY}(rule_\iota)$.
  The agent instance (A_{B_1}, i_{B_1}) has at least one site instance (otherwise, the pattern component $c_{j,\iota}$ would have been made of a single agent with no site instance, which would be absurd since (A_s, i', x_s) is a site instance in $c_{j,\iota}$).
  Let x_w be a site name, such that the triple (A_{B_1}, i_{B_1}, x_w) is a site instance in the pattern $c_{j,\iota}$.
  We have $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MAY}(rule_\iota)$, so the agent instance (A_{B_1}, i_{B_1}) is removed by the rule $rule_\iota$. As a consequence, the site (A_{B_1}, i_{B_1}, x_w) is modified by the rule $rule_\iota$.
  By Constraint 1a of Definition 6.4.2, there is a path in the pattern component $c_{j,\iota}$ from the site instance (A_s, i', x_s) to the site instance (A_{B_1}, i_{B_1}, x_w) that is compatible with the aCM.
  By Lemma 6.3.4, there is a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance (A_s, i_s, x_s) to the site instance $(A_{B_1}, \phi_{j,\iota}(A_{B_1}, i_{B_1}), x_w)$ that is compatible with the aCM.
  By Constraint 1b of Definition 6.4.2, there is an arc between the site (A_{B_1}, x_w) and the site (A_{B_1}, x_{B_1}) in the aCM.
  As a consequence, there is a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance $(A_{B_1}, \phi_{j,\iota}(A_{B_1}, i_{B_1}), x_w)$ to the site instance $(A_{B_1}, \phi_{j,\iota}(A_{B_1}, i_{B_1}), x_{B_1})$ that is compatible with the aCM.
  By the previous case, we know that there exists a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance $(A_{B_1}, \phi_{j,\iota}(A_{B_1}, i_{B_1}), x_{B_1})$ to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.
  By Lemma 6.3.3, there exists a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.
- If the site instance $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MUST}(rule_\iota)$.
  The site instance $(A_{B_1}, i_{B_1}, x_{B_1})$ is modified by the rule $rule_\iota$.
  By Constraint 1a of Definition 6.4.2, there is a path in the

pattern component $c_{j,\iota}$ from the site instance (A_s, i', x_s) to the site instance $(A_{B_1}, i_{B_1}, x_{B_1})$ that is compatible with the aCM.
By Lemma 6.3.4, we can conclude there is a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance (A_s, i_s, x_s) to the site instance $(A_{B_1}, \phi_{j,\iota}(A_{B_1}, i_{B_1}), x_{B_1})$ that is compatible with the aCM.
By the previous case, we know that there exists a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance $(A_{B_1}, \phi_{j,\iota}(A_{B_1}, i_{B_1}), x_{B_1})$ to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.
By Lemma 6.3.3, there exists a path in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

As a consequence, the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ is a target for the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$. That is to say that the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$ is a prefragment.

In all cases, we have expressed the corrected concentration of the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$ as a linear combination of the (corrected) concentration of some fragments.

6.5.2 Contribution of proper production

Let us prove that for every rule index ι, every fragment F, every way $M \in \overleftarrow{\mathcal{M}}(\iota, F)$ of producing the fragment F with the rule $rule_\iota$, we can express, for every position χ, the (corrected) concentration $(\!|\overline{c}_{\chi,\iota,F,M}|\!)$ of the χ-th pattern component of the corresponding refinement of the rule $rule_\iota$ as a linear combination of the corrected concentration of some fragments.

Trivial rules

Let ι be the index of a trivial rule and F be a fragment. By definition, the rule $rule_\iota$ has the following form:

$$\mathtt{A_1}(\mathtt{x}^{\mathtt{B_2@y}}), \mathtt{B_2}(\mathtt{y}^{\mathtt{A_1@x}}) \to \mathtt{A_1}(\mathtt{x}), \mathtt{B_2}(\mathtt{y}).$$

Hence, the number $\omega(\iota)$ of pattern components in the lhs of the rule $rule_\iota$ is equal to 1.

We consider a pair $(\mathscr{A}, B)$ in the set $\overleftarrow{\mathcal{M}}(\iota, F)$.

The rule $rule_\iota$ has no side effect.

As a consequence, we have $B = \varnothing$.

By Definition 5.4.2, it follows that $\mathscr{A} \neq \varnothing$.

We consider two cases according to whether or not the set $\mathscr{A}$ is a singleton.

1. Let us assume that the set $\mathscr{A}$ is a singleton.

   It follows that the fragment F contains a site instance s that is free. Then the pattern component $\overline{c}_{1,\iota,F,M}$ is obtained (up to isomorphism) by binding in F the site instance s to a link to a site in a fresh agent.

   We denote as F' the pattern that is obtained by replacing, in the fragment F the binding state of the site instance s with the binding type that matches the link that is created by the rule.

   Let us prove that the pattern F' is a prefragment.

   - Indeed F and F' only differ by the binding state of s that is free in F and equipped with a binding type in F'. Thus, F and F' have the same set of agents, the same set of site instances, and the same set of links. it follows that any path in F is also a path in F'.

     Let t be a target in F.

     Let us show that t is also a target of F'.

     Let s' be a site instance in F'. Since t is a target in F, there is a path $p = (A_{i_k}, i_k, x_k)_{1 \leqslant k \leqslant n}$ in F from the site s' to the the site t, that is compatible with the aCM.

     Thus, for any k between 1 and $n-1$, there is an arc from the node (A_{i_k}, x_k) and the node $(A_{i_{k+1}}, x_{k+1})$ in the aCM.

     Since the path p is also a path in F', we can conclude that the path p in F' is compatible with the aCM.

     We have proved that the site instance t is a target in F'.

     Thus F' is prefragment.

   We wonder what are the different possibilities to refine the binding type of the site instance s in the prefragment F'. The site s might be bound to a site instance in F', but the so-refined pattern would contain a cycle, that is to say, since we have assumed that no species contains cycles, that its (corrected) concentration would be equal to 0. The only remaining case is the one where the site s is bound to a site in a fresh agent. In this case, the refined pattern is isomorphic to the pattern $\overline{c}_{1,\iota,F,M}$.

We conclude that

$$(\!|\overline{c}_{1,\iota,F,M}|\!) = (\!|F'|\!).$$

2. Let us assume that the set $\mathscr{A}$ contains two elements.

   Then the pattern component $\overline{c}_{1,\iota,F,M}$ is obtained (up to isomorphism) by binding in F two site instances. As a consequence the pattern component $\overline{c}_{1,\iota,F,M}$ has a cycle.

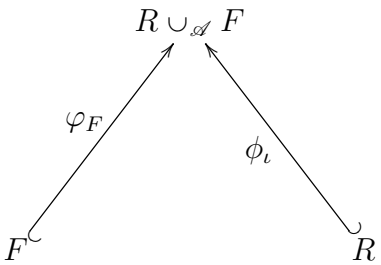
   Since we have assumed that species has no cycle, we can conclude that the concentration of the pattern component $\overline{c}_{1,\iota,F,M}$ is equal to 0.

In all case, we have been able to express the corrected concentration $(\!|\overline{c}_{1,\iota,F,M}|\!)$ as a linear combination of the corrected concentration of some fragments.

Non-trivial rules

We assume that ι is the index of a non trivial rule. We denote by $L \to R$ the rule $rule_\iota$ We consider $(\mathscr{A}, B)$ an element of the set $\in \overleftarrow{\mathcal{M}}(\iota, F)$.

We consider φ_F the embedding from the fragment F to the pattern component $R \cup_{\mathscr{A}} F$ and ϕ_ι be the embedding from the pattern R to the pattern component $R \cup_{\mathscr{A}} F$, that we have used to define the gluing between the fragment F and the rhs R of the rule $rule_\iota$.



Without any loss of generality, we assume that the pattern $R \cup_{\mathscr{A}} F$ matches the pattern R, that is to say that the injective substitution ϕ_ι maps pairs (A, i) to the agent identifiers i.

The pattern component F is a fragment, thus there exists a site instance (A_t, i_t, x_t) in F that is a target.

Let χ be a natural number 1 and $\omega(\iota)$.

We want to prove that the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is a prefragment.

1. We assume that the set $\mathscr{A}$ is not empty.

   We denote as F^{-1} the antecedent of F.

Formally, F^{-1} is defined as the pattern that contains every agent instance $(A, \varphi_F(A, i))$ such that (A, i) is an agent instance in F and the agent instance $(A, \varphi_F(A_i))$ has not been created by the application of the rule $rule_\iota$, every site instance $(A, \varphi_F(A, i), x)$ such that (A, i, x) is a site instance in F and the agent instance $(A, \varphi_F(A_i))$ has not been created by the application of the rule $rule_\iota$, and every bond between the two site instances $(A, \varphi_F(A, i), x)$ and $(A', \varphi_F(A', i'), x')$ such that there is a bond between the site instance (A, i, x) and (A', i', x') in F and the bond between the site instance $(A, \varphi_F(A, i), x)$ and $(A', \varphi_F(A', i'), x')$ has not been created by the rule $rule_\iota$.

We denote as F_χ^{-1} the pattern that is made of every pattern component of F^{-1} that contains at least one agent instance $(A, \phi_\iota(A, i))$ for a given site instance (A, i) in $c_{\chi,\iota}$.

Let us prove that any connected component F_{pc}^{-1} of F_χ^{-1} contains a site that has been modified by the rule $rule_\iota$.

- This is the case if F^{-1} is connected (by assumption, since $B = \varnothing$, F and P overlaps on a site that is modified by the rule $rule_\iota$).
- If F_{pc}^{-1} is disconnected, each connected component of it contains a site the binding site of which has changed.

Then, we propose to prove both following intermediary results:

(a) Whenever a pattern component of F_χ^{-1} contains the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$, there exists a path in this pattern component from every site instance of this pattern component to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$, that is compatible with the aCM.

(b) Whenever a pattern component F_{pc}^{-1} of F_χ^{-1} does not contain the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$, then it contains a site instance $(A_\bullet, i_\bullet, x_\bullet)$ such that the following properties are all satisfied:
   - there exists an agent identifier i such that $(A_\bullet, i, x_\bullet)$ is a site instance in F and $i_\bullet = \varphi_F(A_\bullet, i)$;
   - there exists an agent identifier i such that $(A_\bullet, i, x_\bullet)$ is a site instance in $c_{\chi,\iota}$ and $i_\bullet = \phi_\iota(A_\bullet, i)$;
   - for every site instance (A_s, i_s, x_s) of F_{pc}^{-1}, there exists a path in the pattern component F_{pc}^{-1} from the site instance (A_s, i_s, x_s) to the site instance $(A_\bullet, i_\bullet, x_\bullet)$ that is compatible with the aCM.

Let us prove these properties:

(a) Let F_o^{-1} be the pattern component of F_χ^{-1} that contains the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$.

Let (A_s, i_s, x_s) be a site instance in F_o^{-1}.

Let i' be an agent identifier such that (A_s, i', x_s) is a site instance in F and $\varphi_F(A_s, i') = i_s$.

Since the site instance (A_t, i_t, x_t) is a target in F, there exists a path in the pattern component F from the site instance (A_s, i', x_s) to the site instance (A_t, i_t, x_t) that is compatible with the aCM.

By Lemma 6.3.4, there exists a path p in the pattern $P \cup_{\mathscr{A}} F$ from the site instance (A_s, i_s, x_s), to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$, that is compatible with the aCM.

Since the pattern F is acyclic, and both site instances (A_s, i_s, x_s) and $(A_t, \varphi_F(A_t, i_t), x_t)$ belong to the fragment F, it follows that the path p remains in the pattern F_{pc}^{-1}.

Thus there exists a path in the pattern component F_o^{-1} from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

(b) Let F_{pc}^{-1} be a pattern component of F_χ^{-1} that does not contain the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$.

Necessarily, the pattern component F_{pc}^{-1} has been disconnected from the rest of F.

That is to say that it contains a site the binding state of which has changed.

Let $(A_\bullet, i_\bullet, x_\bullet)$ be a site instance in F_{pc}^{-1} that has lost a bond.

Since $(A_\bullet, i_\bullet, x_\bullet)$ is a site instance in F_{pc}^{-1} there exists an agent identifier i such that $(A_\bullet, i, x_\bullet)$ is a site instance in F and $i_\bullet = \varphi_F(A_\bullet, i)$.

It is not possible to create a bond in a rule, without having the two corresponding site instances in the lhs of this rule.

As a consequence, there exists an agent identifier i' such that $(A_\bullet, i', x_\bullet)$ is a site instance in $c_{\chi,\iota}$ and $i_\bullet = \phi_\iota(A_\bullet, i')$.

Since F is acyclic, and by definition of F^{-1}, there is a path in the pattern component F_o^{-1} from every site instance to the site instance $(A_\bullet, i_\bullet, x_\bullet)$, that is compatible with the aCM.

Let (A_s, i_s, x_s) be a site instance in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$, one of the following property is satisfied:

(a) (A_s, i_s, x_s) is a site instance in F_χ^{-1};

(b) there exists an agent identifier i'_s such that (A_s, i'_s, x_s) is $c_{\chi,\iota}$ and $i_s = \phi_\iota(A_s, i'_s)$.

We consider several cases according to whether the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ contains the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$, or not, and according to whether or not it has been modified by the rule $rule_\iota$.

(a) We assume that the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ contains the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$.

We will prove that there is a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

i. If the site instance (A_s, i_s, x_s) belongs to a pattern component of F_χ^{-1} that contains the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$, there is a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

ii. If there exists an agent identifier i such that (A_s, i, x_s) is a site instance of $c_{\chi,\iota}$ and $i_s = \phi_\iota(A_s, i)$.
There exists a site instance (A_m, i_m, x_m) in $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ satisfying the following constraints:

A. there exists an agent identifier $i_{m'}$ such that the site instance $(A_m, i_{m'}, x_m)$ belongs to the pattern component $c_{\chi,\iota}$ and $i_m = \varphi_\iota(A_m, i_{m'})$;

B. the pattern component of F_χ^{-1} that contains the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ also contains the the site instance (A_m, i_m, x_m);

C. the site $(A_m, i_{m'}, x_m)$ is modified by the rule $rule_\iota$.

Then, by Constraint 1a of Definition 6.4.2, there exists a path in the pattern component $c_{\chi,\iota}$ from the site instance (A_s, i, x_s) to the site instance $(A_m, i_{m'}, x_m)$ that is compatible with the aCM.
By Lemma 6.3.4, there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance (A_m, i_m, x_m) that is compatible with the aCM.
By Lemma 6.3.3, and the precedent case, there is a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t), x_t)$ that is compatible with the aCM.

iii. If (A_s, i_s, x_s) belongs to a pattern component of F_χ^{-1} that does not contain the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$.
Then, there exists a site (A_m, i_m, x_m) in $c_{\chi,\iota}$ such that there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_m, \phi_\iota(A_m, i_m), x_m)$ that is compatible with the aCM.
By Lemma 6.3.3, and the precedent case, there is a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t), x_t)$ that is compatible with the aCM.

Thus, the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is a prefragment.

(b) We assume that: $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)} = c_{\chi,\iota}$.
Then, by Constraint 2 of Definition 6.4.2, $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is a prefragment.

(c) We assume that $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)} \neq c_{\chi,\iota}$ and that the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ does not contain the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$.
Let (A_m, i_m, x_m) be a site instance in $c_{\chi,\iota}$ that has been modified by the rule $rule_\iota$.
We will prove that there is a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_m, \phi_\iota(A_m, i_m), x_m)$ that is compatible with the aCM.

i. If there exists an agent identifier i such that (A_s, i, x_s) is a site instance of $c_{\chi,\iota}$ and $i_s = \phi_\iota(A_s, i)$.
By Constraint 1a of Definition 6.4.2, there exists a path in the pattern component $c_{\chi,\iota}$ from the site instance (A_s, i, x_s) to the site instance $(A_m, i_{m'}, x_m)$ that is compatible with the aCM.
By Lemma 6.3.4, there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_m, \phi_\iota(A_m, i_m), x_m)$ that is compatible with the aCM.

ii. If (A_s, i_s, x_s) belongs to a pattern component of F_χ^{-1}.
There exists a site $(A_\bullet, i_\bullet, x_\bullet)$ in $c_{\chi,\iota}$ such that there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_\bullet, \phi_\iota(A_\bullet, i_\bullet), x_\bullet)$ that is compatible with the aCM.
By Lemma 6.3.3, and the precedent case, we conclude that there is a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_m, \phi_\iota(A_m, i_m), x_m)$ that is compatible with the aCM.

Thus, the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is a prefragment.

2. We assume that the set $\mathscr{A}$ is empty.

   The set B is a singleton that we write $((A_{B_1}, i_{B_1}, x_{B_1}), (A_{B_2}, i_{B_2}, x_{B_2}))$.

   Let χ be a natural number between 1 and $\omega(\iota)$.

   (a) If the pattern component $c_{\chi,\iota}$ does not contain the agent instance (A_{B_1}, i_{B_1}).
   Then we have $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)} = c_{\chi,\iota}$.
   By Constraint 2 of Definition 6.4.2, the pattern component $c_{\chi,\iota}$ is a prefragment.
   Thus, $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)} = c_{\chi,\iota}$.
   We can conclude that $(\![\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}]\!) = (\![c_{\chi,\iota}]\!)$.

   (b) If the pattern component $c_{\chi,\iota}$ contains the agent instance (A_{B_1}, i_{B_1}).
   The pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is equal up to isomorphism to the pattern component $(c_{\chi,\iota} \cup_\varnothing F)^{\uparrow}_{\{(A_{B_1},i_{B_1}),(A_{B_2},i_{B_2})\}}$ (that is to say, the disjoint union between $c_{\chi,\iota}$ and F, to which we have added a bond between the site instance $(A_{B_1}, i_{B_1}, x_{B_1})$ and the site instance $(A_{B_2}, i_{B_2}, x_{B_2})$.
   We want to prove that the site $(A_t, \varphi_F(A_t, i_t), x_t)$ is a target in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$.
   Let (A_s, i_s, x_s) be a site instance in $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$.

      i. We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in F and $i_s = \varphi_F(A, i')$.
      We denote as F' the pattern component that is obtained by replacing the binding state of the site instance $(A_{B_2}, i_{B_2}, x_{B_2})$ with '?'.
      Since (A_t, i_t, x_t) is a target in F, there is a path in the pattern component F from the site instance (A_s, i'_s, x_s) to the site instance (A_t, i_t, x_t) that is compatible with the aCM.
      Since F and F' only differ by the state of a site, there is a path in the pattern component F' from the site instance (A_s, i'_s, x_s) to the site instance (A_t, i_t, x_t) that is compatible with the aCM.
      We notice that the injective substitution φ_F induces an embedding between F' and $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$.
      By Lemma 6.3.4, there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

ii. We assume that $(A_s, i_s, x_s) = (A_{B_1}, \phi_\iota(A_{B_1}, i_{B_1}), x_{B_1})$.
By Constraint 1c of Definition 6.4.2, there is an arc in the aCM between the site (A_s, x_s) and the site (A_{B_2}, x_{B_2}).
As a consequence, there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_{B_2}, \varphi_F(A_{B_2}, i_{B_2}), x_{B_2})$ that is compatible with the aCM.
By the previous case, we already know that there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance $(A_{B_2}, \varphi_F(A_{B_2}, i'_{B_2}), x_{B_2})$ to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.
By Lemma 6.3.3, we can conclude that there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

iii. We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in $c_{j,\iota}$ such that $i_s = \phi_\iota(A, i')$.

- If $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MAY}(rule_\iota)$.
The agent instance (A_{B_1}, i_{B_1}) has at least one site instance (otherwise, the pattern component $c_{j,\iota}$ would have been made of a single agent with no site instance, which would be absurd since (A_s, i', x_s) is a site instance in $c_{j,\iota}$).
Let x_w be a site name, such that the triple (A_{B_1}, i_{B_1}, x_w) is a site instance in the pattern $c_{j,\iota}$.
We have $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MAY}(rule_\iota)$, so the agent instance (A_{B_1}, i_{B_1}) is removed by the rule $rule_\iota$. As a consequence, the site (A_{B_1,i_{B_1},x_w}) is modified by the rule $rule_\iota$.
By Constraint 1a of Definition 6.4.2, there is a path in the pattern component $c_{j,\iota}$ from the site instance (A_s, i', x_s) to the site instance (A_{B_1}, i_{B_1}, x_w) that is compatible with the aCM.
By Lemma 6.3.4, there is a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_{B_1}, \phi_\iota(A_{B_1}, i_{B_1}), x_w)$ that is compatible with the aCM.
By Constraint 1b of Definition 6.4.2, there is an arc between the site (A_{B_1}, x_w) and the site (A_{B_1}, x_{B_1}) in the aCM.
Thus, there is a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$

from the site instance $(A_{B_1}, \phi_\iota(A_{B_1}, i_{B_1}), x_w)$ to the site instance $(A_{B_1}, \phi_\iota(A_{B_1}, i_{B_1}), x_{B_1})$ that is compatible with the aCM.
By the previous case, we already know that there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance $(A_{B_1}, \phi_\iota(A_{B_1}, i_{B_1}), x_{B_1})$ to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.
By Lemma 6.3.3, there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

- If the site instance $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MUST}(rule_\iota)$.
  The site instance $(A_{B_1}, i_{B_1}, x_{B_1})$ is modified by the rule $rule_\iota$.
  By Constraint 1a of Definition 6.4.2, there is a path in the pattern component $c_{j,\iota}$ from the site instance (A_s, i', x_s) to the site instance $(A_{B_1}, i_{B_1}, x_{B_1})$ that is compatible with the aCM.
  By Lemma 6.3.4, we can conclude there is a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_{B_1}, \phi_\iota(A_{B_1}, i_{B_1}), x_{B_1})$ that is compatible with the aCM.
  By the previous case, we already know that there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance $(A_{B_1}, \phi_\iota(A_{B_1}, i_{B_1}), x_{B_1})$ to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.
  By Lemma 6.3.3, there exists a path in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ from the site instance (A_s, i_s, x_s) to the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ that is compatible with the aCM.

As a consequence, the site instance $(A_t, \varphi_F(A_t, i_t), x_t)$ is a target for the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$. That is to say that the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is a prefragment

In all cases, we have expressed the (corrected) concentration of the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ as a linear combination of the (corrected) concentration of some fragments.

6.5.3 Balance

We have provided, for any fragment F, an explicit definition of the proper production $\delta^+_\star(F)$ of the fragment F and of the proper consumption $\delta^-_\star(F)$ of the fragment F, as a polynomial expression of the concentration of the other fragments F'.

We consider the function $\mathbb{F}^\sharp$ that maps each abstract state $\rho^\sharp$ from $\mathcal{V}^\sharp$ to $\mathbb{R}$, to the function mapping each fragment F to the value of the expression $\delta^+_\star(F) - \delta^-_\star(F)$. The function $\mathbb{F}^\sharp$ is well-defined.

Proposition 6.5.1. *We have* $\psi \circ \mathbb{F} = \mathbb{F}^\sharp \circ \psi$.

Proof. By construction of $\mathbb{F}^\sharp$. □

We can conclude with the soundness of our model reduction.

Theorem 6.5.2. *The tuple* $(\mathcal{V}, \mathbb{F}, \mathcal{V}^\sharp, \psi, \mathbb{F}^\sharp)$ *is an abstraction.*

6.6 Conclusion

In this chapter, we have designed a static analysis of the flow of information among the sites of species. This abstraction is summarised in a contact map. This contact map is indeed a compact description of the set of all species, that is obtained by fusing every instance of a same agent type. We annotate this contact map, by inspecting the rule set, and detecting when the state of a site may have an influence on the behaviour of other sites. The result of our analysis can be used to define a set of self-consistent fragments, from which we can derive an exact model reduction of the initial system. This result is proved formally, with respect to the initial semantics of the model.

It is worth noting that we have never worked explicitly on the underlying reaction networks. Moreover, our flow analysis takes into account remote controls, when the sites of an agent control the behaviour of some other agents. This is mandatory to get a sound abstraction [5]. We also manage side effects.

We also notice that our framework is a strict improvement of the model reduction that was proposed in [28], in the sense that each fragment in the framework of [28] is necessarily a prefragment in the present analysis, but not the converse. That is to say, that we get smaller fragments with the analysis that we have described in this chapter. In [28] is studied a set of 71 rules expanding into 18 051 984 143 555 729 567 species, reduced into 175 988 fragments.

In the following chapters, we show that we can tune arbitrarily the accuracy of our analysis, by distinguishing the flow of information according to some contextual information.

Chapter 7

Context-sensitive abstractions

In the previous chapter, we have used contact maps to abstract the flow of information between sites of species. This abstraction is context-insensitive: we can either say that the state of a site may influence another site, or that the state of a site will never influence another site. In this chapter, we wonder whether it is possible or not to enhance the context-sensitivity of this approach.

For this purpose we introduce the notion of Σ-graph [21]. Σ-graph can be used to denote mixtures, patterns, contact maps. They can be unfolded/folded to describe/abstract away some particular contexts. This is a convenient tool to describe types and properties about species.

7.1 Category of Σ-graphs

7.1.1 Σ-graphs

Let us give the definition of a Σ-graph.

Firstly, we introduce the set `Ext` as:

$$\texttt{Ext} := \{\epsilon, -\} \cup \{(A, s) | A \in \mathcal{A}, s \in \Sigma_\lambda(A)\}$$

to describe some linking states.

Starting from a signature Σ we can define the concept of Σ-graph.

Definition 7.1.1 (Σ-graph)**.** *A Σ-graph is a tuple*

$$G = (\mathcal{A}_G, type_G, \mathcal{S}_G, \mathcal{L}_G, pk_G)$$

where:

1. $\mathcal{A}_G$ *is a set of agents;*

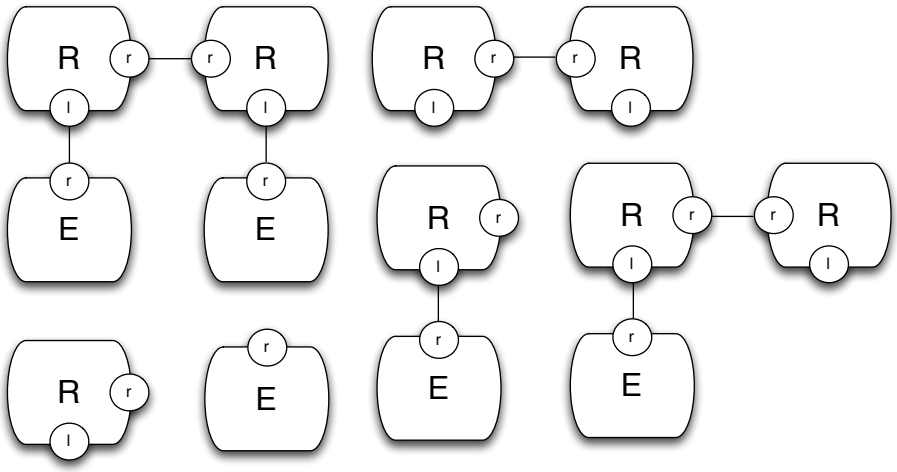


Figure 7.1: A Σ-graph.

2. *$type_G : \mathcal{A}_G \to \mathcal{A}$ is an agent type assignement function;*

3. *$\mathcal{S}_G$ is a set of sites, such that $\mathcal{S}_G \subseteq \{(i,x) \mid i \in \mathcal{A}_G$ and $x \in \Sigma(type_G(i))\}$;*

4. *$\mathcal{L}_G$ is a symmetric link relation, such that $\mathcal{L}_G \subseteq (\{(i,x) \in \mathcal{S}_G \mid x \in \Sigma_\lambda(type_G(i))\} \cup \mathtt{Ext})^2 \backslash \mathtt{Ext}^2$;*

5. *and pk_G maps each site $(i,x) \in \mathcal{S}_G$ such that $x \in \Sigma_\iota$ to a set of internal states $pk_G(i,x) \in \wp(\mathbb{I})$.*

Given a Σ-graph G, we write $\mathcal{A}_G$ for the set of its agents, $type_G$ for its typing function, $\mathcal{S}_G$ for the set of its sites, $\mathcal{L}_G$ for the set of its links and pk_G for its internal states map. An example of Σ-graph is given in Figure 7.1.

7.1.2 Homomorphisms between Σ-graphs

Two Σ-graphs, G and G', can be put in relation by *structure-preserving* functions from the agents of G to the agents of G'. Those functions are called *homomorphisms*. By structure-preserving we mean that:

1. agents of a specific type are mapped to agents of the same type;

2. if an agent exhibits a site, the site is present also in the image of that agent;

3. the state of sites is preserved;

4. the image of a site corresponding to a link holds a higher (or equal) linking information.

The *link information order* is given by a subtyping relation $\leqslant_{\mathtt{type}}$, that is the least reflexive relation $\leqslant_{\mathtt{type}}$ such that for all $A \in \mathcal{A}$ and $x \in \Sigma$ and i such that $\mathtt{type}_G(i) = A$, we have:

$$- \ \leqslant_{\mathtt{type}} (A, i) \leqslant_{\mathtt{type}} (n, i).$$

Homomorphisms are formally defined as follows.

Definition 7.1.2. *A* homomorphism *of Σ-graphs $h : G \to H$ is a (total) function on agents $h : \mathcal{A}_G \to \mathcal{A}_H$ such that:*

1. *$type_G(i) = type_H(h(i))$ for all $i \in \mathcal{A}_G$;*
2. *if $(i, x) \in \mathcal{S}_G$ then $(h(i), x) \in \mathcal{S}_H$;*
3. *$pk_G(i, x) \subseteq pk_H(h(i), x)$ for every $(i, x) \in \mathcal{S}_G$ such that $x \in \Sigma_\iota(type_G(i))$;*
4. *$((h(i), x), (h(i'), x')) \in \mathcal{L}_H$ for all $((i, x), (i', x')) \in \mathcal{L}_G \cap \mathcal{S}_G^2$;*
5. *there exists $s' \in \mathcal{S}_H \cup \mathtt{Ext}$ such that $((h(i), x), s') \in \mathcal{L}_H$ and $s \leqslant_H s'$ for all $((i, x), s) \in \mathcal{L}_G$ such that $s \in \mathtt{Ext}$.*

In Figure 7.2, it is possible to observe some examples of Σ-graphs homomorphisms. There we have:

- a homomorphism from $\mathcal{G}_1$ to $\mathcal{G}_2$: the agents $\mathtt{A}(\mathtt{x_u}, \mathtt{y_u})$ and $\mathtt{A}(\mathtt{x_u}, \mathtt{y_p})$ are mapped to the agent $\mathtt{A}(\mathtt{x_u}, \mathtt{y_{u|p}})$, whereas the agents $\mathtt{A}(\mathtt{x_p}, \mathtt{y_u})$ and $\mathtt{A}(\mathtt{x_p}, \mathtt{y_p})$ are mapped to the agent $\mathtt{A}(\mathtt{x_p}, \mathtt{y_{u|p}})$;

- a homomorphism from $\mathcal{G}_1$ to $\mathcal{G}_3$: the agents $\mathtt{A}(\mathtt{x_u}, \mathtt{y_u})$ and $\mathtt{A}(\mathtt{x_p}, \mathtt{y_u})$ are mapped to the agent $\mathtt{A}(\mathtt{x_{u|p}}, \mathtt{y_u})$, whereas the agents $\mathtt{A}(\mathtt{x_u}, \mathtt{y_p})$ and $\mathtt{A}(\mathtt{x_p}, \mathtt{y_p})$ are mapped to the agent $\mathtt{A}(\mathtt{x_{u|p}}, \mathtt{y_p})$;

- a homomorphism from $\mathcal{G}_1$ to $\mathcal{G}_4$: all the agents are mapped to the agent $\mathtt{A}(\mathtt{x_{u|p}}, \mathtt{y_{u|p}})$;

- a homomorphism from $\mathcal{G}_2$ to $\mathcal{G}_4$: all the agents are mapped to the agent $\mathtt{A}(\mathtt{x_{u|p}}, \mathtt{y_{u|p}})$;

- a homomorphism from $\mathcal{G}_3$ to $\mathcal{G}_4$: all the agents are mapped to the agent $\mathtt{A}(\mathtt{x_{u|p}}, \mathtt{y_{u|p}})$.

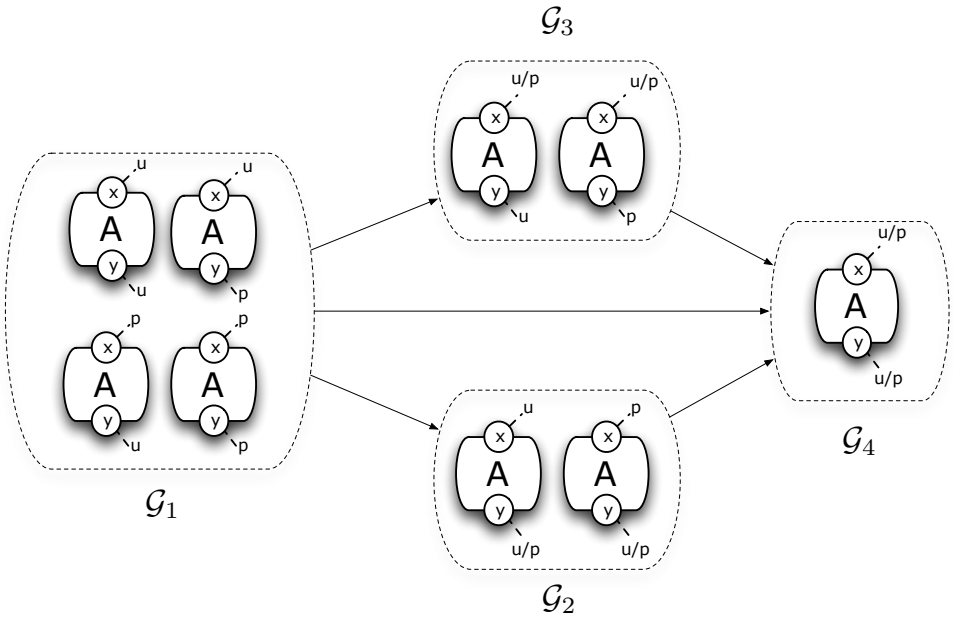


Figure 7.2: Σ-graphs morphisms.

We write u|p when the state of a site can be u or p but we ignore the exact value.

Given a homomorphism h we say that h is an *embedding* every time that h is injective. Generally, given two Σ-graphs G and H, it is possible to have more than one homomorphism between them. So the number of homomorphisms between G and H is denoted by $[G, H]$. We say that h is an *automorphism*, whenever $G = H$ and h is a bijection. As we can notice, the identity function is always an automorphism. Homomorphisms compose in the usual way. Moreover, whenever two homomorphisms $f : G \to H$ and $g : H \to G$ are such that their composition is the identity homomorphism over H, then f and g are called *isomorphisms* and G and H are said to be isomorphic. This is denoted by $G \approx H$.

In the following, all the constructions are defined up to isomorphisms. Isomorphisms (and automorphisms) are all embeddings.

7.2 Basic elements of Abstract Interpretation

Here we recall the main concepts of Abstract Interpretation. The interested reader can find more details in [20].

7.2.1 Ordered sets

Let us start by reminding some general definitions.

Definition 7.2.1 (Partially ordered set)**.** *A* partially ordered set *(or* poset*)* $\langle P, \sqsubseteq \rangle$ *is a non empty set* P *with a partial order* $\sqsubseteq$*, that is a reflexive, anti-symmetric, and transitive binary relation.*

If it exists, the *greatest lower bound* (or glb) of a set P is denoted by $\bigsqcap P$. In a similar way, if the *least upper bound* (or lub) of P exists, it is denoted by $\bigsqcup P$. If they exist, we denote by $\bot$ the least element and $\top$ the greatest element.

Definition 7.2.2 (Lattice)**.** *A* lattice $\langle P, \sqsubseteq, \sqcup, \sqcap \rangle$ *is a poset where each pair of elements* $a, b \in P$ *has a least upper bound, denoted by* $a \sqcup b$*, and a lower upper bound, denoted by* $a \sqcap b$*.*

Definition 7.2.3 (Complete lattice)**.** *A lattice* $\langle P, \sqsubseteq, \sqcup, \sqcap \rangle$ *is said to be* complete *if any set* $P' \subseteq P$ *has a greatest lower bound and a least upper bound.*

In this case, it is denoted by $\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$*.*

A classical example of complete lattice is the *power-set* of any set S, $\langle \mathcal{P}(S), \subseteq, \cap, \cup, \varnothing, S \rangle$.

Definition 7.2.4 (Sublattice)**.** *Let* $\langle P, \sqsubseteq, \sqcup, \sqcap \rangle$ *be a lattice. Let* S *be a subset of* P *and let* $\sqsubseteq_S$*,* $\sqcup_S$ *and* $\sqcap_S$ *be the restriction to* S *of* $\sqsubseteq$*,* $\sqcup$ *and* $\sqcap$*, respectively.*

Then $\langle S, \sqsubseteq_S, \sqcup_S, \sqcap_S \rangle$ *is a sublattice of* $\langle P, \sqsubseteq, \sqcup, \sqcap \rangle$ *if and only if for all* $x, y \in S$ *we have* $x \sqcup y \in S$ *and* $x \sqcap y \in S$*.*

7.2.2 Galois connections

Definition 7.2.5 (Galois connection)**.** *A* Galois connection *between two complete lattices* $\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$ *and* $\langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle$ *is a pair of maps:*

$$\alpha \in P \to P^\sharp, \gamma \in P^\sharp \to P$$

such that:

$$\forall p \in P : \forall p^\sharp \in P^\sharp : \alpha(p) \sqsubseteq^\sharp p^\sharp \Leftrightarrow p \sqsubseteq \gamma(p^\sharp).$$

In which case we write:

$$\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle.$$

Galois connections enjoy several properties (the interested reader can find a good summary with all references to mathematical literature in [19]). For example, $\gamma \circ \alpha$ is *extensive*:

$$\forall p \in P : p \sqsubseteq \gamma \circ \alpha(p) \tag{7.1}$$

since $\alpha(p) \sqsubseteq^\sharp \alpha(p)$ by reflexivity, hence $p \sqsubseteq \gamma \circ \alpha(p)$ by 7.2.5 with $p^\sharp = \alpha(p)$. If we abstract and than concretize we obtain something that is bigger than what we started from. This means that the loss of information in our process of abstraction is sound.

The same way $\alpha \circ \gamma$ is *reductive*:

$$\forall p^\sharp \in P^\sharp : \alpha \circ \gamma(p^\sharp) \sqsubseteq^\sharp p^\sharp \tag{7.2}$$

since $\gamma(p^\sharp) \sqsubseteq \gamma(p^\sharp)$ by reflexivity, hence $\alpha \circ \gamma(p^\sharp) \sqsubseteq^\sharp p^\sharp$ by 7.2.5 with $p = \gamma(p^\sharp)$. If we have an abstract element, we concretize and after we abstract it again, we obtain something that is equal, or smaller, than our starting point. That means that our abstraction is as precise as possible and we do not loose any information by the concretization process.

It follows that:

1. α is *monotone*: since $p_1 \sqsubseteq p_2$ implies $p_1 \sqsubseteq \gamma \circ \alpha(p_2)$ by 7.1 and transitivity whence $\alpha(p_1) \sqsubseteq^\sharp \alpha(p_2)$ by 7.2.5;

2. γ is *monotone*: since $p_1^\sharp \sqsubseteq^\sharp p_2^\sharp$ implies $\alpha \circ \gamma(p_1^\sharp) \sqsubseteq^\sharp p_2^\sharp$ by 7.2 and transitivity whence $\gamma(p_1^\sharp) \sqsubseteq \gamma(p_2^\sharp)$ by 7.2.5.

$$\alpha \in \langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle \xrightarrow{m} \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle \tag{7.3}$$

$$\gamma \in \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle \xrightarrow{m} \langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle \tag{7.4}$$

The meaning of monotonicity is that the soundness of the approximation is preserved by α and γ.

Equations 7.1, 7.2, 7.3 and 7.4 imply 7.2.5, so an alternative definition of Galois connection can be given as:

$$\begin{aligned} &\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle \\ &\iff \begin{cases} [\alpha \in \langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle \xrightarrow{m} \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle] \textit{ and} \\ [\gamma \in \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle) \xrightarrow{m} \langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle \textit{ and} \\ [\forall p \in P : p \sqsubseteq \gamma \circ \alpha(p)] \textit{ and} \\ [\forall p^\sharp \in P^\sharp : \alpha \circ \gamma(p^\sharp) \sqsubseteq^\sharp p^\sharp] \end{cases} \end{aligned} \tag{7.5}$$

We can observe that:

$$P(\sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} P^\sharp(\sqsubseteq^\sharp) \text{ if and only if } P^\sharp(\sqsubseteq^{\sharp^{-1}}) \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} P(\sqsubseteq^{-1})$$

where the inverse $\sqsubseteq^{-1}$ of the partial order $\sqsubseteq$ is $\sqsupseteq$. Thus, any result can be interpreted dually by reversing the order.

For all $p \in P$ and $p^\sharp \in P^\sharp$, we have $\alpha \circ \gamma(p^\sharp) \sqsubseteq^\sharp p^\sharp$ by 7.2, whence by monotony $\gamma \circ \alpha \circ \gamma(p^\sharp) \sqsubseteq^\sharp \gamma(p^\sharp)$. Moreover, $\gamma(p^\sharp) \sqsubseteq^\sharp \gamma \circ \alpha \circ \gamma(p^\sharp)$, by 7.1 when p is $\gamma(p^\sharp)$. By antisymmetry, we conclude that:

$$\forall p^\sharp \in P^\sharp : \gamma \circ \alpha \circ \gamma(p^\sharp) = \gamma(p^\sharp). \tag{7.6}$$

The same way, we have $\alpha \circ \gamma \circ \alpha(p) \sqsubseteq^\sharp \alpha(p)$ for all $p \in P$. Moreover 7.1 implies that for all $p \in P$ we have $p \sqsubseteq \gamma \circ \alpha(p)$, and, by monotony we have $\alpha(p) \sqsubseteq^\sharp \alpha \circ \gamma \circ \alpha(p)$. By antisymmetry, we conclude that:

$$\forall p \in P : \alpha \circ \gamma \circ \alpha(p) = \alpha(p). \tag{7.7}$$

By recalling the definitions of closure operators (upper and lower), we can observe that a Galois connection defines a couple of closure operators. $\alpha \circ \gamma$ is the lower closure operator and $\gamma \circ \alpha$ is the upper one.

Definition 7.2.6 (Upper closure operator)**.** *Let $\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$ be a complete lattice. An operator ρ of P is said to be an upper closure operator if and only if:*

1. *ρ is monotone, i.e. $\forall x, y \in P,\ x \sqsubseteq y \Rightarrow \rho(x) \sqsubseteq \rho(y)$;*
2. *ρ is extensive, i.e. $\forall x \in P,\ x \sqsubseteq \rho(x)$;*
3. *ρ is idempotent, i.e. $\forall x \in P,\ \rho(x) = \rho(\rho(x))$.*

Definition 7.2.7 (Lower closure operator)**.** *Let $\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$ be a complete lattice. An operator ρ of P is said to be an lower closure operator if and only if:*

1. *ρ is monotone;*
2. *ρ is reductive, i.e. $\forall x \in P,\ \rho(x) \sqsubseteq x$;*
3. *ρ is idempotent.*

So, an immediate consequence is

$$P(\sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} P^\sharp(\sqsubseteq^\sharp) \implies \gamma \circ \alpha \text{ is a upper closure operator} \tag{7.8}$$

and

$$P(\sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} P^\sharp(\sqsubseteq^\sharp) \implies \alpha \circ \gamma \text{ is an lower closure operator.} \tag{7.9}$$

Idempotence can be interpreted as the fact that all the information is lost once for all during the abstraction process, so the result of two abstractions performed one after the other will be the same as if apply the abstraction just once.

Another consequence is that we do not need to consider an abstract and a concrete domain, but we can use the concrete domain as the abstract one after applying to it the upper closure operator $\gamma \circ \alpha$.

Definition 7.2.8 (Moore family)**.** *Let $(P, \sqsubseteq)$ be a poset with a top element $\top$. A Moore family is a subset $M \subseteq P$ such that:*

- $\top \in M$
- *if $X \in \mathcal{P}(M) \backslash \{\varnothing\}$ then $\sqcap X$ exists in P and $\sqcap X \in M$*

or equivalently

- *if $X \in \mathcal{P}(M)$ then $\sqcap X$ exists in P and $\sqcap X \in M$*

that is to say M is closed under meet.

In particular, the use of *Moore families* is justified by the following:

Proposition 7.2.9. *Let $\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$ and $\langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle$ be two complete lattices. If $\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle$ then $\gamma^*(P^\sharp)$ is a Moore family (where $\gamma^\star(P^\sharp) = \{\gamma(p) \mid p \in P\}$).*

Proof. See [20]. □

The reader interested in this equivalent approach can have a look to [19].

In a Galois connection one function uniquely and absolutely determines the other:

Proposition 7.2.10. *If $\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle$ and $\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp \rangle$ then $(\alpha_1 = \alpha_2)$ if and only if $(\gamma_1 = \gamma_2)$.*

Proof. See [20]. □

So providing that we are in a complete lattice, we can perform an Abstract Interpretation by defining the abstraction or, indifferently, the concretization function. The following method gives us the way to uniquely determine the adjoined function.

Proposition 7.2.11. *If* $\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp\rangle$ *then, for all* $p \in P$, $\alpha(p)$ *is equal to the greatest lower bound* $\bigsqcup^\sharp\{p^\sharp | p \sqsubseteq \gamma(p^\sharp)\}$ *of the inverse image by* γ *of the set of upper bounds of* p.

For all $p^\sharp \in P^\sharp$ *we have* $\gamma(p^\sharp) = \bigsqcup\{p | \alpha(p) \sqsubseteq^\sharp p^\sharp\}$.

Proof. See [20]. □

As a consequence, Galois connections between complete lattices preserve bounds.

Proposition 7.2.12. *If* $\langle P, \sqsubseteq, \sqcup, \sqcap, \bot, \top\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle P^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp\rangle$, *then* $\alpha \in P(\vee) \xrightarrow{a} P^\sharp(\vee^\sharp)$ *preserves the least upper bounds and* $\gamma \in P^\sharp(\wedge^\sharp) \xrightarrow{a} P(\wedge)$ *preserves the greatest upper bounds.*

Proof. See [20]. □

7.3 Abstraction of relations among sites

Now, given a Σ-graph G, we want to consider a binary relation over its sites. This relation can be used, for instance, to abstract the flow of information among sites, as we will see in the following chapter.

7.3.1 Relations among sites

A way to represent a relation between sites of a Σ-graph is given by the so-called *annotated* Σ*-graph*.

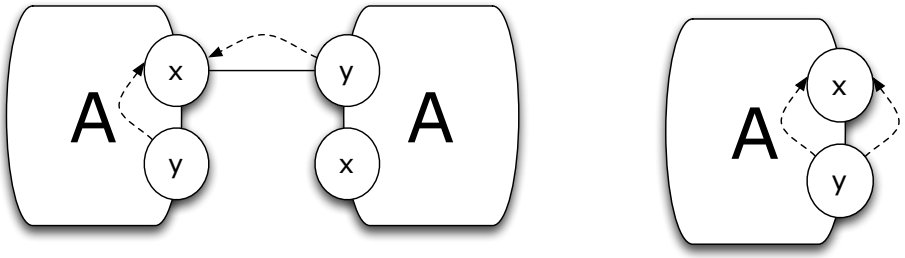
Definition 7.3.1 (Annotated Σ-graphs)**.** *An annotated* Σ*-graph* G_a *is a pair* $(G, \rightsquigarrow_{G_a})$ *where* G *is a* Σ*-graph and* $\rightsquigarrow_{G_a}$ *is a subset of* $\{(i,x),(i',x') \mid i \in \mathcal{A}_G, (i,x),(i',x') \in \mathcal{S}_G\} \uplus (\mathcal{L}_G \cap \mathcal{S}_G^2)$.

Ordered pairs of sites in $\{(n,i),(n,i') \mid n \in \mathcal{A}_G, (n,i),(n,i') \in \mathcal{S}_G\}$ are called *internal edges* and are denoted as $(n,i) \overset{\vee}{\rightsquigarrow}_{G^a} (n,i')$, whereas ordered pairs in $\mathcal{L}_G \cap {\mathcal{S}_G}^2$ are called *external edges* and are denoted as $(n,i) \overset{\wedge}{\rightsquigarrow}_{G^a} (n',i')$. An ordered pair of sites can be connected by both an internal edge and an external edge. We omit the symbols $\vee$ and $\wedge$ when they are not important.

Example 7.3.2. *If we consider the* Σ*-graph* G *in Figure 7.1 and the following relation:*

$$\rightsquigarrow_{G^a} := \{(n,i),(n,i') \mid n \in \mathcal{A}_G, (n,i),(n,i') \in \mathcal{S}_G\} \uplus (\mathcal{L}_G \cap \mathcal{S}_G^2),$$

the corresponding annotated Σ*-graph* $G_a = (G, \rightsquigarrow_{G^a})$ *is given in Figure 7.4.*



(a) (b)

Figure 7.3: Example of annotated Σ-graphs.

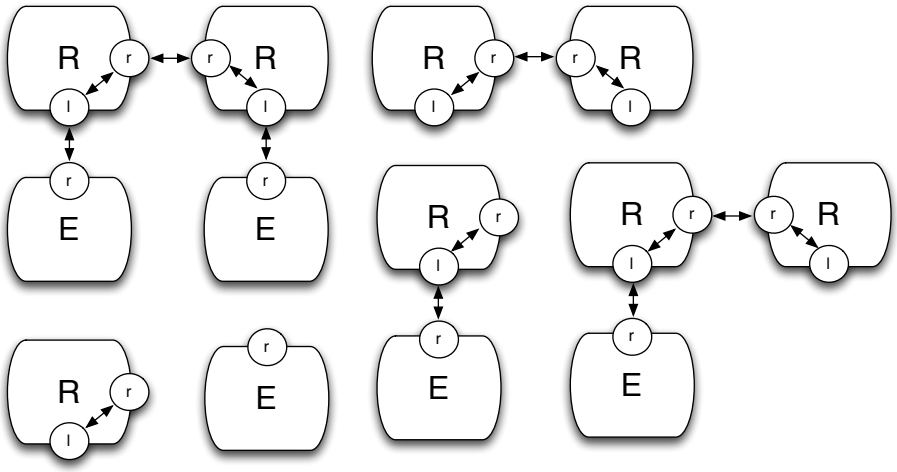


Figure 7.4: Annotated relation on sites of the Σ-graph in Figure 7.1.

For every Σ-graph G, we denote as $\mathcal{R}_G$ the relation over the sites of G, that is defined as follows:

$$\mathcal{R}_G := \{(n,i),(n,i') \mid n \in \mathcal{A}_G, (n,i),(n,i') \in \mathcal{S}_G\} \uplus (\mathcal{L}_G \cap \mathcal{S}_G^2).$$

$\mathcal{R}_G$ is the most complete relation over the sites of G. Moreover, the set $\wp(\mathcal{R}_G)$ is the set of all the potential relations over the sites of G. It forms a complete lattice, for inclusion. The bottom element of $\wp(\mathcal{R}_G)$ is the empty annotation, whereas the top element is the relation $\mathcal{R}_G$.

Let also consider another Σ-graph $\bar{G}$. The set $\wp(\mathcal{R}_{\bar{G}})$ is a complete lattice as well.

Now we want to build a correspondence between the relation $\mathcal{R}_G$ on the sites of G and the relation $\mathcal{R}_{\bar{G}}$ on the sites of $\bar{G}$. In particular we want to

have a method to switch the level of resolution from the one of G to the one of $\bar{G}$.

7.3.2 Approximation of relations among sites

Let G, $\bar{G}$ be two Σ-graphs. We define a function $\alpha_{G,G'} : \mathcal{P}(\mathcal{R}_G) \to \mathcal{P}(\mathcal{R}_{\bar{G}})$ that takes an annotation on the graph G and returns an annotation on the graph $\bar{G}$.

Definition 7.3.3 (Abstraction)**.** *Let G, $\bar{G}$ be two Σ-graphs and let $\mathbf{r}$ be a relation in $\mathcal{P}(\mathcal{R}_G)$.*

$$\alpha_{G,\bar{G}} = \begin{cases} \mathcal{R}_G \to \mathcal{R}_{\bar{G}} \\ \mathbf{r} \mapsto \{((\phi(i), x), (\phi(i'), x')) \,|\, \phi \in \mathcal{M}(G, \bar{G}),\ ((i, x), (i', x')) \in \mathbf{r}\}. \end{cases}$$

The function $\alpha_{G,\bar{G}}$ maps each relation in $\mathcal{P}(\mathcal{R}_G)$ of G to its best (smallest) abstract approximation in $\mathcal{P}(\mathcal{R}_{\bar{G}})$. Intuitively, every edge in an annotation of G is interpreted as a positive information: we want to keep this edge. $\bar{G}$ is another Σ-graph, hence it allows for distinguishing among the states of sites according to a different set of contexts. The abstraction reports every edge of the annotation G, in the annotation of $\bar{G}$, in every compatible context. Compatibility between contexts is formalised by the means of a homomorphism.

Definition 7.3.4 (Concretization)**.** *Let G and $\bar{G}$ be two Σ-graph and let $\bar{\mathbf{r}}$ be a relation in $\mathcal{P}(\mathcal{R}_{\bar{G}})$.*

$$\gamma_{G,\bar{G}} = \begin{cases} \mathcal{R}_{\bar{G}} \to \mathcal{R}_G \\ \bar{\mathbf{r}} \mapsto \{((i, x), (i', x')) \mid \forall \phi \in \mathcal{M}(G, \bar{G}), (\phi(i), x)\bar{\mathbf{r}}(\phi(i'), x')\} \end{cases}$$

The function $\gamma_{G,\bar{G}}$ maps each relation in $\mathcal{P}(\mathcal{R}_{\bar{G}})$ to the biggest concrete relation in $\mathcal{P}(\mathcal{R}_G)$ corresponding to it. Intuitively, the annotation of $\bar{G}$ has to be understood negatively. In there is no edge between two sites of $\bar{G}$, then we know that in this specific context there is no relation. Thus, the concretization collects everything that is known about the absence of relation.

Theorem 7.3.5. *Let G,$\bar{G}$ be two Σ-graphs and let $\mathcal{R}_G$ be the relation defined in 7.3.1. So*

$$\langle \mathcal{P}(\mathcal{R}_G), \subseteq, \varnothing, \mathcal{R}_G, \cup, \cap \rangle \underset{\alpha_{G,\bar{G}}}{\overset{\gamma_{G,\bar{G}}}{\leftrightarrows}} \langle \mathcal{P}(\mathcal{R}_{\bar{G}}), \subseteq, \varnothing, \mathcal{R}_{\bar{G}}, \cup, \cap \rangle .$$

Proof. As we have seen in Definition 7.2.5, the pair $(\alpha_{G,\bar{G}}, \gamma_{G,\bar{G}})$ forms a Galois connection if, and only if, for all $\mathbf{r} \in \mathcal{R}_G$ and $\bar{\mathbf{r}} \in \mathcal{R}_{\bar{G}}$ we have:

$$\alpha_{G,\bar{G}}(\mathbf{r}) \subseteq \bar{\mathbf{r}} \Leftrightarrow \mathbf{r} \subseteq \gamma_{G,\bar{G}}(\bar{\mathbf{r}}).$$

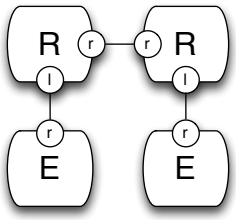


Figure 7.5: A dimer.

1. Let us prove that if $\alpha_{G,\bar{G}}(\mathbf{r}) \subseteq \bar{\mathbf{r}}$ then $\mathbf{r} \subseteq \gamma_{G,\bar{G}}(\bar{\mathbf{r}})$.

   We consider $((i,x),(i',x')) \in \mathbf{r}$ and a morphism ϕ in $\mathcal{M}(G,\bar{G})$.

   We know that $\phi(i,x)[\alpha_{G,\bar{G}}(\mathbf{r})]\phi(i',x')$.

   So, since $\alpha_{G,\bar{G}}(\mathbf{r}) \subseteq \bar{\mathbf{r}}$, we have that $\phi(i,x)\bar{\mathbf{r}}\phi(i',x')$.

   So, by definition of $\gamma_{G,\bar{G}}$, $((i,x),(i',x')) \in \gamma_{G,\bar{G}}(\mathbf{r})$. So $\mathbf{r} \subseteq \gamma_{G,\bar{G}}(\bar{\mathbf{r}})$.

2. Let us prove that if $\mathbf{r} \subseteq \gamma_{G,\bar{G}}(\bar{\mathbf{r}})$ then $\alpha_{G,\bar{G}}(\mathbf{r}) \subseteq \bar{\mathbf{r}}$.

   Let us consider a morphism ϕ in $\mathcal{M}(G,\bar{G})$ and a couple of sites $((i,x),(i',x'))$ such that $\phi(i) = \bar{i}$ and $\phi(i') = \bar{i}'$ and $(i,x)\mathbf{r}(i',x')$.

   Since $\mathbf{r} \subseteq \gamma_{G,\bar{G}}(\bar{\mathbf{r}})$, we have that $(i,x)\gamma_{G,\bar{G}}(\bar{\mathbf{r}})(i',x')$.

   By definition of $\gamma_{G,\bar{G}}$, we have: $(\phi(i),x)\bar{\mathbf{r}}(\phi(i'),x')$.

   So we get that: $((\bar{i},\bar{x}),(\bar{i}',\bar{x}')) \in \mathbf{r}$.

□

Example 7.3.6. *To see how both functions work, let us consider the example given by the dimer in Figure 7.5. We will not consider all the possible relationships between sites, but we will restrict our example to the ones showed in the sublattice $\mathcal{C}$ of Figure 7.6.*

The sublattice $\mathcal{C}$ will be used both as the concrete and as the abstract domain, so $\alpha : \mathcal{C} \to \mathcal{C}$ and $\gamma : \mathcal{C} \to \mathcal{C}$. The abstraction is showned in Figure 7.7 and the concretization in Figure 7.8.

Since the two functions, $\alpha_{G,\bar{G}}$ and $\gamma_{G,\bar{G}}$, form a Galois connection they enjoy several properties. In particular, their compositions are both closure operators: $\gamma_{G,\bar{G}} \circ \alpha_{G,\bar{G}}$ is the upper one (Definition 7.2.6), whereas $\alpha_{G,\bar{G}} \circ \gamma_{G,\bar{G}}$ is the lower (Definition 7.2.7).

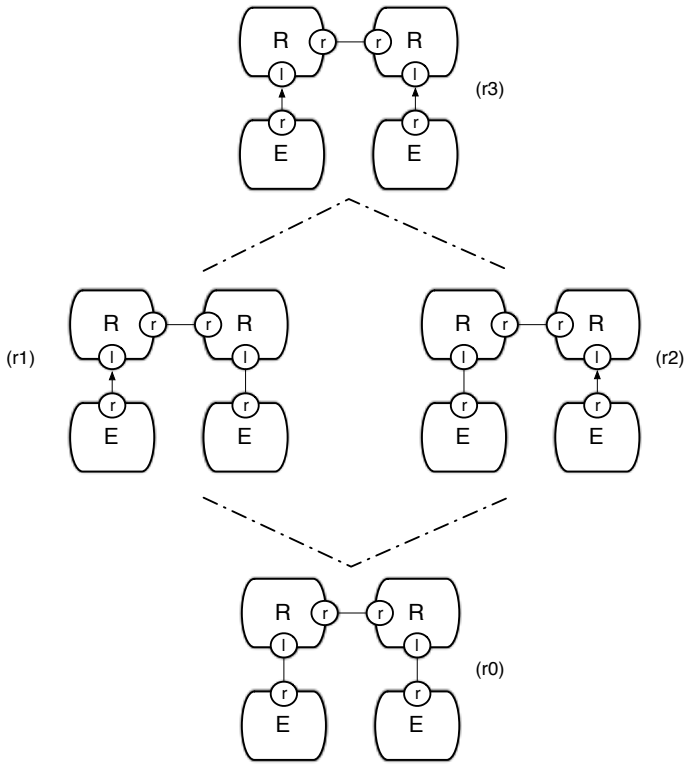


Figure 7.6: Lattice $\mathcal{C}$.

Example 7.3.7 (continued)**.** *Figure 7.9 shows the result of their application to the lattice in Figure 7.6. The red arrows correspond to* $\alpha_{G,\bar{G}} \circ \gamma_{G,\bar{G}}$*; the blue ones correspond to* $\gamma_{G,\bar{G}} \circ \alpha_{G,\bar{G}}$*.*

The upper closure $\gamma_{G,\bar{G}} \circ \alpha_{G,\bar{G}}$ characterises the granularity of our abstraction. It consists in reporting every edge in the annotation of G in every other context that cannot be discriminated from the initial one in $\bar{G}$. Two annotations of G having the same image by $\gamma_{G,\bar{G}} \circ \alpha_{G,\bar{G}}$ describe the same information. In the case when G and $\bar{G}$ are equal, this amounts to propagate edges along every automorphism.

The lower closure $\alpha_{G,\bar{G}} \circ \gamma_{G,\bar{G}}$ allows to simplify the annotation of $\bar{G}$. It propagates the information we have about the absence of edges that can be described at the level of resolution of G, in $\bar{G}$. In the case when G and $\bar{G}$ are equal, this amounts to remove every edge between two sites, if such an edge is missing between the image of these sites by an automorphism.

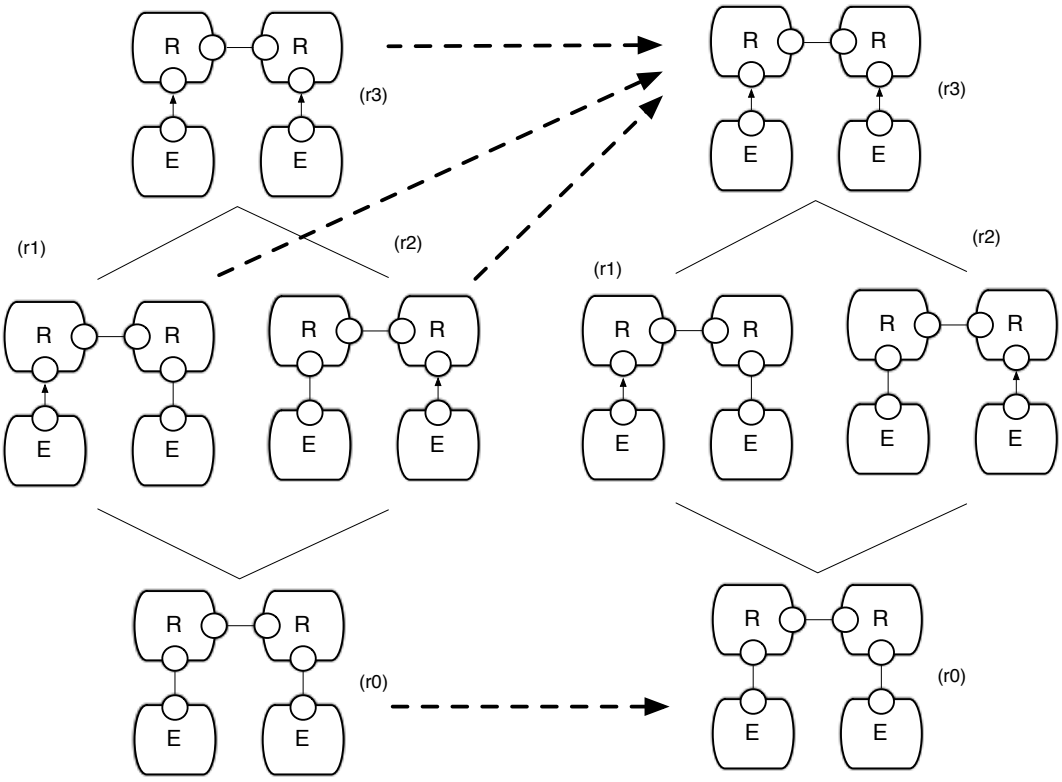


Figure 7.7: Abstraction. The dotted arrows represent the function α.

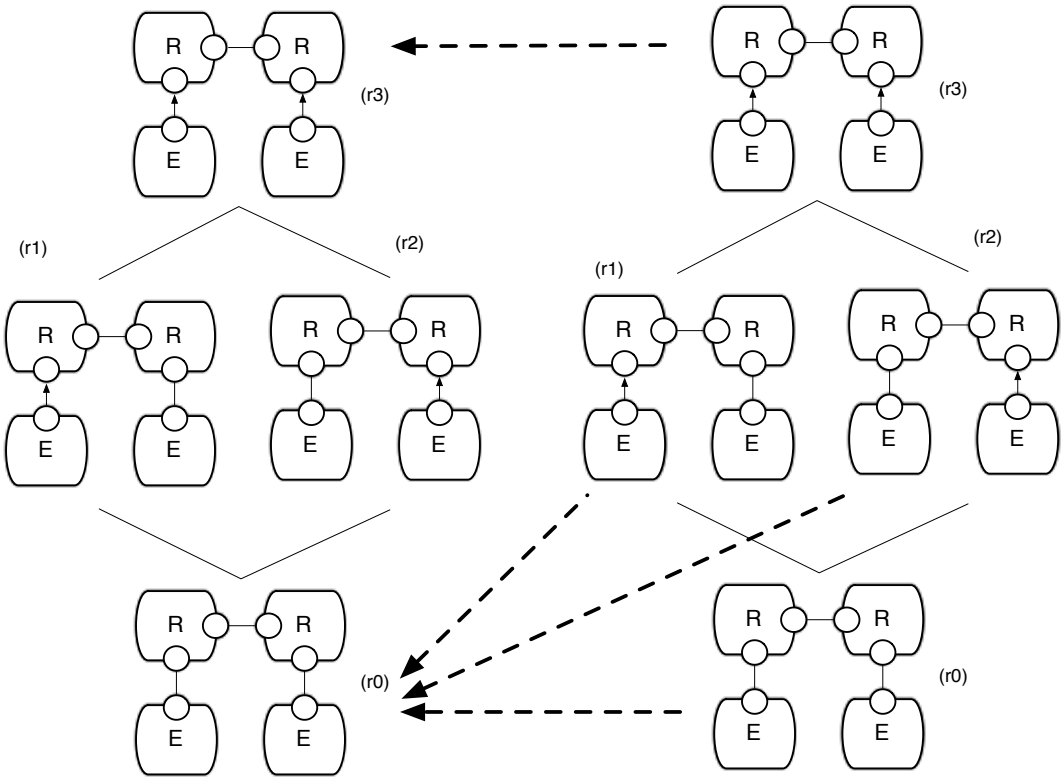


Figure 7.8: Concretization. The dotted arrows represent the function γ.

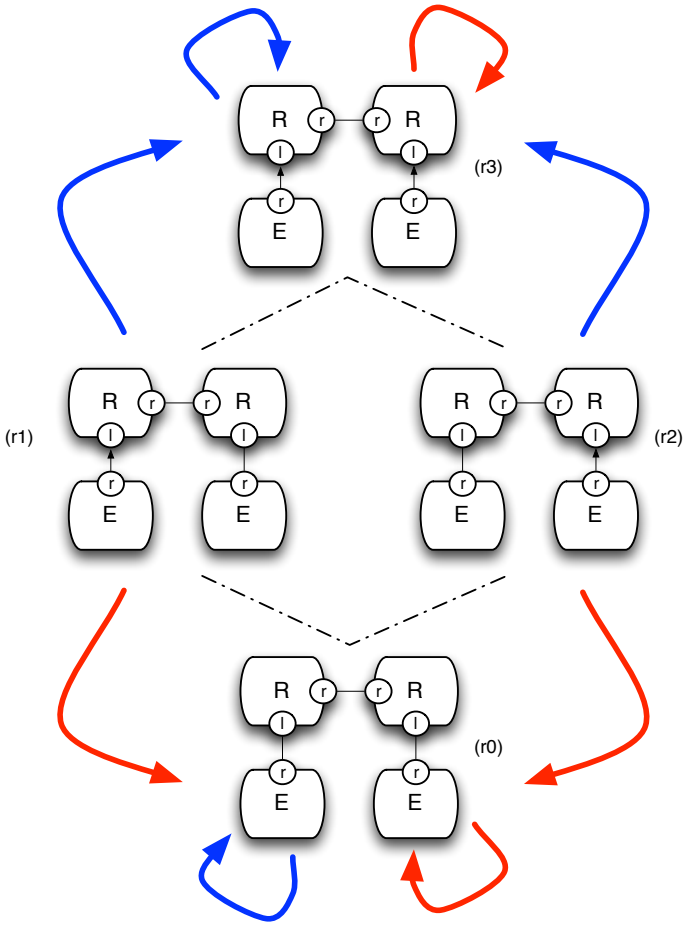


Figure 7.9: Closure operators.

Chapter 8

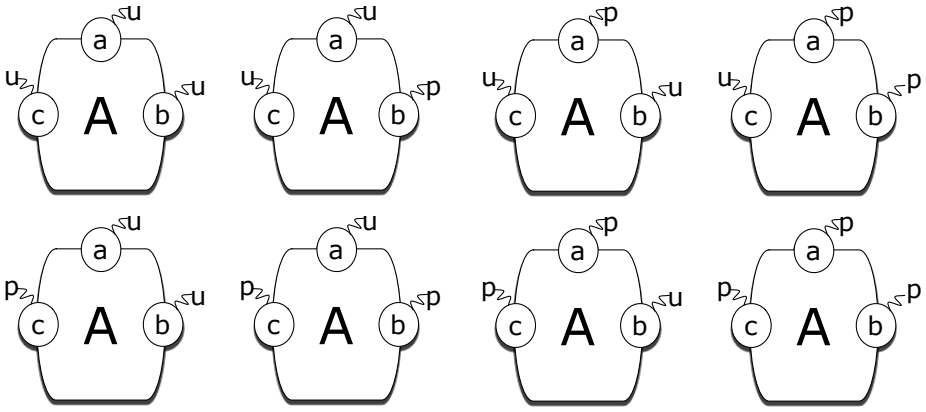
Partitioned flow

In this chapter, we show how to use an arbitrary Σ-graph to summarise the potential flow of information between the sites of species in a context-sensitive way. Then we use this summary to derive a set of self-consistent fragments. This way, we can tune the resolution of our model reduction according to the contexts which can be distinguished in the Σ-graph (which is left as a parameter of our abstraction).

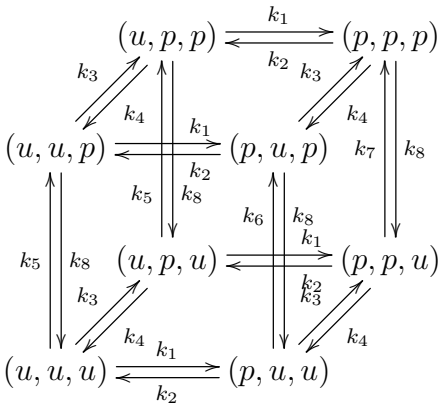
8.1 Case study

Before describing the framework formally, we introduce a case study. We consider one kind of protein P with three phosphorylation sites, called a, b, and c. Each site can be phosphorylated, or not, thus each instance of P can take 8 (as 2^3) configurations. We consider that any site can get phosphorylated or lose its phosphorylation, thus there are 24 chemical reactions (3 per configurations). Configurations are listed in Figure 8.1(a), reactions are drawed in Figure 8.1(b). The protein in the state $P(a_{w_a}, b_{w_b}, c_{w_c})$ is denoted as a triple (w_a, w_b, w_c) of symbols among u and p. In general, the rate constant of phosphorylation (resp. dephosphorylation) can depend on the state of the other sites. Here we make the assumption that only the phosphorylation rate constant of the third site depends on the state of the two other sites, but we assume that the phosphorylation rate constant of the third site is the same in the configurations (u,u,u) and (u,p,u). Thus, our system is parameterised by 12 kinetic rates (e.g.. see Fig.8.1(b)).

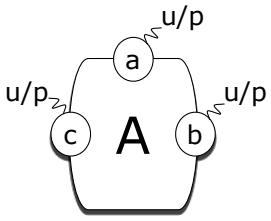
The behaviour of the system is formalised by the means of the following system of differential equations, which describes the derivatives of the concentrations of each configuration of P:



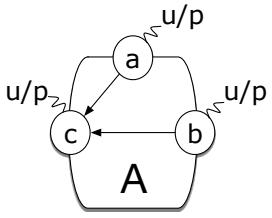
(a) Catalogue of species.



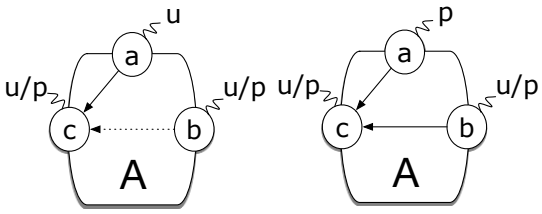
(b) Reactions.



(c) Contact map.



(d) Context-insensitive abstraction of the flow of information.



(e) Context-sensitive abstraction of the flow of information. The dotted arrow represents a dependency that can be dropped if the rate constant k_1 is zero.

Figure 8.1: Case study

$$
\begin{array}{rcl}
[(u,u,u)]' &=& k_2[(p,u,u)] + k_4[(u,p,u)] + k_8[(u,u,p)] - (k_1+k_3+k_5)[(u,u,u)] \\
[(u,u,p)]' &=& k_2[(p,u,p)] + k_4[(u,p,p)] + k_5[(u,u,u)] - (k_1+k_3+k_8)[(u,u,p)] \\
[(u,p,p)]' &=& k_2[(p,p,p)] + k_3[(u,u,p)] + k_5[(u,p,u)] - (k_1+k_4+k_8)[(u,p,p)] \\
[(u,p,u)]' &=& k_2[(p,p,u)] + k_3[(u,u,u)] + k_8[(u,p,p)] - (k_1+k_4+k_5)[(u,p,u)] \\
[(p,p,u)]' &=& k_1[(u,p,u)] + k_3[(p,u,u)] + k_8[(p,p,p)] - (k_2+k_4+k_7)[(p,p,u)] \\
[(p,p,p)]' &=& k_1[(u,p,p)] + k_3[(p,u,p)] + k_7[(p,p,u)] - (k_2+k_4+k_8)[(p,p,p)] \\
[(p,u,p)]' &=& k_1[(u,u,p)] + k_4[(p,p,p)] + k_6[(p,u,u)] - (k_2+k_3+k_8)[(p,u,p)] \\
[(p,u,u)]' &=& k_1[(u,u,u)] + k_4[(p,p,u)] + k_8[(p,u,p)] - (k_2+k_3+k_6)[(p,u,u)].
\end{array}
$$

Now, we wonder whether or not our model can be coarse-grained: we are looking for a set macro-variables which are defined as a linear combination of the variables of the initial systems (so called micro-variables) that are self-consistent. That is to say that the derivatives of the macro-variables must be expressed as functions of only the macro-variables. In Chapter 6, we have introduced a framework for detecting self-consistent coarse-graining thanks to an over-approximation of the flow of information between the states of the sites of proteins. Indeed the flow of information can be summarised by annotating a contact map (which describes the different kinds of proteins, their sites, their potential phosphorylation states and their potential binding) with an oriented relation over the sites, which summarises how each site may influence the other ones: an arrow from a site s_1 to a site s_2 means that the capability of modifying the state of the site s_2 may change according to the state of the site s_1.

The annotated contact map for our case study is given in Figure 8.1(d). This is a context-insensitive approximation since all the information about the sites of P is summarised in a single node, regardless the states of its sites, which prevents from providing distinct annotations for a given site depending on the state of the other sites. The arrow from the site a (resp. b) site to the site c comes from the fact that the phosphorilation rate constant of the site c may depend on the state of the site a (resp. b) (since k_6 may not be equal to the rate constant k_5 (resp. k_7)). No other arrows are required, since the phosphorylation/dephosphorylation rates of both the sites a and b do not depend on the states of other sites (indeed, we can check on Figure 8.1(b) that the rates of the corresponding reactions are the same four by four). As a result, since the behaviour of the site c site depends on the state of all the other sites, no coarse-graining can be found in this way.

Indeed, without further assumptions, the model cannot be coarse-grained by any means. But interestingly, if we set the rate constant k_1 equal to 0, we can abstract away the relation between the state of the sites b and c in the case when the site a is activated. This is formalised by the following differential equations:

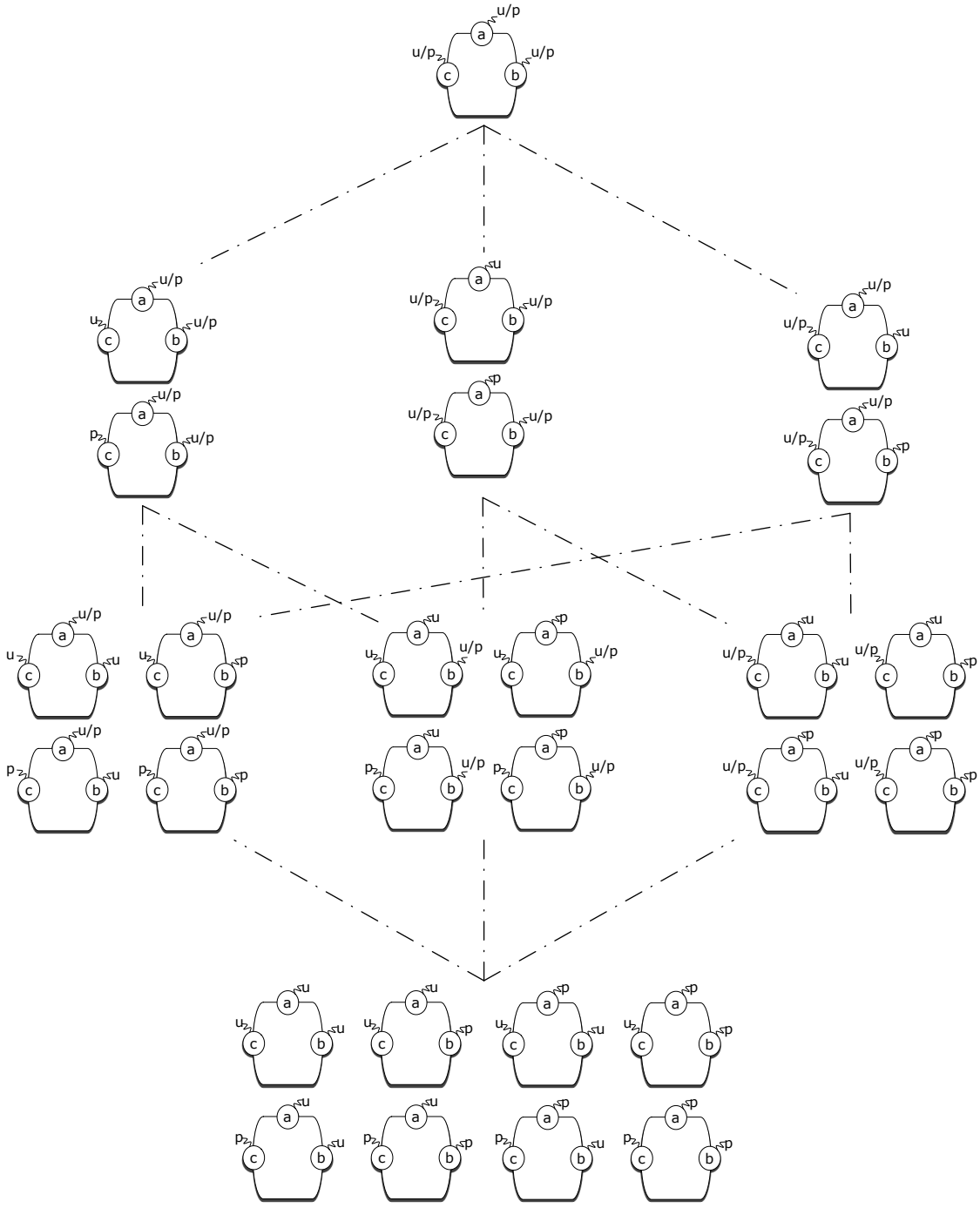


Figure 8.2: Hierarchy of Σ-graphs.

$$
\begin{array}{rcl}
[(?,u,?)]' & = & k_4[(?,p,?)] - k_3[(?,u,?)] \\
[(?,p,?)]' & = & k_3[(?,u,?)] - k_4[(?,p,?)] \\
[(u,?,p)]' & = & k_2([(p,u,p)] + [(p,p,p)]) + k_5[(u,?,u)] - k_8[(u,?,p)] \\
[(u,?,u)]' & = & k_2([(p,u,u)] + [(p,p,u)]) + k_8[(u,?,p)] - k_5[(u,?,u)] \\
[(p,p,u)]' & = & k_3[(p,u,u)] + k_8[(p,p,p)] - (k_2 + k_4 + k_7)[(p,p,u)] \\
[(p,p,p)]' & = & k_3[(p,u,p)] + k_7[(p,p,u)] - (k_2 + k_4 + k_8)[(p,p,p)] \\
[(p,u,p)]' & = & k_4[(p,p,p)] + k_6[(p,u,u)] - (k_2 + k_3 + k_8)[(p,u,p)] \\
[(p,u,u)]' & = & k_4[(p,p,u)] + k_8[(p,u,p)] - (k_2 + k_3 + k_6)[(p,u,u)],
\end{array}
$$

where the macro-variables are intentionally defined as fragments of configurations (question marks denote sites which have been cut away), and extensionally as linear combinations of the configurations which contain these fragments:

$$
\begin{array}{rcl}
[(?,u,?)] & = & [(u,u,u)] + [(u,u,p)] + [(p,u,u)] + [(p,u,p)] \\
[(?,p,?)] & = & [(u,p,u)] + [(u,p,p)] + [(p,p,u)] + [(p,p,p)] \\
[(u,?,u)] & = & [(u,u,u)] + [(u,p,u)] \\
[(u,?,p)] & = & [(u,u,p)] + [(u,p,p)].
\end{array}
$$

This coarse-graining can be discovered by tuning the context-sensitivity of the information flow analysis. Indeed, the behaviour of the protein P can be partitioned into two distinct modes. Whenever the site a is phosphorylated, the evolution of the state of the site c is controlled by the state of both sites a and b. But whenever the site a is not phosphorylated, the evolution of the state of the site c is not controlled by the state of the site b anymore (this can be checked in Figure 8.1(b) where the phosphorylation (resp. unphosphorylation) rate constant of the site c is the same whatever the protein is in the state (u,u,u) or (u,p,u) (resp. (u,u,p) or (u,p,p))).

This accurate approximation of the flow of information is out of the reach of context-insensitive analysis. Thus we propose to use arbitrary Σ-graphs where different annotations can be written according to the state of well chosen sites, unlike the contact map. An example Σ-graph is given in Figure 8.1(e). We notice that two nodes are used to describe the protein P, according to whether or not the site a is phosphorylated. Then, we can annotate our Σ-graph with context-sensitive information about the flow of information and obtain the plain arrows in Figure 8.1(e). Interestingly, in the left pattern component, there is no flow of information from any site into the site b and no flow of information from the site b into the site c. As a consequence, the fragments of proteins that contain the sites a and c, and the ones that only contain the site b are good candidates as macro-variables. Yet, since in the right pattern component there is a potential flow of information from the sites a and b into the site c, any micro-variable where the site a is phosphorylated

has to be preserved. Thus, we find again the set of macro-variables

$$\left\{ \begin{array}{l} [(?,u,?)],[(?,p,?)],[(u,?,u)],[(u,?,p)], \\ [(p,p,u)],[(p,p,p)],[(p,u,p)],[(p,u,u)] \end{array} \right\},$$

which is self-consistent, as we have shown previously.

Then we may wonder why this coarse-graining is not self-consistent when the phosphorylation reaction of the site a is not knocked out. This is because configurations of the form $(u,?,?)$ can now be transformed into configurations of the form $(p,?,?)$. Then, so as to express the concentration of the configurations of the form $(p,?,?)$ which are produced this way, we need to express the configurations of the form $(u,?,?)$ with at least the same fine-grained level of description. This is captured by the right-gluing construction in [36]. In the present framework, it is necessary to duplicate the arrow from the site b and the site c, from the right pattern component into the left one. The resulting arrow, drawn in dotted in Figure 8.1(e), prevents any coarse-graining.

Between the catalogue of species (Figure 8.1(a)) and the contact map (Figure 8.1(c)), we have a lot of intermediary representations. We show some of them in Figure 8.2.

In the following, we see how to abstract the flow of information at an arbitrary grain of description and we prove that it always provides a sound set of *fragments*.

8.2 Context-sensitive model reduction

As we have previously seen for the case of contact maps, an annotated Σ-graph can be interpreted as a symbolic representation of a set of patterns, called prefragments. Unlike in the context-insensitive case, we have no efficient ways to make the distinction between fragments and prefragments. Thus we characterize prefragments only, and we define a fragment as a prefragment the concentration of which cannot be expressed as a linear combination of the concentration of some other prefragments.

8.2.1 Prefragments

As we mentioned before, a special kind of Σ-graphs can be used to describe patterns. These are the so called *site graphs* that are formally defined as follows:

Definition 8.2.1 (Site graph)**.** *A site graph G is a Σ-graph such that:*

1. *the set* $\mathcal{A}_G$ *is finite;*

2. *the link relation* $\mathcal{L}_G$ *is irreflexive;*

3. *for any site* $s \in \mathcal{S}_G$, $(s, x) \in \mathcal{L}_G$ *and* $(s, y) \in \mathcal{L}_G$ *implies* $x = y$*;*

4. *for any site* $(i, x) \in \mathcal{S}_G$ *such that* $x \in \Sigma_\iota$, *the set* $pk_G(s)$ *contains at most one element.*

We can observe that site graphs have no link that immediately loops back to the same site and have at most one link from any site.

We can specialise the concept of site graphs further, and obtain a characterisation of mixtures.

Definition 8.2.2 (Mixture)**.** *A mixture is a site graph that additionally satisfies:*

1. $\mathcal{L} \subseteq (\mathcal{S}_G \cup \{\epsilon\})^2$*;*

2. $\mathcal{S}_G = \{(i, x) \mid i \in \mathcal{A}_G \text{ and } x \in \Sigma_{ag-st}(type(i))\}$*;*

3. *for any* $(i, x) \in \mathcal{S}_G$ *such that* $x \in \Sigma_\lambda(type_G(i))$, *there exists* $s' \in \texttt{Ext}$ *such that* $((i, x), s') \in \mathcal{L}_G$*;*

4. *for any* $(i, x) \in \mathcal{S}_G$ *such that* $x \in \Sigma_\iota$, *the set* $pk_G(i, x)$ *is a singleton.*

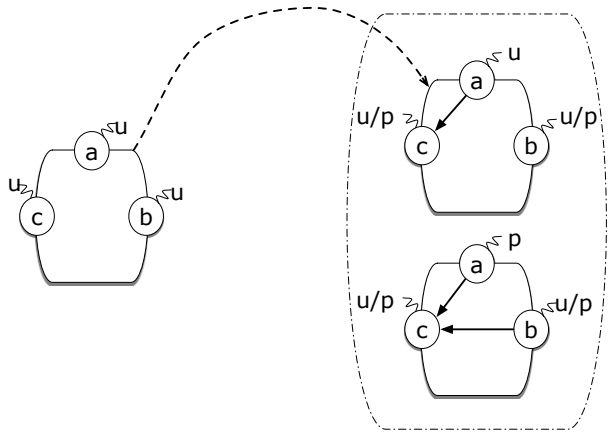
We can notice that, in addition to site graphs, mixtures specify the states of all their sites and have neither external links, nor binding types.

As an abuse of notation, we make no difference between a pattern and a site graph, which allows us to use homomorphisms from patterns to Σ-graphs.

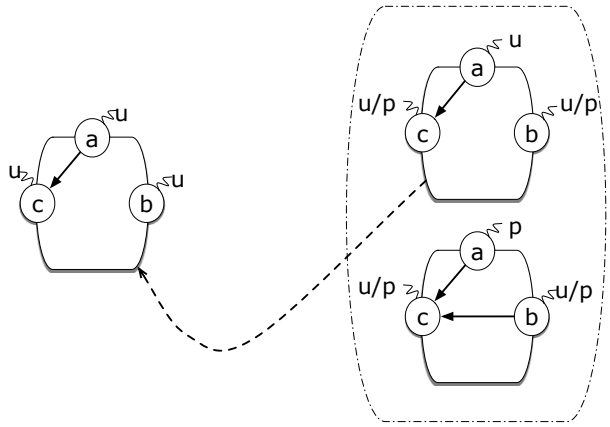
We can now safely define the notion of prefragments, in our context-sensitive framework: given an annotated Σ-graph G_a, a site graph P is a prefragment if we get a directed relation over its sites when we annotate it by the meet of the inverse image of the annotation of G_a by any homomorphism between P and G_a.

Definition 8.2.3. *Let* G_a *be an annotated* Σ*-graph and let* P *be site graph. We define the* canonical annotation of P by the annotated Σ-graph G_a *as follows:*

- *for any* $(i, x), (i', x') \in \mathcal{S}_P$ *and any* $w \in \{\vee, \wedge\}, (i, x) \overset{\square}{\rightsquigarrow}_{P,G_a} (i', x')$ *if and only if for all homomorphisms* $h : P \to G$, $(\phi(i), x) \overset{\square}{\rightsquigarrow}_{G_a} (h(i'), x')$.



(a) Homomorphism from a site graph to an annotated Σ-graph.



(b) Canonical annotation of the site graph.

Figure 8.3: Example of canonical annotation of a site graph by an annotated Σ-graph.

We remind (see Definition 8.2.9) that the symbol $\wedge$ denotes internal edges, whereas the symbol $\vee$ denotes external edges.

We notice that annotating a site graph P according to an annotated Σ-graph G_a amounts to apply the concretization $\gamma_{P,G}$ to G_a.

In Figure 8.3 an example of canonical annotation is showed. Specifically, Figure 8.3(a) shows (by the dotted arrow) all the homomorphisms from a site graph to an annotated Σ-graph; Figure 8.3(b) shows how the pattern is annotated accordingly.

Another example is given in Figure 8.4. There the canonical annotation is the empty one.

Definition 8.2.4 (Prefragment)**.** *Given an annotated Σ-graph G_a, we say that a site graph P is a prefragment for G_a if and only if the set of sites $\mathcal{S}_P$ and the transitive and reflexive closure of the relation $\rightsquigarrow_{P,G_a}$ form a directed set (i.e. for any $s_1, s_2 \in \mathcal{S}_P$, there exists $s \in \mathcal{S}_P$ such that $s_1 \rightsquigarrow^{\star}_{P,G_a} s$ and $s_2 \rightsquigarrow^{\star}_{P,G_a} s$).*

We can observe that prefragments are connected. Moreover, since the set $\mathcal{S}_P$ is finite, a site graph P is a prefragment if and only if there exists a site $s^{\bullet} \in \mathcal{S}_P$ such that for every site $s \in \mathcal{S}_P$, $s \rightsquigarrow^{\star}_{P,G_a} s^{\bullet}$.

In such a case, we say that the site $s^{\bullet}$ is a target of the prefragment ($s^{\bullet}$ may be not unique).

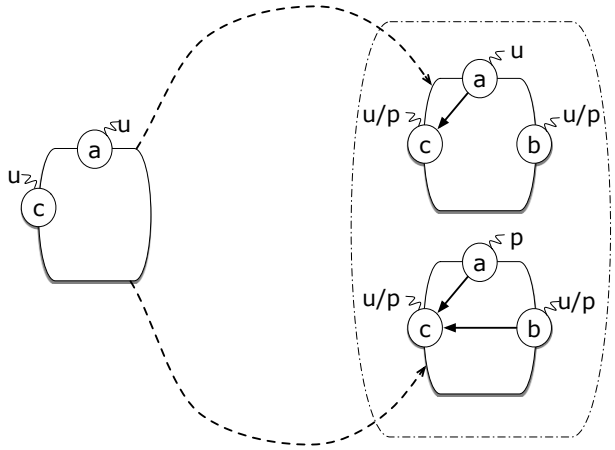
8.2.2 Flow analysis

Now we want to see in which way we should annotate a Σ-graph in order to have a self consistent change of variables. For this purpose, we introduce the concept of *summary graph*.

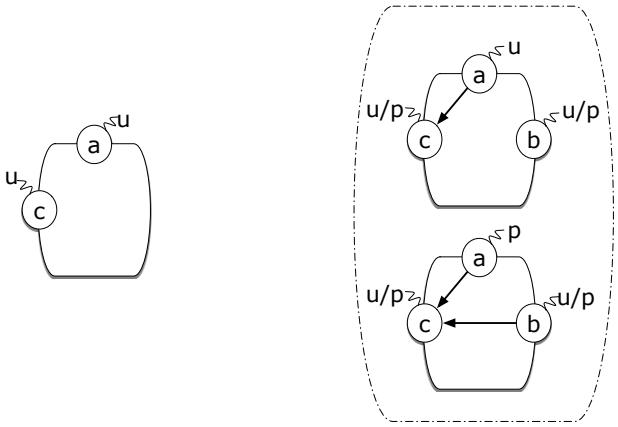
Definition 8.2.5 (summary graph)**.** *A summary graph G is a Σ-graph with this three properties:*

1. *$\mathcal{L} \subseteq (\mathcal{S} \cup \{\dashv\})^2$;*
2. *for every species $V \in \mathcal{V}$, there exists a homomorphism $h : V \to G$;*
3. *for every homomorphism $h : P \to G$ from a connected site graph P to G, there exists a species $V \in \mathcal{V}$, an embedding $\phi : P \hookrightarrow V$ and a homomorphism $h' : V \to G$ such that $h = h'\phi$.*

Roughly speaking, summary graphs are used to abstract information about the potential overlaps between the left and right hand sides of rules and connected site graphs, such that the common region contains sites that are



(a) Homomorphism from a site graph to an annotated Σ-graph.



(b) Canonical annotation of the site graph.

Figure 8.4: Example of canonical annotation by an annotated Σ-graph.

modified by the rules. This will allow us to express the proper consumption and the proper production of these site graphs.

The set of summary graphs is exactly the smallest set of Σ-graphs that contains the disjoint union of the family of species and that is stable by disjoint union and folding of agent nodes (folding of agent nodes can be formalised as the application of a regular epimorphism). It follows that the disjoint union of all species is a summary graph (the most concrete one), and the contact map is also a summary graph (the most abstract one).

Now we study the factors that we need to consider in order to annotate the summary graph in a way that will allow us to derive a sound and complete set of fragments.

First of all, let us introduce some notion.

Definition 8.2.6 (Path in a site graph)**.** *Let P be a site graph. We call a path in P a sequence of steps $(i_1, x_1) \overset{\square_1}{\rightsquigarrow}_{P_\top} \cdots \overset{\square_{n-1}}{\rightsquigarrow}_{P_\top} (i_n, x_n)$ in $\mathcal{S}_P$, such that $((i_k, x_k), (i_{k+1}, x_{k+1})) \in \mathcal{L}_P$ for every k between 1 and $n-1$ that satisfies $\square_k = \vee$.*

Definition 8.2.7 (Alternated path)**.** *A path $(i_1, x_1) \overset{\square_1}{\rightsquigarrow}_{P_\top} \cdots \overset{\square_{n-1}}{\rightsquigarrow}_{P_\top} (i_n, x_n)$ is said to be alternated if for any integer j between 1 and $k-1$:*

$$\square_j = \vee \text{ if and only if } \square_{j+1} = \wedge.$$

Lemma 8.2.8. *Let G_a be an annotated summary graph. Let $\phi : P \to P'$ be a homomorphism between two site graphs P and P'. Let $(i_1, x_1), (i_1, x_2) \in \mathcal{S}_P$ be two sites such that $(i_1, x_1) \rightsquigarrow^\star_{P,G_a} (i_2, x_2)$ then $(\phi(i_1), x_1) \rightsquigarrow^\star_{P',G_a} (\phi(i_2), x_2)$.*

Proof. Let $(i, x), (i', x') \in \mathcal{S}_P$ be two site instances in P and $w \in \{\vee, \wedge\}$ such that $(i, x) \overset{\square}{\rightsquigarrow}_{P,G_a} (i', x')$.

Let h be a homorphism between P' and G. By composition, $h\phi$ is a homorphism between P and G.

Since $(i, x) \overset{\square}{\rightsquigarrow}_{P,G_a} (i', x')$, it follows, by Definition 8.2.3 that:

$$(h(\phi(i)), x) \overset{\square}{\rightsquigarrow}_{G_a} (h(\phi(i')), x').$$

Thus, by Definition 8.2.3, $(\phi(i), x) \overset{\square}{\rightsquigarrow}_{P,G_a} (\phi(i'), x')$.

It follows, by induction, that : $(\phi(i_1), x_1) \rightsquigarrow^\star_{P',G_a} (\phi(i_2), x_2)$. □

Now we have all the elements we need to describe the requirements that an annotated summary graph should have in order to be able to induce a self-consistent change of variables.

Definition 8.2.9. *Let G_a be an annotated summary graph. We say that G_a is compatible with a model if, and only if, the following constraints are satisfied:*

1. *Direct flow:*

   For every non trivial rule $r : L \to R$*, for every homomorphism* h *from the lhs* L *of the rule* r*, to the* Σ*-graph* G*,*

   (a) *for every arc* $(i, x) \overset{\square}{\rightsquigarrow}_{L_\top} (i', x')$ *that occurs in an alternated path in* L*, that ends in a site instance which is modified by the rule* r*, we have:* $(h(i), x) \overset{\square}{\rightsquigarrow}_{G_a} (h(i'), x')$

   (b) *for every site instance* $(A, i, x) \in \mathcal{MAY}(r)$*, for every site site name* x' *such that the pair* (i, x') *denotes a site instance in* L*, we have:* $(h(i), x') \overset{\wedge}{\rightsquigarrow}_{G_a} (h(i), x)$*;*

   (c) *for every site instance* $(A, i, x) \in \mathcal{MAY}(r) \cup \mathcal{MUST}(r)$ *such that* $(A, i, x) \in \mathcal{MAY}(r)$*, for every site instance* $(i', x') \in \mathcal{S}_G$ *such that* $((h(i), x), (i', x')) \in \mathcal{L}_G$*, we have:* $(h(i), x) \overset{\vee}{\rightsquigarrow}_{G_a} (i', x')$*;*

   (d) *for every site instance* $(A, i, x) \in \mathcal{MUST}(r)$ *such that the binding state of the site instance* (A, i, x) *in* L *is the symbol '*$-$*', for every site instance* $(i', x') \in \mathcal{S}_G$ *such that* $((h(i), x), (i', x')) \in \mathcal{L}_G$*, we have:* $(h(i), x) \overset{\vee}{\rightsquigarrow}_{G_a} (i', x')$*;*

   (e) *for every site instance* $(A, i, x) \in \mathcal{MUST}(r)$ *such that the binding state of the site instance* (A, i, x) *in* L *is a binding state that we denote as* $A'@x'$*, and for every site instance* $(i'', x'') \in \mathcal{S}_G$ *such that* $type_G(i'') = A'$*,* $x'' = x'$*, and* $((h(i), x), (i'', x'')) \in \mathcal{L}_G$*, we have:* $(h(i), x') \overset{\vee}{\rightsquigarrow}_{G_a} (i'', x'')$*;*

2. *Indirect flow:*

   For every non trivial rule $r : L \to R$*, for every pattern component* c *in the lhs* L *of the rule* r*, there exists a site instance* (i, x) *in* c *such that, for every site instance* (i', x') *in* c*, we have:* $(i', x') \rightsquigarrow_{L,G_a} (i, x)$*.*

3. *Backward compatibility:*

   For every rule (trivial or not) r*, for every ground refinement* $R_L \to R_R$ *of the rule* r*, for every homomorphism* h_L *from* R_L *to* G*, for every homorphism* h_R *from* R_R *to* G*, for every two paths:*

$$p = (i_1, x_1) \overset{\square 1}{\rightsquigarrow}_{R_{L\top}} \ldots \overset{\square n-1}{\rightsquigarrow}_{R_{L\top}} (i_n, x_n)$$

   and:

$$p' = (i'_1, x'_1) \overset{\square' 1}{\rightsquigarrow}_{R_{L\top}} \ldots \overset{\square' n'-1}{\rightsquigarrow}_{R_{L\top}} (i'_{n'}, x'_{n'})$$

   such that:

- $(i_n, x_n) = (i'_{n'}, x_{n'})$;
- *there exists* k *between* 1 *and* n *such that the site instance:*

$$(type_{R_L}(i_k), i_k, x_k)$$

is modified by the reaction $R_L \to R_R$;

- *for every arc* $(i, x) \overset{\square}{\rightsquigarrow}_{R_{L\top}} (i', x')$ *that occur either in the path* p *or in the path* p'*, the three following conditions are satisfied:*
  - *i.* $(type_{R_R}(i), i, x)$ *and* $(type_{R_R}(i'), i', x')$ *are site instances in* R_R;
  - *ii.* $(i, x) \overset{\square}{\rightsquigarrow}_{R_{R\top}} (i', x')$;
  - *iii.* $(h_R(i), x) \overset{\square}{\rightsquigarrow}_{R_R, G_a} (h_R(i'), x')$;

for every arc $(i, x) \overset{\square}{\rightsquigarrow}_{R_{L\top}} (i', x')$ *occuring in* p *or* p'*, we have:*

$$(h_L(i), x) \overset{\square}{\rightsquigarrow}_{G_a} (h_L(i'), x').$$

Let us give some intuitions about Definition 8.2.9.

Direct flow (constraint 1) is obtained by annotating the image of every alternating path between a site that is tested and a site that is modified (constraint 1a), by every homomorphism between the lhs of a rule and the summary graph G (constraint 1a). These paths are prolongated in order to take into account potential side effects (constraints 1b, 1c, 1d, and 1e). Considering paths allows us to deal with distance control, that is to say agents which test the state of some sites, without modifying the states of any sites. Moreover, only considering alternating paths avoids spurious flows within agents.

Indirect flows (constraint 2) ensure that any pattern component in the lhs of a rule is a prefragment, and this even if it contains no modified site.

Annotating a summary graph so that it copes with direct and indirect flows, could be seen as an abstraction. Let us consider U the disjoint union of all the lhs of rules. Let us ignore side effects for the sake of simplicity. Firstly, we annotate U with every alternated path from a site that is tested to a site that is modified in the lhs of a rule. Then, we complete the annotation of U so that each connected component contains a target. The constraints 1 and 2 ensures that G_a contains at least the annotation of $\alpha_{U,G}$.

Lastly, backward compatibility (constraint 3 ensures that prefragments that overlap with the lhs of rules are always more refined than the ones that overlap with the rhs. This comes from the fact that a prefragment that overlaps with a rhs of a rule on a modified site $s^\bullet$, contains at least one target t. Then we consider another site instance s in the prefragment. The path

p denotes a path between the site $s^\bullet$ and the target t, whereas the path p' denotes a path between the site s and the target t. Thus we copy these paths at any place in the annotated summary graph G_a which matches both with a potential antecedent of the prefragment and the context of application of at least one reaction.

It is worth noting that Constraint 3 of Definition 8.2.9 can be over-approximated, so as to avoid having to refer to ground reactions. Instead, it is enough to explore only the refinements of the rules, that describes the paths of interest, and some contextual information which can be tuned by a parametric strategy (such as refining the agents at distance less than k from the gluing between the rhs of the rule and the two paths of interest). We would get more annotations in the Σ-graph G, but the soundness of our approach would still hold.

Example 8.2.10. *Let us have a look to Figure 8.5. The set of fragments will depend on our choice for the summary graph. If we perform the fragmentation from summary graph (1), which corresponds to the catalogue of species, we obtain the following fragments:*

$$\begin{aligned}
\mathcal{F}_1 &= P(a_p, b_u, c_u),\\
\mathcal{F}_2 &= P(a_p, b_p, c_u),\\
\mathcal{F}_3 &= P(a_u, c_u),\\
\mathcal{F}_4 &= P(a_u),\\
\mathcal{F}_5 &= P(a_p),\\
\mathcal{F}_6 &= P(b_u),\\
\mathcal{F}_7 &= P(b_p),\\
\mathcal{F}_8 &= P(c_p).
\end{aligned}$$

If we start from summary graph (2), we obtain:

$$\begin{aligned}
\mathcal{F}_1 &= P(a_u, b_u, c_u),\\
\mathcal{F}_2 &= P(a_u, b_p, c_u),\\
\mathcal{F}_3 &= P(a_p, b_u, c_u),\\
\mathcal{F}_4 &= P(a_p, b_p, c_u),\\
\mathcal{F}_5 &= P(a_u),\\
\mathcal{F}_6 &= P(a_p),\\
\mathcal{F}_7 &= P(b_u),\\
\mathcal{F}_8 &= P(b_p),\\
\mathcal{F}_9 &= P(c_p).
\end{aligned}$$

If we start from summary graph (3) we obtain the same fragmentation then in the concrete case (1).

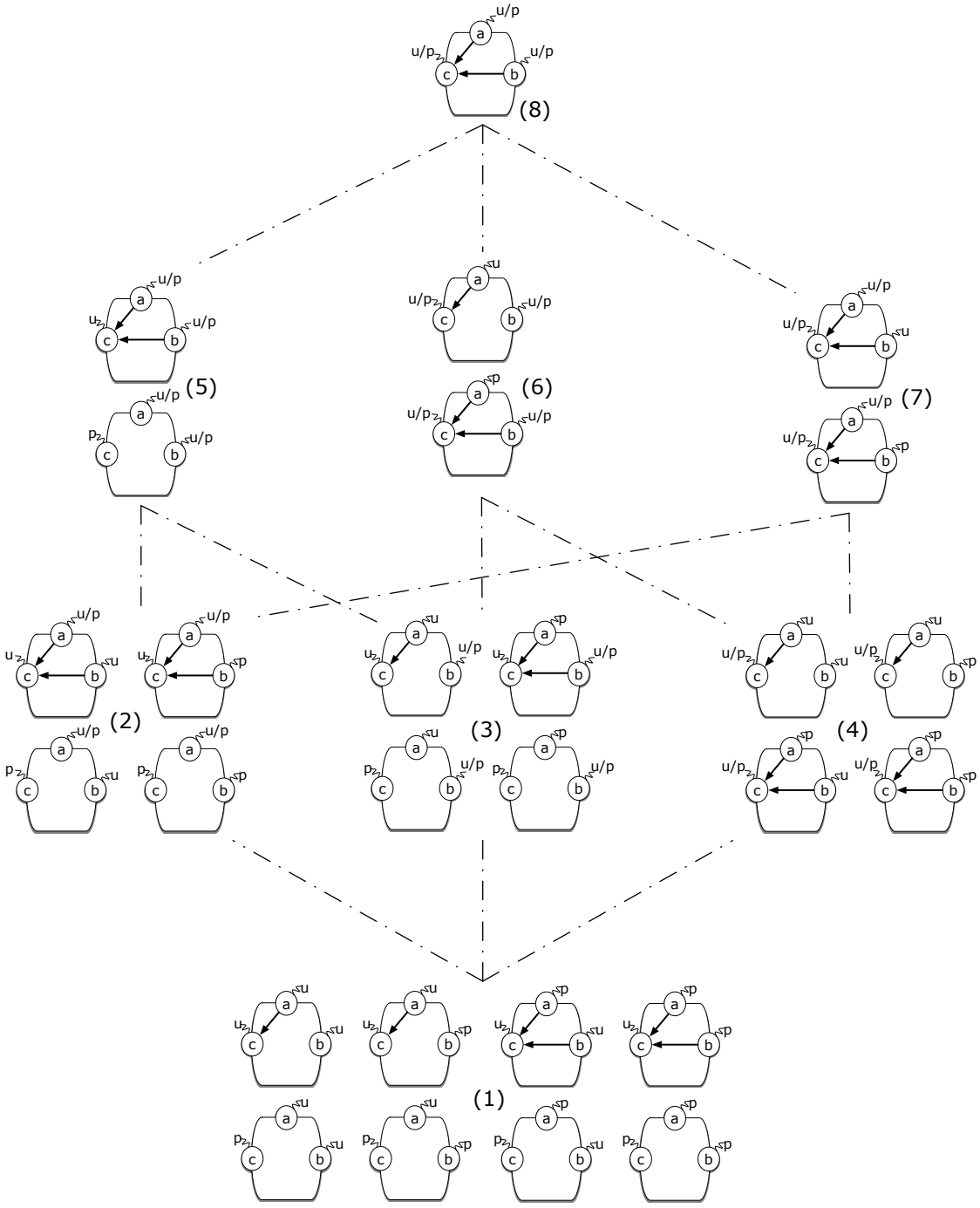


Figure 8.5:

If we start from summary graph (4):

$$\begin{aligned}
\mathcal{F}_1 &= P(a_p, b_u, c_u),\\
\mathcal{F}_2 &= P(a_p, b_u, c_p),\\
\mathcal{F}_3 &= P(a_p, b_p, c_u),\\
\mathcal{F}_4 &= P(a_p, b_p, c_p),\\
\mathcal{F}_5 &= P(a_u, c_u),\\
\mathcal{F}_6 &= P(a_u, c_p),\\
\mathcal{F}_7 &= P(b_u),\\
\mathcal{F}_8 &= P(b_p).
\end{aligned}$$

If we start from summary graph (5) we obtain the same fragmentation then with the summary graph (2).

If we start from summary graph (6):

$$\begin{aligned}
\mathcal{F}_1 &= P(a_u, c_u),\\
\mathcal{F}_2 &= P(a_u, c_p),\\
\mathcal{F}_3 &= P(a_p, b_u, c_u),\\
\mathcal{F}_4 &= P(a_p, b_p, c_u),\\
\mathcal{F}_5 &= P(a_p, b_u, c_p),\\
\mathcal{F}_6 &= P(b_p, b_p, c_p),\\
\mathcal{F}_7 &= P(b_u),\\
\mathcal{F}_8 &= P(b_p).
\end{aligned}$$

If we start from summary graph (7):

$$\begin{aligned}
\mathcal{F}_1 &= P(a_u, b_u, c_u),\\
\mathcal{F}_2 &= P(a_u, b_u, c_p),\\
\mathcal{F}_3 &= P(a_u, b_p, c_u),\\
\mathcal{F}_4 &= P(a_u, b_p, c_p),\\
\mathcal{F}_5 &= P(a_p, b_u, c_u),\\
\mathcal{F}_6 &= P(a_p, b_u, c_p),\\
\mathcal{F}_7 &= P(a_p, b_p, c_u),\\
\mathcal{F}_8 &= P(a_p, b_p, c_p).
\end{aligned}$$

That is exactly our catalogue of species.

Again, if we start the fragmentation from the summary graph (8), that corresponds to the contact map, we obtain the full set of species.

8.3 Reduced model

We recall the notations of Chapter 5.

We assume that the model is made of the following set of rules $\mathcal{R}$:

$$\begin{array}{lcll} r_1 & := & c_{1,1}, \ldots, c_{\omega(1),1} \rightarrow p_{1,1}, \ldots, p_{\theta(1),1} & k_1 \\ r_2 & := & c_{1,2}, \ldots, c_{\omega(2),2} \rightarrow p_{1,2}, \ldots, p_{\theta(2),2} & k_2 \\ \ldots & := & \ldots & \\ r_h & := & c_{1,h}, \ldots, c_{\omega(h),h} \rightarrow p_{1,h}, \ldots, p_{\theta(h),h} & k_h \end{array}$$

where for every rule r_ι, $\omega(\iota)$ returns the number of components in the left hand side and $\theta(\iota)$ returns the number of component in the right hand side.

We denote by $\mathbb{F}$ the semantics function, which maps each state $\rho : \mathcal{V} \rightarrow \mathbb{R}$ to the function of the concentration derivatives $\mathcal{V} \rightarrow \mathbb{R}$

$$\begin{aligned} \mathbf{card}([P,P]) \times \mathbb{F}(\rho)(P) = & \sum_{\iota=1}^{h} \sum_{M \in \overleftarrow{\mathcal{M}}(\iota,P)} \prod_{\chi=1}^{\omega(\iota)} (\!|\overline{c}_{\chi,\iota,F,M}|\!)_\rho \\ & - \sum_{\iota=1}^{h} \gamma_\iota \sum_{j=1}^{\omega(\iota)} \sum_{M \in \overrightarrow{\mathcal{M}}(\iota,j,P)} (\!|c_{j,\iota} \cup_M P|\!)_\rho \prod_{\substack{\chi=1 \\ \chi \neq j}}^{\omega(\iota)} (\!|c_{\chi,\iota}|\!)_\rho \end{aligned}$$

where for every rule index ι, every position j,

- $\overrightarrow{\mathcal{M}}(\iota, j, P)$ denotes the set of the potential extended gluings (e.g. see Definition 5.3.8) between the pattern component $c_{j,\iota}$ and the pattern P;

- $\overleftarrow{\mathcal{M}}(\iota, P)$ is the set of the extended preconditions of the refinements of the rule $rule_\iota$ according to the overlap between the rhs of the rule $rule_\iota$ and the pattern P;

- and for every $M \in \overleftarrow{\mathcal{M}}(\iota, P)$, $(\overline{c}_{\chi,\iota,F,M})_{1 \leqslant \chi \leqslant \omega(\iota)}$ is the list of the pattern components in the extended precondition $pred(\iota, P, M)$.

8.3.1 Contribution of proper consumption

We already know, that the pattern components that occur in the lhs of a rule are all prefragments. Let us prove that if F is a prefragment, ι a rule index, and j a position, and any way $M \in \overrightarrow{\mathcal{M}}(\iota, j, F)$ to glue the prefragment F on the j-th pattern component of the rule $rule_\iota$, then the corrected concentration $(\!|c_{j,\iota} \cup_M F|\!)$ of pattern component $(c_{j,\iota} \cup_M F)$ is either equal to 0 or to the corrected concentration of a prefragment.

Trivial rules We assume that ι is the index of a trivial rule. By definition, the rule $rule_\iota$ has the following form:

$$\mathtt{A_1}(\mathtt{x}^{\mathtt{B_2@y}}), \mathtt{B_2}(\mathtt{y}^{\mathtt{A_1@x}}) \to \mathtt{A_1}(\mathtt{x}), \mathtt{B_2}(\mathtt{y}).$$

Hence, the number $\omega(\iota)$ of pattern components in the lhs of the rule $rule_\iota$ is equal to 1, and j is equal to 1 as well.

We consider $(\mathscr{A}, B) \in \overrightarrow{\mathcal{M}}(\iota, j, F)$ an extended gluing between the pattern component $c_{1,\iota}$ and the prefragment F.

The rule $rule_\iota$ has no side effect.

As a consequence, we have $B = \varnothing$.

By Definition 5.3.8, it follows that $\mathscr{A} \neq \varnothing$.

We consider two cases according to whether or not the set $\mathscr{A}$ is a singleton.

1. Let us assume that the set $\mathscr{A}$ is a singleton.

   It follows that the (extended) gluing between c_1 and F is obtained (up to isomorphism) by replacing a binding type carried by a site instance s, by a link to a fresh agent.

   We wonder what are the different possibilities to refine the binding type of the site instance s in the prefragment F. The site s might be bound to a site instance in F, but the so-refined pattern would contain a cycle, that is to say, since we have assumed that no species contains cycles, that its (corrected) concentration would be equal to 0. The only remaining case, is the one where the site s is bound to a site in a fresh agent. In this case, the refined pattern is isomorphic to the pattern $c_{1,\iota} \cup_{\mathscr{A}} F$.

   We conclude that:

   $$(\![c_{1,\iota} \cup_{(\mathscr{A},B)} F]\!) = (\![F]\!).$$

2. Let us assume that the set $\mathscr{A}$ contains two elements.

   Let us write $\mathscr{A} = \{(1, i_1), (2, i_2)\}$.

   In that case, either there is a link between the site instances (A, i_1, x) and (B, i_2, y) in the prefragment F, or not.

   (a) If there is a link between the site instances (A, i_1, x) and (B, i_2, y) in the prefragment F, then $c_{1,\iota} \cup_{(\mathscr{A},B)} F$ is isomorphic to the pattern component F.

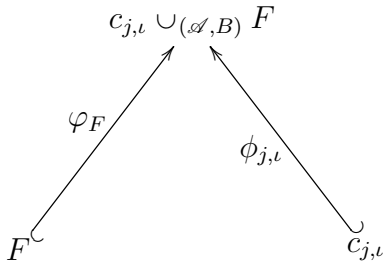
   (b) Otherwise, the pattern component $c_{1,\iota} \cup_{(\mathscr{A},B)} F$ contains a cycle, that is to say, since we have assumed that no species contains cycles, that its (corrected) concentration is equal to 0.

In all case, the corrected concentration $(\!|c_{1,\iota} \cup_{(\mathscr{A},B)} F|\!)$ is either equal to 0, or to the corrected concentration of a prefragment.

Non-trivial rules We assume that ι is the index of a non trivial rule. We consider $(\mathscr{A}, B) \in \overrightarrow{\mathcal{M}}(\iota, j, F)$ an extended gluing between the pattern component $c_{j,\iota}$.

Thanks to Constraint 2 of Definition 8.2.9, for every position χ such that $j \neq \chi$, the pattern component $c_{\chi,\iota}$ is a prefragment. We are left to express the (corrected) concentration of the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$.

We consider φ_F the embedding from the prefragment F to the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$ and $\phi_{j,\iota}$ be the embedding from the pattern component $c_{j,\iota}$ and the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$, that have been used to define the extended gluing $c_{j,\iota} \cup_{(\mathscr{A},B)} F$.



The pattern component F is a prefragment, thus there exists a site instance (A_t, i_t, x_t) in F that is a target. We consider two cases according to whether the set $\mathscr{A}$ is empty, or not.

1. We assume that the set $\mathscr{A}$ is not empty.

   By Definition 5.3.8, the set B is empty.

   That is to say that: $c_{j,\iota} \cup_{(\mathscr{A},B)} F = c_{j,\iota} \cup_{\mathscr{A}} F$.

   By Definition 5.3.5, for every site instance (A, i, x) in $c_{j,\iota} \cup_{\mathscr{A}} F$, we know that either there exists an agent identifier i' such that (A, i', x) is a site instance in F and $i = \varphi_F(i')$, or that there exists an agent identifier i' such that (A, i', x) is a site instance in $c_{j,\iota}$ and $i = \phi_{j,\iota}(i')$.

   We propose to show that the site instance $(A_t, \varphi_F(i_t), x_t)$ is a target in the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$.

   Let (A_s, i_s, x_s) be a site instance in the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$.

   (a) We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in F and $i_s = \varphi_F(i')$.

Since (A_t, i_t, x_t) is a target in F, we have:

$$(i'_s, x_s) \rightsquigarrow^{\star}_{F,G_a} (i_t, x_t).$$

By Lemma 8.2.8, we have:

$$(i_s, x_s) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{\mathscr{A}} F,G_a} (\varphi_F(i_t), x_t).$$

(b) We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in $c_{j,\iota}$ and $i_s = \phi_{j,\iota}(i')$.
By Definition 5.3.8, there exists a site instance (A_m, i_m, x_m) in $c_{j,\iota}$ and an agent identifier i'_m such that:

- (A_m, i'_m, x_m) is a site instance in F;
- $\varphi_F(i'_m) = \phi_{j,\iota}(i_m)$;
- the site instance (A_m, i_m, x_m) is modified by the rule $rule_\iota$.

By Constraint 1 of Definition 8.2.9, we have:

$$(A_s, i', x_s) \rightsquigarrow^{\star}_{c_{j,\iota},G_a} (A_m, i_m, x_m).$$

By Lemma 8.2.8, it follows that:

$$(A_s, i_s, x_s) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{\mathscr{A}} F,G_a} (A_m, \phi_{j,\iota}(i_m), x_m).$$

By the previous case, we know that:

$$(A_m, \varphi_F(i'_m), x_m) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{\mathscr{A}} F,G_a} (A_t, \varphi_F(i_t), x_t).$$

Since $\varphi_F(i'_m) = \phi_{j,\iota}(i_m)$, we can conclude that:

$$(A_s, i_s, x_s) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{\mathscr{A}} F,G_a} (A_t, \varphi_F(i_t), x_t).$$

2. We assume that the set $\mathscr{A}$ is empty.
The set B is a singleton that we write $((A_{B_1}, i_{B_1}, x_{B_1}), (A_{B_2}, i_{B_2}, x_{B_2}))$.
By Definition 5.3.5, we know that for any site instance (A, i, x) in $c_{j,\iota} \cup_{(\mathscr{A},B)} F$, at least one of the following condition is satisfied:

- either there exists an agent identifier i' such that (A, i', x) is a site instance in F and $i = \varphi_F(i')$,
- or there exists an agent identifier i' such that (A, i', x) is a site instance in $c_{j,\iota}$ and $i = \phi_{j,\iota}(i')$,
- or $(A, i, x) = (A_{B_1}, \phi_{j,\iota}(i_{B_1}), x_{B_1})$.

Let us show that the site instance $(A_t, \varphi_F(i_t), x_t)$ is a target in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$.

Let (A_s, i_s, x_s) be a site instance in the pattern component $c_{j,\iota} \cup_{(\mathscr{A},B)} F$.

(a) We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in F and $i_s = \varphi_F(i')$.
Since (A_t, i_t, x_t) is a target in F, we have:

$$(A_s, i'_s, x_s) \rightsquigarrow^{\star}_{F,G_a} (A_t, i_t, x_t).$$

By Lemma 8.2.8, it follows that:

$$(A_s, i_s, x_s) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F,G_a} (A_t, \varphi_F(i_t), x_t).$$

(b) We assume that $(A_s, i_s, x_s) = (A_{B_1}, \phi_{j,\iota}(i_{B_1}), x_{B_1})$.
By Constraint 1c of Definition 8.2.9 and thanks to Lemma 8.2.8, we have:

$$(A_s, i_s, x_s) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F,G_a} (A_{B_2}, \varphi_F(i_{B_2}), x_{B_2}).$$

By the previous case, we know that:

$$(A_{B_2}, \varphi_F(i'_{B_2}), x_{B_2}) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F,G_a} (A_t, \varphi_F(i_t), x_t).$$

Thus, we can conclude that:

$$(A_s, i_s, x_s) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F,G_a} (A_t, \varphi_F(i_t), x_t).$$

(c) We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in $c_{j,\iota}$ such that $i_s = \phi_{j,\iota}(i')$.

- If $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MAY}(rule_\iota)$.
The agent instance (A_{B_1}, i_{B_1}) has at least one site instance (otherwise, the pattern component $c_{j,\iota}$ would have been made of a single agent with no site instance, which would be absurd since (A_s, i', x_s) is a site instance in $c_{j,\iota}$).
Let x_w be a site name, such that the triple (A_{B_1}, i_{B_1}, x_w) is a site instance in the pattern $c_{j,\iota}$.
We have $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MAY}(rule_\iota)$, so the agent instance (A_{B_1}, i_{B_1}) is removed by the rule $rule_\iota$. As a consequence, the site (A_{B_1,i_{B_1},x_w}) is modified by the rule $rule_\iota$.

By Constraint 1 of Definition 8.2.9, we have:

$$(A_s, i', x_s) \rightsquigarrow^{\star}_{c_{j,\iota}, G_a} (A_{B_1}, i_{B_1}, x_w).$$

By Lemma 8.2.8, it follows that:

$$(A_s, i_s, x_s) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F, G_a} (A_{B_1}, \phi_{j,\iota}(i_{B_1}), x_w).$$

By Constraint 1b of Definition 8.2.9 and Lemma 8.2.8, we have:

$$(A_{B_1}, \phi_{j,\iota}(i_{B_1}), x_w) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F, G_a} (A_{B_1}, \phi_{j,\iota}(i_{B_1}), x_{B_1}).$$

Thus, we have:

$$(A_{B_1}, \phi_{j,\iota}(i_{B_1}), x_{B_1}) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F, G_a} (A_t, \varphi_F(i_t), x_t).$$

Thus, we can conclude that:

$$(A_s, i_s, x_s) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F, G_a} (A_t, \varphi_F(i_t), x_t).$$

- If the site instance $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MUST}(rule_\iota)$.
  The site instance $(A_{B_1}, i_{B_1}, x_{B_1})$ is modified by the rule $rule_\iota$.
  By Constraint 1 of Definition 8.2.9, we have:

  $$(A_s, i', x_s) \rightsquigarrow^{\star}_{c_{j,\iota}, G_a} (A_{B_1}, i_{B_1}, x_{B_1}).$$

  By Lemma 8.2.8, we can conclude that:

  $$(A_s, i_s, x_s) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F, G_a} (A_{B_1}, \phi_{j,\iota}(i_{B_1}), x_{B_1}).$$

  By the previous case, we know that:

  $$(A_{B_1}, \phi_{j,\iota}(i_{B_1}), x_{B_1}) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F, G_a} (A_t, \varphi_F(i_t), x_t).$$

  Thus, we can conclude that:

  $$(A_s, i_s, x_s) \rightsquigarrow^{\star}_{c_{j,\iota} \cup_{(\mathscr{A},B)} F, G_a} (A_t, \varphi_F(i_t), x_t).$$

As a consequence, the site instance $(A_t, \varphi_F(i_t), x_t)$ is a target for the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$. That is to say that the pattern component $c_{j,\iota} \cup_{\mathscr{A}} F$ is a prefragment.

8.3.2 Contribution of proper production

Let us prove that for every rule index ι, every prefragment F, every way $M \in \overleftarrow{\mathcal{M}}(\iota, F)$ of producing the prefragment F with the rule $rule_\iota$, for every position χ, the (corrected) concentration $(\!|\overline{c}_{\chi,\iota,F,M}|\!)$ of the χ-th pattern component of the corresponding refinement of the rule $rule_\iota$, is equal to 0 or to the corrected concentration of a prefragment.

Trivial rules

Let ι be the index of a trivial rule and F be a prefragment. By definition, the rule $rule_\iota$ has the following form:

$$\mathtt{A_1(x^{B_2@y})}, \mathtt{B_2(y^{A_1@x})} \rightarrow \mathtt{A_1(x)}, \mathtt{B_2(y)}.$$

Hence, the number $\omega(\iota)$ of pattern components in the lhs of the rule $rule_\iota$ is equal to 1.

We consider a pair $(\mathscr{A}, B) \in \overleftarrow{\mathcal{M}}(\iota, F)$.

The rule $rule_\iota$ has no side effect.

As a consequence, we have $B = \varnothing$.

By Definition 5.4.2, it follows that $\mathscr{A} \neq \varnothing$.

We consider two cases according to whether or not the set $\mathscr{A}$ is a singleton.

1. Let us assume that the set $\mathscr{A}$ is a singleton.

   We denote $\mathscr{A} = ((A_{m_1}, i_{m_1}, s_{m_1}), (A_{m_2}, i_{m_2}, s_{m_2}))$ It follows that the prefragment F contains a site instance s that is free. Then the pattern component $\overline{c}_{1,\iota,F,M}$ is obtained (up to isomorphism) by binding in F the site instance s to a link to a fresh agent.

   We denote as F' the pattern that is obtained by replacing, in the prefragment F the binding state of the site instance s with the binding type that matches the link that is created by the rule.

   Let us prove that the pattern F' is a prefragment.

   - The pattern component F is a prefragment.

     Thus there exists a target (A_t, i_t, x_t).

     Let h_ℓ be a homomorphism between F' and G.

     The only difference between the pattern components F' and $\overline{c}_{1,\iota,F,M}$ is that a binding type in F' is replaced with a bond to a fresh agent.

Since a summary graph contains only free sites and explicit bounds, we can deduce that h_ℓ can be extended into an homomorphism h'_ℓ from $\overline{c}_{1,\iota,F,M}$ to G.

We denote as φ_F an embedding between F' and $\overline{c}_{1,\iota,F,M}$ such that: $h'_\ell \varphi_F = h_\ell$.

By Constraint 3 of Definition 8.2.5, there exists a species $V \in \mathcal{V}$, an embedding ϕ from $\overline{c}_{1,\iota,F,M}$ to V and an homomorphism h''_ℓ from V to G such that $h'_\ell = h''_\ell \phi$.

The embedding ϕ defines $R_L \to R_R$ a ground refinement of the rule.

By constraint 2 of Definition 8.2.5, there exists a homomorphism h_R between R_R and G.

Let (A_s, i_s, x_s) be a site instance in F'.

Since (A_t, i_t, x_t) is a target in F, there exist two paths:

$$(i_1, x_1) \overset{\square_1}{\rightsquigarrow}_{F,G_a} \dots \overset{\square_{n-1}}{\rightsquigarrow}_{F,G_a} (i_n, x_n)$$

and:

$$(i'_1, x'_1) \overset{\square'_1}{\rightsquigarrow}_{F,G_a} \dots \overset{\square'_{n'-1}}{\rightsquigarrow}_{F,G_a} (i'_{n'}, x'_{n'})$$

such that:

- $(i_1, x_1) = (i_{m_2}, x_{m_2})$;
- $(i_n, x_n) = (i_t, x_t)$;
- $(i'_1, x'_1) = (i_s, x_s)$;
- $(i'_{n'}, x_{n'}) = (i_t, x_t)$.

It follows that the rule $rule_\iota$, the ground refinement $R_L \to R_R$, the homomorphisms h''_ℓ and h_R, and both paths:

$$(\phi(\varphi_F(i_1)), x_1) \overset{\square_1}{\rightsquigarrow}_{R_{L\top}} \dots \overset{\square_{n-1}}{\rightsquigarrow}_{R_{L\top}} (\phi(\varphi_F(i_n)), x_n)$$

and:

$$(\phi(\varphi_F(i'_1)), x'_1) \overset{\square'_1}{\rightsquigarrow}_{R_{L\top}} \dots \overset{\square'_{n'-1}}{\rightsquigarrow}_{R_{L\top}} (\phi(\varphi_F(i'_{n'})), x'_{n'})$$

satisfies the assumptions in Constraint 3 of Definition 8.2.9.

Thus, for k between 1 and n', we have:

$$(h''_\ell(\phi(\varphi_F(i'_k))), x'_k) \overset{\square'_k}{\rightsquigarrow}_{G_a} (h''_\ell(\phi(\varphi_F(i'_{k+1}))), x'_{k+1}).$$

It follows that:

$$(h'_\ell(\varphi_F(i'_k)), x'_k) \overset{\square'_k}{\rightsquigarrow}_{G_a} (h'_\ell(\varphi_F(i'_{k+1})), x'_{k+1}).$$

Then:

$$(h_\ell(i'_k), x'_k) \overset{\square'_k}{\rightsquigarrow}_{G_a} (h_\ell(i'_{k+1}), x'_{k+1}).$$

We have proved that:

$$(h_\ell(i_s), x_s) \overset{\square}{\rightsquigarrow}^\star_{G_a} (h_\ell(i_t), x_t),$$

for every homomorphism h_ℓ from F' to G.
That is to say that:

$$(i_s, x_s) \overset{\square}{\rightsquigarrow}^\star_{F',G_a} (i_t, x_t)$$

By Constraint 3 of Definition 8.2.9, the pattern F' is a prefragment.

We wonder what are the different possibilities to refine the binding type of the site instance s in the prefragment F'. The site s might be bound to a site instance in F, but the so-refined pattern would contain a cycle, that is to say, since we have assumed that no species contains cycles, that its (corrected) concentration would be equal to 0. The only remaining case, is the one where the site s is bound to a site in a fresh agent. In this case, the refined pattern is isomorphic to the pattern $\overline{c}_{1,\iota,F,M}$.

We conclude that

$$(\![\overline{c}_{1,\iota,F,M}]\!) = (\![F']\!).$$

2. Let us assume that the set $\mathscr{A}$ contains two elements.

   Then the pattern component $\overline{c}_{1,\iota,F,M}$ is obtained (up to isomorphism) by binding in F two site instances. As a consequence the pattern component $\overline{c}_{1,\iota,F,M}$ has a cycle.

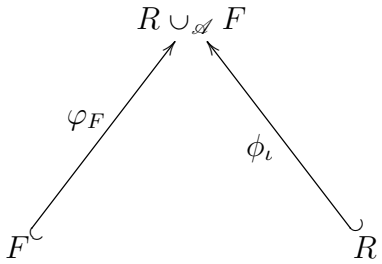
   Since we have assumed that species has no cycle, we can conclude that the concentration of the pattern component $\overline{c}_{1,\iota,F,M}$ is equal to 0.

In all case, the corrected concentration $(\![\overline{c}_{1,\iota,F,M}]\!)$ is equal to 0, or to the corrected concentration of a prefragment.

Non-trivial rules

We assume that ι is the index of a non trivial rule. We denote by $L \to R$ the rule $rule_\iota$ We consider $(\mathscr{A}, B)$ an element of the set $\in \overleftarrow{\mathcal{M}}(\iota, F)$.

We consider φ_F the embedding from the prefragment F to the pattern component $R \cup_{\mathscr{A}} F$ and ϕ_ι be the embedding from the pattern R to the pattern component $R \cup_{\mathscr{A}} F$, that we have used to define the gluing between the fragment F and the rhs R of the rule $rule_\iota$.



Without any loss of generality, we assume that the pattern $R \cup_{\mathscr{A}} F$ matches the pattern R, that is to say that the injective substitution ϕ_ι maps pairs (A, i) to the agent identifiers i.

The pattern component F is a prefragment, thus there exists a site instance (A_t, i_t, x_t) in F that is a target.

Let χ be a natural number 1 and $\omega(\iota)$.

We want to prove that the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is a prefragment.

1. We assume that the set $\mathscr{A}$ is not empty.

   We denote as F^{-1} the antecedent of F.

   Formally, F^{-1} is defined as the pattern that contains every agent instance $(A, \varphi_F(i))$ such that (A, i) is an agent instance in F and the agent instance $(A, \varphi_F(i))$ has not been created by the application of the rule $rule_\iota$, every site instance $(A, \varphi_F(i), x)$ such that (A, i, x) is a site instance in F and the agent instance $(A, \varphi_F(i))$ has not been created by the application of the rule $rule_\iota$, and every bond between the two site instances $(A, \varphi_F(i), x)$ and $(A', \varphi_F(i'), x')$ such that there is a bond between the site instance (A, i, x) and (A', i', x') in F and the bond between the site instance $(A, \varphi_F(i), x)$ and $(A', \varphi_F(i'), x')$ has not been created by the rule $rule_\iota$.

   We denote as F_χ^{-1} the pattern that is made of every pattern component of F^{-1} that contains at least one agent instance $(A, \phi_\iota(i))$ for a given site instance (A, i) in $c_{\chi,\iota}$.

   Let us prove that any connected component F_{pc}^{-1} of F_χ^{-1} contains a site that has been modified by the rule $rule_\iota$.

- This is the case if F^{-1} is connected (by assumption, since $B = \varnothing$, F and P overlaps on a site that is modified by the rule $rule_\iota$).
- If F_{pc}^{-1} is disconnected, each connected component of it contains a site the binding site of which has changed.

Then, we propose to prove both following intermediary results:

(a) Whenever a pattern component of F_χ^{-1} contains the site instance $(A_t, \varphi_F(i_t), x_t)$, for every site instance (A_s, i_s, t_s) of this pattern component, we have:

$$(i_s, t_s) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

(b) Whenever a pattern component F_{pc}^{-1} of F_χ^{-1} does not contain the site instance $(A_t, \varphi_F(i_t), x_t)$, then it contains a site instance $(A_\bullet, i_\bullet, x_\bullet)$ such that the following properties are all satisfied:

- there exists an agent identifier i such that $(A_\bullet, i, x_\bullet)$ is a site instance in F and $i_\bullet = \varphi_F(i)$;
- there exists an agent identifier i such that $(A_\bullet, i, x_\bullet)$ is a site instance in $c_{\chi,\iota}$ and $i_\bullet = \phi_\iota(i)$;
- for every site instance (A_s, i_s, x_s) of F_{pc}^{-1}, we have:

$$(i_s, x_s) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (i_\bullet, x_\bullet).$$

Let us prove these properties:

(a) Let F_{pc}^{-1} be a pattern component of F_χ^{-1} that contains the site instance $(A_t, \varphi_F(i_t), x_t)$.

Let (A_s, i_s, x_s) be a site instance in F_{pc}^{-1}.

Let i' be an agent identifier such that (A_s, i', x_s) is a site instance in F and $\varphi_F(i') = i_s$.

Since the site instance (A_t, i_t, x_t) is a target in F, we have:

$$(i', x_s) \rightsquigarrow^\star_{F,G_a} (i_t, x_t).$$

Let us prove that:

$$(i_s, x_s) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

- The pattern component F is a prefragment.
  Let h_ℓ be a homomorphism between $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ and G.

By Constraint 3 of Definition 8.2.5, there exists a species $V \in \mathcal{V}$, an embedding ϕ from $\overline{c}_{1,\iota,F,M}$ to V and an homomorphism h'_ℓ from V to G such that $h_\ell = h'_\ell \phi$.
The embedding ϕ defines $R_L \to R_R$ a ground refinement of the rule.
By constraint 2 of Definition 8.2.5, there exists a homomorphism h_R between R_R and G.
Let (A_m, i_m, x_m) be a site instance in F_{pc}^{-1} that has been modified by the rule $rule_\iota$.
Since (A_t, i_t, x_t) is a target in F, there exist two paths:

$$(i_1, x_1) \overset{\square_1}{\rightsquigarrow}_{F,G_a} \dots \overset{\square_{n-1}}{\rightsquigarrow}_{F,G_a} (i_n, x_n)$$

and:

$$(i'_1, x'_1) \overset{\square'_1}{\rightsquigarrow}_{F,G_a} \dots \overset{\square'_{n'-1}}{\rightsquigarrow}_{F,G_a} (i'_{n'}, x'_{n'})$$

such that:

- $(i_1, x_1) = (i_m, x_m)$;
- $(i_n, x_n) = (i_t, x_t)$;
- $(i'_1, x'_1) = (i', x_s)$;
- $(i'_{n'}, x_{n'}) = (i_t, x_t)$.

It follows that the rule $rule_\iota$, the ground refinement $R_L \to R_R$, the homomorphisms h'_ℓ and h_R, and both paths:

$$(\phi(\varphi_F(i_1)), x_1) \overset{\square_1}{\rightsquigarrow}_{R_{L\top}} \dots \overset{\square_{n-1}}{\rightsquigarrow}_{R_{L\top}} (\phi(\varphi_F(i_n)), x_n)$$

and:

$$(\phi(\varphi_F(i'_1)), x'_1) \overset{\square'_1}{\rightsquigarrow}_{R_{L\top}} \dots \overset{\square'_{n'-1}}{\rightsquigarrow}_{R_{L\top}} (\phi(\varphi_F(i'_{n'})), x'_{n'})$$

satisfies the assumptions in Constraint 3 of Definition 8.2.9.
Thus, for k between 1 and n', we have:

$$(h'_\ell(\phi(\varphi_F(i'_k))), x'_k) \overset{\square'_k}{\rightsquigarrow}_{G_a} (h'_\ell(\phi(\varphi_F(i'_{k+1}))), x'_{k+1}).$$

Then:

$$(h_\ell(\varphi_F(i'_k)), x'_k) \overset{\square'_k}{\rightsquigarrow}_{G_a} (h_\ell(\varphi_F(i'_{k+1})), x'_{k+1}).$$

We have proved that:

$$(h_\ell(i_s), x_s) \overset{\square}{\rightsquigarrow}^\star_{G_a} (h_\ell(\varphi_F(i_t), x_t),$$

for every homomorphism h_ℓ from $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$to G.
Thus, we can conclude that:

$$(i_s, x_s) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

(b) Let F_{pc}^{-1} be a pattern component of F_{χ}^{-1} that does not contain the site instance $(A_t, \varphi_F(i_t), x_t)$.

Necessarily, the pattern component F_{pc}^{-1} has been disconnected from the rest of F.

That is to say that it contains a site the binding state of which has changed.

Let $(A_{\bullet_t}, i_{\bullet_t}, x_{\bullet_t})$ be a site instance in F_{pc}^{-1} that has lost a bond.

Since $(A_{\bullet_t}, i_{\bullet_t}, x_{\bullet_t})$ is a site instance in F_{pc}^{-1} there exists an agent identifier $i'_{\bullet_t}$ such that $(A_{\bullet_t}, i'_{\bullet_t}, x_{\bullet_t})$ is a site instance in F and $i_{\bullet_t} = \varphi_F(i'_{\bullet_t})$.

It is not possible to create a bond in a rule, without having the two corresponding site instances in the lhs of this rule.

As a consequence, there exists an agent identifier $i''_{\bullet_t}$ such that $(A_{\bullet_t}, i''_{\bullet_t}, x_{\bullet_t})$ is a site instance in $c_{\chi,\iota}$ and $i_{\bullet_t} = \phi_\iota(i''_{\bullet_t})$.

Let $(A_{\bullet_s}), i_{\bullet_s}, x_{\bullet_s})$ a site instance in F_{pc}^{-1}, and i'_s be an agent identifier such that $(A_{\bullet_s}, i'_s, x_{\bullet_s})$ is a site instance in F and $i_{\bullet_s} = \varphi_F(i'_s)$.

Since F is acyclic, there is path:

$$(i_{\bullet_s}, x_{\bullet_s}) \rightsquigarrow^{\star}_{\mathrm{F},G_a} (i_{\bullet_t}, x_{\bullet_t}).$$

that only passes through sites (i, x) such that the site instance $(type_F(i), \varphi_F(i), x)$ belongs to the pattern component F_{pc}^{-1}.

By Constraint 3 of Definition 8.2.9 (we skip some steps, since the reasoning is very similar to the previous ones), we can conclude that:

$$(A_{\bullet_s}, i_{\bullet_s}, x_{\bullet_s}) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (A_{\bullet_t}, i_{\bullet_t}, x_{\bullet_t}).$$

Let (A_s, i_s, x_s) be a site instance in the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$, one of the following property is satisfied:

(a) (A_s, i_s, x_s) is a site instance in F_{χ}^{-1};

(b) there exists an agent identifier i'_s such that (A_s, i'_s, x_s) is $c_{\chi,\iota}$ and $i_s = \phi_\iota(i'_s)$.

We consider several cases according to whether the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ contains the site instance $(A_t, \varphi_F(i_t), x_t)$, or not, and according to whether or not it has been modified by the rule $rule_\iota$.

(a) We assume that the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ contains the site instance $(A_t, \varphi_F(i_t), x_t)$.

We will prove that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

i. We have already proved it if the site instance (A_s, i_s, x_s) belongs to a pattern component of F_χ^{-1} that contains the site instance $(A_t, \varphi_F(i_t), x_t)$.

ii. If there exists an agent identifier i such that (A_s, i, x_s) is a site instance of $c_{\chi,\iota}$ and $i_s = \phi_\iota(i)$.
There exists a site instance (A_m, i_m, x_m) in $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ satisfying the following constraints:

   A. there exists an agent identifier i_{m_1} such that the site instance $(A_m, i_{m'}, x_m)$ belongs to the pattern component $c_{\chi,\iota}$ and $i_m = \varphi_\iota(i_{m'})$;

   B. the pattern component of F_χ^{-1} that contains the site instance $(A_t, \varphi_F(i_t), x_t)$ also contains the the site instance (A_m, i_m, x_m);

   C. the site $(A_m, i_{m'}, x_m)$ is modified by the rule $rule_\iota$.

   Then, by Constraint 1 of Definition 8.2.9, it follows that:

$$(i, x_s) \rightsquigarrow^{\star}_{c_{\chi,\iota},G_a} (i_{m'}, x_m).$$

   By Lemma 8.2.8, we get that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (i_m, x_m).$$

   Thus, from the precedent case, we can conclude that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

iii. If (A_s, i_s, x_s) belongs to a pattern component of F_χ^{-1} that does not contain the site instance $(A_t, \varphi_F(i_t), x_t)$.
Then, there exists a site (A_m, i_m, x_m) in $c_{\chi,\iota}$ such that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\phi_\iota(i_m), x_m).$$

   Thus, from the precedent case, we can conclude that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

Thus, the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is a prefragment.

(b) We assume that: $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)} = c_{\chi,\iota}$.
Then, by Constraint 2 of Definition 8.2.9, $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is a prefragment.

(c) We assume that $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)} \neq c_{\chi,\iota}$ and that the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ does not contain the site instance $(A_t, \varphi_F(i_t), x_t)$.
Let (A_m, i_m, x_m) be a site instance in $c_{\chi,\iota}$ that has been modified by the rule $rule_\iota$.
We will prove that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\phi_\iota(i_m), x_m).$$

i. If there exists an agent identifier i such that (A_s, i, x_s) is a site instance of $c_{\chi,\iota}$ and $i_s = \phi_\iota(i)$.
By Constraint 1 of Definition 8.2.9, we have:

$$(i, x_s) \rightsquigarrow^{\star}_{c_{\chi,\iota},G_a} (i_{m'}, x_m).$$

By Lemma 8.2.8, it follows that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\phi_\iota(i_m), x_m).$$

ii. If (A_s, i_s, x_s) belongs to a pattern component of F_χ^{-1}.
There exists a site $(A_\bullet, i_\bullet, x_\bullet)$ in $c_{\chi,\iota}$ such that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\phi_\iota(i_\bullet), x_\bullet).$$

Thus, from the precedent case, we can conclude that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\phi_\iota(i_m), x_m).$$

Thus, the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is a prefragment.

2. We assume that the set $\mathscr{A}$ is empty.
The set B is a singleton that we write $((A_{B_1}, i_{B_1}, x_{B_1}), (A_{B_2}, i_{B_2}, x_{B_2}))$.
Let χ be a natural number between 1 and $\omega(\iota)$.

(a) If the pattern component $c_{\chi,\iota}$ does not contain the agent instance (A_{B_1}, i_{B_1}).
Then we have $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)} = c_{\chi,\iota}$.
By Constraint 2 of Definition 8.2.9, the pattern component $c_{\chi,\iota}$ is a prefragment.
Thus, $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)} = c_{\chi,\iota}$.
We can conclude that $(\![\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}]\!) = (\![c_{\chi,\iota}]\!)$.

(b) If the pattern component $c_{\chi,\iota}$ contains the agent instance (A_{B_1}, i_{B_1}).
The pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is equal up to isomorphism to the pattern component $(c_{\chi,\iota} \cup_{\varnothing} F)^{\uparrow}_{\{(A_{B_1},i_{B_1}),(A_{B_2},i_{B_2})\}}$ (that is to say, the disjoint union between $c_{\chi,\iota}$ and F, to which we have added a bond between the site instance $(A_{B_1}, i_{B_1}, x_{B_1})$ and the site instance $(A_{B_2}, i_{B_2}, x_{B_2})$.
We want to prove that the site $(A_t, \varphi_F(i_t), x_t)$ is a target in the pattern component $(\!|\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}|\!)$.
Let (A_s, i_s, x_s) be a site instance in $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$.

i. We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in F and $i_s = \varphi_F(i')$.
Since (A_t, i_t, x_t) is a target in F, we have:

$$(i'_s, x_s) \rightsquigarrow^{\star}_{F,G_a} (i_t, x_t).$$

By Constraint 3 of Definition 8.2.9 (we skip some steps, since the reasoning is very similar to the previous ones), we can conclude that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

ii. We assume that $(A_s, i_s, x_s) = (A_{B_1}, \phi_\iota(i_{B_1}), x_{B_1})$.
By Constraint 1c of Definition 8.2.9, we have:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_{B_2}), x_{B_2}).$$

By the previous case, we know that:

$$(\varphi_F(i'_{B_2}), x_{B_2}) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

By Lemma 6.3.3, we can conclude that:

$$(i_s, x_s) \rightsquigarrow^{\star}_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

iii. We assume that there exists an agent identifier i' such that the triple (A_s, i', x_s) is a site instance in $c_{j,\iota}$ such that $i_s = \phi_\iota(i')$.

- If $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MAY}(rule_\iota)$.
The agent instance (A_{B_1}, i_{B_1}) has at least one site instance (otherwise, the pattern component $c_{j,\iota}$ would have been made of a single agent with no site instance, which would be absurd since (A_s, i', x_s) is a site instance in $c_{j,\iota}$).

Let x_w be a site name, such that the triple (A_{B_1}, i_{B_1}, x_w) is a site instance in the pattern $c_{j,\iota}$.
We have $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MAY}(rule_\iota)$, so the agent instance (A_{B_1}, i_{B_1}) is removed by the rule $rule_\iota$. As a consequence, the site (A_{B_1,i_{B_1},x_w}) is modified by the rule $rule_\iota$.
By Constraint 1 of Definition 8.2.9, we have:

$$(i', x_s) \rightsquigarrow^\star_{c_{j,\iota},G_a} (i_{B_1}, x_w).$$

By Lemma 8.2.8, it follows that:

$$(i_s, x_s) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\phi_\iota(i_{B_1}), x_w).$$

By Constraint 1b of Definition 8.2.9, we have:

$$(\phi_\iota(A_{B_1}, i_{B_1}), x_w) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\phi_\iota(i_{B_1}), x_{B_1}).$$

By the previous case, we know that:

$$(\phi_\iota(i_{B_1}), x_{B_1}) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

Thus, we can conclude that:

$$(i_s, x_s) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

- If the site instance $(A_{B_1}, i_{B_1}, x_{B_1}) \in \mathcal{MUST}(rule_\iota)$.
The site instance $(A_{B_1}, i_{B_1}, x_{B_1})$ is modified by the rule $rule_\iota$.
By Constraint 1 of Definition 8.2.9, we have:

$$(i', x_s) \rightsquigarrow^\star_{c_{j,\iota},G_a} (i_{B_1}, x_{B_1}).$$

By Lemma 8.2.8, it follows that:

$$(i_s, x_s) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\phi_\iota(i_{B_1}), x_{B_1}).$$

By the previous case, we know that:

$$(\phi_\iota(i_{B_1}), x_{B_1}) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

Thus, we can conclude that:

$$(i_s, x_s) \rightsquigarrow^\star_{\overline{c}_{\chi,\iota,F,(\mathscr{A},B)},G_a} (\varphi_F(i_t), x_t).$$

As a consequence, the site instance $(A_t, \varphi_F(i_t), x_t)$ is a target ifor the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$. That is to say that the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is a prefragment.

In all cases, the corrected concentration of the pattern component $\overline{c}_{\chi,\iota,F,(\mathscr{A},B)}$ is equal to 0, or to the corrected concentration of a prefragment.

8.3.3 Balance

We have provided, for any prefragment F, an explicit definition of the proper production $\delta^+_\star(F)$ of the prefragment F and of the proper consumption $\delta^-_\star F$ of the prefragment F, as a polynomial expression of the concentration of the other prefragments F'.

We consider the function $\mathbb{F}^\sharp$ that maps each abstract state $\rho^\sharp$ from $\mathcal{V}^\sharp$ to $\mathbb{R}$, to the value of the expression $\delta^+_\star(F) - \delta^-_\star(F)$. The function $\mathbb{F}^\sharp$ is well-defined.

Proposition 8.3.1. *We have* $\psi \circ \mathbb{F} = \mathbb{F}^\sharp \circ \psi$.

Proof. By construction of $\mathbb{F}^\sharp$. □

We can conclude with the soundness of our model reduction.

Theorem 8.3.2. *The tuple* $(\mathcal{V}, \mathbb{F}, \mathcal{V}^\sharp, \psi, \mathbb{F}^\sharp)$ *is an abstraction.*

8.4 Conclusion

In this Chapter, we have provided a context-sensitive abstraction of the flow of information between the sites of species. The level of resolution can be specified by the choice of a summary graph, which allows to distinguish less or more contexts. Each level of resolution provides its own fragmentation which is proved to be self-consistent.

Two canonic cases could be considered. If we take the summary graph as the contact map of the model. We get the model reduction that is proposed in Chapter 6. If we take the summary graph as the disjoint union of all the species, we get a framework that is similar to the one in [36] (additionnaly, our framework deals with side effects, and it has a better approximation of the flow of information both in the case of trivial rules and in the case of rules that may test that a site is bound, without specifying explicitly to which site).

When the summary graph is finite, our analysis of the flow of information always terminates and provides a finite symbolic representation of the variables of the reduced system. It is worth noting that in Kappa, deciding whether the set of species is finite or not, is undecidable [38].

Chapter 9

Conclusion

9.1 Contributions

In this thesis, we propose a general framework to reduce the differential semantics of biological models expressed in rule-based languages. We present two different analyses, that are performed at the level of rules. So we never need to execute the original model. Moreover we supply a general method to combine these model reductions. The results are proved to be correct by Abstract Interpretation.

The first analysis we propose is based on symmetries between sites. We formally define the notion of symmetries at the level of interaction sites, then we lift this definition to the level of the operational semantics. We show that two symmetric sites induce an equivalence relation over the species of the model and that this equivalence relation itself induces a bisimulation over the states of the system. This defines a model reduction. In fact, a system with symmetries is equivalent to a reduced system where the states are lumped according to their equivalence classes.

The second method we propose tracks the information flow between different regions of chemical species. Our analysis detects, for each site, which parts of the system influence its behaviour. This information is used to cut the molecular species in smaller pieces and to write a new system, by abstracting away the correlations that do not control the behaviour of any site. Our framework extends and improves the one presented in [25,28], where the species are cut just according to their type.

We introduce a hierarchy of representation both for species and the description of the flow of information among their sites. We show how every grain of resolution in the hierarchy can be used to obtain a self-consistent change of variables. We provide some criteria to ensure the soundness of the

annotation. These criteria concern the flow of information and the backward compatibility. Information flow captures the information that flows from sites that are tested to sites that are modified, whereas the backward compatibility ensures that, for every reaction, the abstraction of the reactants is at least as refined as the abstraction of its products. These criteria assure that we are able to express properly the consumption and the production of fragments in terms of other fragments. The context insensitive analyses computed from a contact map is just a particular case in the hierarchy. Specifically, the most abstract one. Interestingly, we notice that each annotated contact map already satisfies the backward compatibility criterion, thus there is not need to impose constraints for it. It could be interesting to investigate about other specific classes of representation that would enjoy other useful algebraic properties.

Our framework is a strict improvement of the one in [28], that has been successfully used to reduce a large section of the EGFR system. There, it is considered a set of 71 rules expanding into 18 051 984 143 555 729 567 species, reduced into 175 988 fragments.

Model reduction plays a role at several levels. It supplies a more compact representation, facilitating reasoning about inspected systems. Moreover, by changing the point of view, it assists in finding new understandings. On the computational side, it reconnects the system to the realm of feasible ODEs. Finally, it can be used as a starting point for other kind of anlyses: for example a reduction based on tropicalization [46, 47].

9.2 Future works

9.2.1 Dealing with cycles

In our framework, we have assumed that both species and fragments have no cycle.

Let us consider a small example to understand the difficulties which arise with species that exhibits cycles. We consider the set of species in Figure 9.1 and the following rule:

$$\mathtt{A} \to \varnothing.$$

We wonder the quantity of the pattern $B(x!\_, y!\_)$ that is consumed by the rule. In order to properly compute this quantity, we should consider both the patterns where just the site x is bound and the patterns where x and y are bound, paying attention to not overcount. This can be expressed as an alternated sum. The same kind of attention should be used when we want to compute the production of patterns.

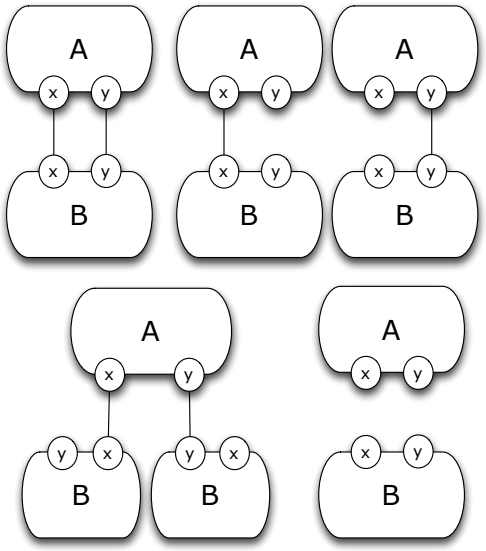


Figure 9.1: A set of species.

9.2.2 Tuning the context-sensitivity of the analysis

We showed that summary graphs can be used to tune the context sensitivity of the analysis. Our framework is generic and is correct whichever summary graph is chosen. Yet the choice of the summary graph is left as a parameter of our abstraction, and we provide no criterion to choose between all the possibilities.

One strategy would consist in computing a transition system for each kind of site to abstract each qualitative behaviour. Then we could zoom in the accuracy of the analysis by distinguishing contexts according to the states of the transition system.

The method we proposed allows the user to select any trade-off between context-insensitive and fully context-sensitive abstractions of the information flow. We plan to study methods to change the context during computation.

9.2.3 Contextual symmetries

In this thesis, we proposed a contextual analysis for the flow of information between the sites of species. We wonder if it possible to develop a contextual analyses also to discover sites that exhibit a symmetric behaviour just in a given context and if this can be used to reduce further a model.

To understand better the idea of sites that exhibit a symmetric behaviour in a given context we can have a look at Figure 9.2, that shows a protein A

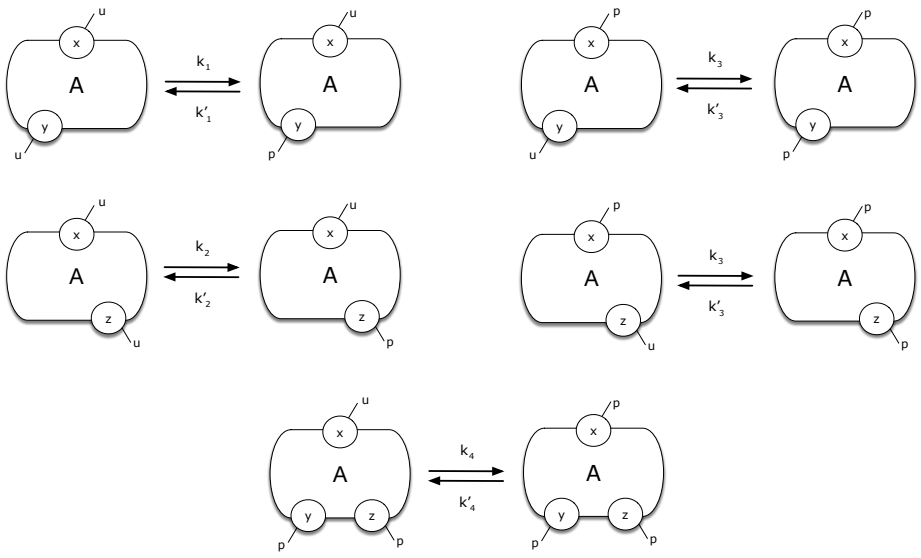


Figure 9.2: System showing contextual symmetries.

with three sites x, y and z. When the site x is phosphorylated, sites y and z have a symmetric behaviour.

Bibliography

[1] Martín Abadi. Secrecy by typing in security protocols. *J. ACM*, 46(5):749–786, September 1999.

[2] Adrien Basso-Blandin, Walter Fontana, and Russ Harmer. A knowledge representation meta-model for rule-based modelling of signalling networks. In César A. Muñoz and Jorge A. Pérez, editors, *Proceedings of the Eleventh International Workshop on Developments in Computational Models, DCM 2015, Cali, Colombia, October 28, 2015.*, volume 204 of *EPTCS*, pages 47–59, 2016.

[3] Andreea Beica, Calin C. Guet, and Tatjana Petrov. Efficient reduction of kappa models by static inspection of the rule-set. In Alessandro Abate and David Safránek, editors, *Hybrid Systems Biology - Fourth International Workshop, HSB 2015, Madrid, Spain, September 4-5, 2015. Revised Selected Papers*, volume 9271 of *Lecture Notes in Computer Science*, pages 173–191. Springer, 2015.

[4] Michael L. Blinov, James R. Faeder, Byron Goldstein, and William S. Hlavacek. Bionetgen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291, 2004.

[5] Nikolay M. Borisov, Alexander S. Chistopolsky, James R. Faeder, and Boris N. Kholodenko. Domain-oriented reduction of rule-based networks models. *IET Systems Biology*, 2(5), 2008.

[6] Nikolay M. Borisov, Nick I. Markevich, Jan B. Hoek, and Boris N. Kholodenko. Signaling through receptors and scaffolds: Independent interactions reduce combinatorial complexity. *Biophysical Journal*, 89, 2005.

[7] François Bourdoncle. Abstract interpretation by dynamic partitioning. *J. Funct. Program.*, 2(4):407–423, 1992.

[8] Peter Buchholz. Exact and ordinary lumpability in finite markov chains. *Journal of Applied Probability*, 31(1):59–75, 1994.

[9] Peter Buchholz. Bisimulation relations for weighted automata. *Theoretical Computer Science*, 393(1-3):109–123, 2008.

[10] Ferdinanda Camporesi and Jérôme Feret. Formal reduction of rule-based models. In *Postproceedings of the Twenty-seventh Conference on the Mathematical Foundations of Programming Semantics, MFPS XXVII*, volume 276C of *Electonic Notes in Theoretical Computer Science*, pages 31–61, Pittsburg, USA, 25–28 May 2011. Elsevier Science Publishers.

[11] Ferdinanda Camporesi, Jérôme Feret, Heinz Koeppl, and Tatjana Petrov. Combining model reductions. In Michael Mislove and Peter Selinger, editors, *Postproceedings of the Twenty-sixth Conference on the Mathematical Foundations of Programming Semantics, MFPS XXVI*, volume 265 of *Electonic Notes in Theoretical Computer Science*, pages 73–96, Ottawa, Cabada, 6–10 May 2010. Elsevier Science Publishers.

[12] Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Comparing chemical reaction networks:a categorical and algorithmic perspective. In *Proceedings of the Thirty-First Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2016.

[13] Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Symbolic computation of differential equivalences. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 137–150. ACM, 2016.

[14] Robert Cartwright and Mattias Felleisen. The semantics of program dependence. *SIGPLAN Not.*, 24(7):13–27, June 1989.

[15] H. Conzelmann, D. Fey, and E.D. Gilles. Exact model reduction of combinatorial reaction networks. *BMC Systems Biology*, 2:78–78, 8 2008.

[16] Holger Conzelmann. *Mathematical Modeling of Cellular Signal Transduction Pathways — A Domain-Oriented Approach to Reduce Combinatorial Complexity*. PhD thesis, Institut für Systemdynamik des Universität Stuttgart, 2008.

[17] Patrick Cousot. Abstract interpretation: Theory and practice. In Dragan Bošnački and Stefan Leue, editors, *Proceedings of the 9th International SPIN Workshop*, Grenoble, France, Lecture Notes in Computer

Science 2318, pages 2–5. Springer-Verlag, Berlin, Germany, 11–13 April 2002.

[18] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY.

[19] Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 269–282, San Antonio, Texas, 1979. ACM Press, New York, NY.

[20] Patrick Cousot and Radhia Cousot. Abstract interpretation and application to logic programs. *Journal of Logic Programming*, 13(2–3):103–179, 1992.

[21] Vincent Danos, Jérôme Feret, Walter Fontana, Russ Harmer, Jonathan Hayman, Jean Krivine, Christopher D. Thompson-Walsh, and Glynn Winskel. Graphs, rewriting and pathway reconstruction for rule-based models. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012*, volume 18 of *LIPIcs*, pages 276–288. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012.

[22] Vincent Danos, Jérôme Feret, Walter Fontana, Russ Harmer, and Jean Krivine. Rule-based modelling of cellular signalling, invited paper. In L. Caires and V.T. Vasconcelos, editors, *Proceedings of the Eighteenth International Conference on Concurrency Theory, CONCUR '2007, Lisbon, Portugal*, volume 4703 of *Lecture Notes in Computer Science*, pages 17–41, Lisbon, Portugal, 3–8 September 2007. Springer, Berlin, Germany.

[23] Vincent Danos, Jérôme Feret, Walter Fontana, Russ Harmer, and Jean Krivine. Rule-based modelling, symmetries, refinements. In Jasmin Fisher, editor, *Proceedings of the First International Workshop, Formal Methods in Systems Biology, FMSB '2008*, volume 5054 of *Lecture Notes in BioInformatics*, pages 103–122, Cambridge, UK, 4–5 June 2008. Springer, Berlin, Germany.

[24] Vincent Danos, Jérôme Feret, Walter Fontana, Russ Harmer, and Jean Krivine. Rule-based modelling and model perturbation. In Corrado Priami, Ralph-Johan Back, and Ion Petre, editors, *Transactions on Computational Systems Biology XI*, volume 5750 of *Lecture Notes in Computer Science*, pages 116–137. Springer Berlin Heidelberg, 2009.

[25] Vincent Danos, Jérôme Feret, Walter Fontana, Russ Harmer, and Jean Krivine. Abstracting the differential semantics of rule-based models: exact and automated model reduction. In Jean-Pierre Jouannaud, editor, *Proceedings of the Twenty-Fifth Annual IEEE Symposium on Logic in Computer Science, LICS '2010*, volume 0, pages 362–381, Edinburgh, UK, 11–14 July 2010. IEEE Computer Society.

[26] Vincent Danos, Jérôme Feret, Walter Fontana, and Jean Krivine. Scalable simulation of cellular signaling networks, invited paper. In Z. Shao, editor, *Proceedings of the Fifth Asian Symposium on Programming Systems, APLAS '2007, Singapore*, volume 4807 of *Lecture Notes in Computer Science*, pages 139–157, Singapore, 29 November – 1 December 2007. Springer, Berlin, Germany.

[27] Vincent Danos, Jérôme Feret, Walter Fontana, and Jean Krivine. Abstract interpretation of cellular signalling networks. In Francesco Logozzo, Doron A. Peled, and Lenore D. Zuck, editors, *Proceedings of the Ninth International Conference on Verification, Model Checking and Abstract Interpretation, VMCAI '2008*, volume 4905 of *Lecture Notes in Computer Science*, pages 83–97, San Francisco, USA, 7–9 January 2008. Springer, Berlin, Germany.

[28] Jérôme Feret, Vincent Danos, Jean Krivine, Russ Harmer, and Walter Fontana. Internal coarse-graining of molecular systems. *Proceedings of the National Academy of Sciences of the United States of America*, April 3 2009.

[29] Jerome Feret, Thomas A. Henzinger, Heinz Koeppl, and Tatjana Petrov. Lumpability abstractions of rule-based systems. *Theoretical Computer Science*, 431(0):137 – 164, 2012. Modelling and Analysis of Biological Systems Based on papers presented at the Workshop on Membrane Computing and Bio-logically Inspired Process Calculi (MeCBIC) held in 2008 (Iasi), 2009 (Bologna) and 2010 (Jena).

[30] Jérôme Feret, Heinz Koeppl, and Tatjana Petrov. Stochastic fragments: A framework for the exact reduction of the stochastic semantics of

rule-based models. *International Journal of Software and Informatics*, 7(4):527 – 604, 2013.

[31] Jérôme Feret and Kim Quyên Lý. Reachability analysis via orthogonal sets of patterns. In *Seventeenth International Workshop on Static Analysis and Systems Biology (SASB'16)*, ENTCS. elsevier, 2017. to appear.

[32] Steven Gay, François Fages, Thierry Martinez, Sylvain Soliman, and Christine Solnon. On the subgraph epimorphism problem. *Discrete Applied Mathematics*, 162:214–228, 2014.

[33] Roberto Giacobazzi and Isabella Mastroeni. Non-standard semantics for program slicing. *Higher-Order and Symbolic Computation*, 16(4):297–339, 2003.

[34] Alexander N. Gorban and Ovidiu Radulescu. Dynamic and static limitation in multiscale reaction networks, revisited. In G.B. Marin, D. West, and G.S. Yablonsky, editors, *Advances in Chemical Engineering: Mathematics in Chemical Kinetics and Engineering*, volume 34 of *Advances in Chemical Engineering*, pages 103 – 173. Academic Press, 2008.

[35] Russ Harmer. Rule-based modelling and tunable resolution. In Barry S. Cooper and Vincent Danos, editors, *Proceedings Fifth Workshop on Developments in Computational Models–Computational Models From Nature, DCM 2009, Rhodes, Greece, 11th July 2009.*, volume 9 of *EPTCS*, pages 65–72, 2009.

[36] Russ Harmer, Vincent Danos, Jérôme Feret, Jean Krivine, and Walter Fontana. Intrinsic information carriers in combinatorial dynamical systems. *Chaos*, 20, September 2010.

[37] Edward L. Ince. *Ordinary Differential Equations*. Dover Publications, 1956.

[38] Peter Kreyßig. Chemical organisation theory beyond classical models: Discrete dynamics and rule-based models.

[39] Thomas G. Kurtz. Solutions of ordinary differential equations as limits of pure jump markov processes. *Journal of Applied Probability*, 7(1):49–58, 1970.

[40] Thomas G. Kurtz. Limit theorems for sequences of jump markov processes approximating ordinary differential processes. *Journal of Applied Probability*, 8(2):344–356, 1971.

[41] Guillaume Madelaine, Cédric Lhoussaine, and Joachim Niehren. Structural simplification of chemical reaction networks preserving deterministic semantics. In Olivier F. Roux and Jérémie Bourdon, editors, *Computational Methods in Systems Biology - 13th International Conference, CMSB 2015, Nantes, France, September 16-18, 2015, Proceedings*, volume 9308 of *Lecture Notes in Computer Science*, pages 133–144. Springer, 2015.

[42] Guillaume Madelaine, Elisa Tonello, Cédric Lhoussaine, and Joachim Niehren. Normalizing chemical reaction networks by confluent structural simplification. In Ezio Bartocci, Pietro Liò, and Nicola Paoletti, editors, *Computational Methods in Systems Biology - 14th International Conference, CMSB 2016, Cambridge, UK, September 21-23, 2016, Proceedings*, volume 9859, pages 201–215. Springer, 2016.

[43] Robin Milner. *Communication and Concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[44] Tatjana Petrov and Heinz Koeppl. Maximal Reduction of ODE Semantics of Rule-based Models: Syntax-Independent Setup. In *International Workshop on Computational Systems Biology*, 2010.

[45] Ovidiu Radulescu, Alexander N. Gorban, Andrei Zinovyev, and Vincent Noël. Reduction of dynamical biochemical reactions networks in computational biology. *Frontiers in Genetics*, 3(131):., 2012.

[46] Ovidiu Radulescu, Sergei Vakulenko, and Dima Grigoriev. Model reduction of biochemical reactions networks by tropical analysis methods. *Mathematical Modelling of Natural Phenomena*, 10(3):124–138, 2015.

[47] Jasha Sommer-Simpson, John Reinitz, Leonid Fridlyand, Louis Philipson, and Ovidiu Radulescu. Hybrid reductions of computational models of ion channels coupled to cellular biochemistry. In Ezio Bartocci, Pietro Liò, and Nicola Paoletti, editors, *Computational Methods in Systems Biology - 14th International Conference, CMSB 2016, Cambridge, UK, September 21-23, 2016, Proceedings*, volume 9859 of *Lecture Notes in Computer Science*, pages 273–288. Springer, 2016.

Résumé

Le comportement d'une cellule dépend de sa capacité à recevoir, propager et intégrer des signaux, constituant ainsi des voies de signalisations. Les protéines s'associent entre elles sur des sites de liaisons, puis modifient leur structure spatiale, ce qui put cacher ou révéler r leurs autres sites de liaisons.

En raison, du grand nombre de différents complexes bio-moléculaires, nous ne pouvons pas écrire ou générer les systèmes d'ODEs sous-jacents. Les langages de réécritures de graphes à sites offrent un bon moyen de décrire ces systèmes. Néanmoins la complexité combinatoire resurgit lorsque l'on veut calculer le comportement de ces systèmes, ce qui justifie l'utilisation d'abstractions.

Nous proposons deux méthodes pour réduire la taille de ces modèles. Elles utilisent respectivement la présence de symétries parmi certains sites et le manque de corrélation entre différentes parties du système. Des sites qui ont la même capacité d'interaction sont liés par une relation de symétrie, qui induit une bisimulation. L'analyse du flot d'information détecte les parties du système qui influencent le comportement de chaque site. Ceci nous autorise à couper les espèces moléculaires en petits morceaux pour écrire un nouveau système. Enfin, nous montrons comment raffiner cette analyse pour tenir compte d'information contextuelle.

Les deux méthodes peuvent être combinées. La solution analytique du modèle réduit est la projection exacte de la solution originelle. Le calcul du modèle réduit se fait au niveau des règles, en évitant l'exécution du modèle initial.

Mots Clés

Voies de signalisation, interprétation abstraite, modélisation par règles de réécritures, réduction de modèles, bisimulations, flot d'information.

Abstract

The behaviour of a cell is driven by its capability to receive, propagate and communicate signals. Proteins can bind together on some binding sites. Post-translational modifications can reveal or hide some sites, so new interactions can be allowed or existing ones can be inhibited.

Due to the huge number of different bio-molecular complexes, we can no longer derive or integrate ODE models. A compact way to describe these systems is supplied by rule-based languages. However combinatorial complexity raises again when one attempt to describe formally the behaviour of the models. This motivates the use of abstractions.

We propose two methods to reduce the size of the models, that exploit respectively the presence of symmetries between sites and the lack of correlation between different parts of the system. Symmetries relates pairs of sites having the same capability of interactions. We show that this induces a bisimulation which can be used to reduce the size of the original model. The information flow analysis detects, for each site, which parts of the system influence its behaviour. This allows us to cut the molecular species in smaller pieces and to write a new system. Moreover we show how to tune the context sensitivity of this analysis.

Both approaches can be combined. The analytical solution of the reduced model is the exact projection of the original one. The computation of the reduced model is performed at the level of rules, without the need of executing the original model.

Keywords

Signaling pathways, abstract interpretation, rule-based modeling, model reduction, bisimulations, information flow.