Subgraph Epimorphisms: Theory and Application to Model Reductions in Systems Biology

Steven Gay

UNIVERSITE PARIS DIDEROT (Paris VII)

École Doctorale 386: Sciences Mathématiques de Paris Centre

Thèse de Doctorat

pour l'obtention du grade de

Docteur en Sciences de l'Université Paris VII

Spécialité Informatique

Steven GAY

Subgraph Epimorphisms: Theory and Application to Model Reductions in Systems Biology

Directeur de thèse : **François FAGES**
Co-directeur de thèse : **Sylvain SOLIMAN**

Travaux réalisés à Inria Paris-Rocquencourt

Présentée et soutenue publiquement le 26 Mai 2015 devant le jury composé de :

M. HABIB Michel,	Président
M. DANOS Vincent,	Rapporteur
M. REGIN Jean-Charles,	Rapporteur
M. PRADALIER Sylvain,	Examinateur
Mme SOLNON Christine,	Examinatrice
M. FAGES François,	Directeur de thèse
M. SOLIMAN Sylvain,	Codirecteur de thèse

Résumé

Cette thèse développe une méthode de morphismes de graphes et l'applique à la réduction de modèles en biologie des systèmes. Nous nous intéressons au problème suivant: l'ensemble des modèles en biologie des systèmes est en expansion, mais aucune relation formelle entre les modèles de cet ensemble n'a été entreprise. Ainsi, la tâche d'organisation des modèles existants, qui est essentielle pour le raffinement et le couplage de modèles, doit être effectuée par le modélisateur. En biomathématiques, les techniques de réduction de modèle sont étudiées depuis longtemps, mais ces techniques sont bien trop restrictives pour être appliquées aux échelles requises en biologie des systèmes.

Nous proposons un cadre de réduction de modèle, basé uniquement sur des graphes, qui permet d'organiser les modèles en un ordre partiel. Les modèles de biologie des systèmes seront représentés par leur *graphe de réaction*. Pour capturer le processus de réduction lui-même, nous étudierons un type particulier de morphismes de graphes: les *épimorphismes de sous-graphe*, qui permettent la fusion et l'effacement de sommets. Nous commencerons en analysant l'ordre partiel qui émerge des opérations de fusion et d'effacement, puis nous développerons des outils théoriques pour résoudre les problèmes calculatoires de notre méthode, et pour finir nous montrerons la faisabilité de l'approche et la précision du cadre "graphes de réactions/épimorphismes de sous-graphe", en utilisant un dépôt de modèles de biologie des systèmes.

Abstract

This thesis develops a framework of graph morphisms and applies it to model reduction in systems biology. We are interested in the following problem: the collection of systems biology models is growing, but there is no formal relation between models in this collection. Thus, the task of organizing the existing models, essential for model refinement and coupling, is left to the modeler. In mathematical biology, model reduction techniques have been studied for a long time, however these techniques are far too restrictive to be applied on the scales required by systems biology.

We propose a model reduction framework based solely on graphs, allowing to organize models in a *partial order*. Systems biology models will be represented by their *reaction graphs*. To capture the process of reduction itself, we study a particular kind of graph morphisms: *subgraph epimorphisms*, which allow both vertex merging and deletion. We first analyze the partial order emerging from the merge/delete graph operations, then develop tools to solve computational problems raised by this framework, and finally show both the computational feasibility of the approach and the accuracy of the reaction graphs/subgraph epimorphisms framework on a large repository of systems biology models.

Remerciements

J'aimerais tout d'abord remercier mes parents de m'avoir soutenu et supporté durant mes années de thèse. J'espère que je serai à la hauteur de l'énergie et de la patience dont ils ont fait preuve à mon égard.

Ensuite, je voudrais remercier François Fages de m'avoir donner la possibilité de faire cette thèse. François m'a transmis son ambition à mener des projets complexes, et sa volonté créative a été contagieuse.

Sylvain Soliman a été le pendant réaliste de ma direction, et m'a par exemple souvent aidé à me sortir des affres de Prolog. Cette thèse lui doit aussi les fusions de réaction, je n'oublie pas ! Merci finalement pour l'exemple de rigueur mais aussi pour m'avoir trainé à la salsa ...

Gregory Batt a su, malgré notre éloignement thématique, partager une sagesse bienvenue. Merci pour les conseils de présentation avisés !

Merci aussi à Thierry Martinez d'avoir été une oreille attentive à mes problèmes. Sa compétence universelle a aussi été une inspiration, et il m'a beaucoup appris lors de nos quotidiennes discussions autour du Red Bull du midi (ou des deux cafés pour Thierry).

J'ai trouvé en David Fournier un excellent camarade, avec qui j'ai partagé des bons moments musicaux (bien qu'un peu répétitifs, je ne savais pas que des gens utilisaient le repeat track avant de connaître David), et trop de délires, sûrement. L'enthousiasme de David pour les nouvelles approches m'a sans doute encouragé à simplement essayer, merci pour l'inspiration !

Inria (ou sous son ancien nom, l'INRIA) met beaucoup en oeuvre pour accompagner ses doctorants, et son personnel a toujours su m'aider dans les démarches administratives. L'INRIA a aussi su être un lieu de partage, avec ses danseurs de salsa, ses omnisportifs et ses colloques. Le philosophe lui donnerait un seul défaut : c'est loin (mais c'est beau).

Je remercie le Kenyū de m'avoir accueilli et de m'avoir transmis les valeurs du kendō, j'aurais finalement autant appris l'humilité en keikogi qu'à me battre contre des problèmes NP-difficiles ! La persévérance des anciens pendant les entrainements à transmettre inlassablement a eu beaucoup d'impact sur moi, c'est sûrement le comportement qui m'a le plus impressionné et transformé. Quant à toujours finir ce qu'on a commencé, il va me falloir encore quelques entraînements.

Merci à Pierre Schaus de m'avoir donné la possibilité de commencer un post-doc sans avoir de doctorat, j'espère ne pas l'avoir trompé sur la marchandise.

Les hommes construisent des maisons, les femmes en font des foyers, dit-on. Alors, merci à la petite abeille de m'avoir accueilli dans sa ruche. Grâce à la danse sans huit, j'ai trouvé quelqu'un qui navigue sur ma longueur d'onde, puisse la musique résonner longtemps encore !

Contents

Chapter 1

Introduction

Model Reduction in Systems Biology. As most fields in science, molecular cell biology uses models. Models allow the compact storage of accumulated knowledge and the prediction of future events.

One of the main concerns of molecular biology is the interaction between molecules, and since there are many different molecules in nature, we can either regroup this large quantity of information in big models that contain the many interactions the molecules can have, or in many models for every interaction of interest.

Most often, we are interested in the effect of a molecule not only on another molecule in a cell, but on the cell as a whole: does it grow and divide? Does it die? Does it produce some molecule of interest? This field of biology where the cell is treated as a whole is known as *systems biology*: there, models often try to capture the effect of a molecule on some subsytem of the cell, e.g. through signalling processes. Such models can have many agents in interaction.

However, in order to make a model of the action of a new molecule on a system, using exhaustive models that reflect every known interaction is impractical. If we want quantitative results on our interaction of interest, we need quantitative data on the whole model. However, molecular interactions are often known only at a qualitative level, and even if some quantitative data is known, it will rarely be transferrable to another biological setup.

The models we want to work on are simplified, *reduced* models that only contain what is necessary, while still showing the global behaviours we want to capture. Finding kinetic parameters for model calibration is easier on smaller models, since the search space is smaller.

Thus, models are typically divided into two categories: knowledge maps, of qualitative nature, that contain all interactions known for some subsystem of the cell, which allow qualitative prediction; and working models, of quantitative nature, that are more frugal but allow quantitative predictions.

Classification of Models by Model Reduction. When we have worked on a model M and obtained a model M', how can we relate M' to the mass of published models, collected in repositories such as BioModels [43]? Most results on model reduction only answer to the problem of preserving some behaviour of M and when simplifying to M'.

However, other questions are of interest: Is M' also a reduced version of another existing model M_0, simpler than M? Is M' unexpectedly an extension of a very simple model M'', for which data has been published and could be reused for M'? Maybe M' cannot be related by reduction to any existing model: how far is it from other models? Harder questions still: given two models M_1 and M_2, can we extract a model of what is common to M_1 and M_2? Can we build a model that factorizes what is in common to M_1 and M_2, but also keeps the specificities of both models?

Abstraction to Reaction Graphs. In order to answer these questions, our approach is to abstract models to *reaction graphs*, and capture typical model reductions by *graph rewriting*. Let us consider a classical reduction, the Michaelis-Menten reduction.

The Michaelis-Menten kinetics has been published in 1913, and is a standard kinetics that can be found in many models today. It is a reduction of a catalytic mechanism with 3 reactions and 4 molecular species, and Mass Action Law kinetics:

$$E + M \underset{k_d}{\overset{k_c}{\rightleftarrows}} F \to_{k_p} E + P$$

with associated system of ODEs

$$\begin{cases} \dot{M} &= -k_c \cdot E \cdot M + k_d \cdot F \\ \dot{E} &= -k_c \cdot E \cdot M + k_d \cdot F + k_p \cdot F \\ \dot{F} &= k_c \cdot E \cdot M - k_d \cdot F - k_p \cdot F \\ \dot{P} &= k_p \cdot F \end{cases}$$

into a single reaction with 3 species and a more complex kinetics:

$$E + M \to_{[\frac{k_m \cdot C \cdot M}{K_m + M}]} E + P$$

with associated ODEs

$$\begin{cases} \dot{M} &= -\frac{k_m \cdot C \cdot M}{K_m + M} \\ \dot{E} &= 0 \\ \dot{P} &= \frac{k_m \cdot C \cdot M}{K_m + M} \end{cases}$$

This reduction is deemed valid in some regions of the concentration space, as is shown for one set of parameters in Fig. 1.1, and there are several ways to derive it. However, the study of the specifics is not the focus of this work.

Figure 1.1: An enzymatic mechanism compared to the Michaelis-Menten reduced version. Under some conditions, the Michaelis-Menten version displays the same behaviour as the full mechanism, while using only one reaction instead of three.

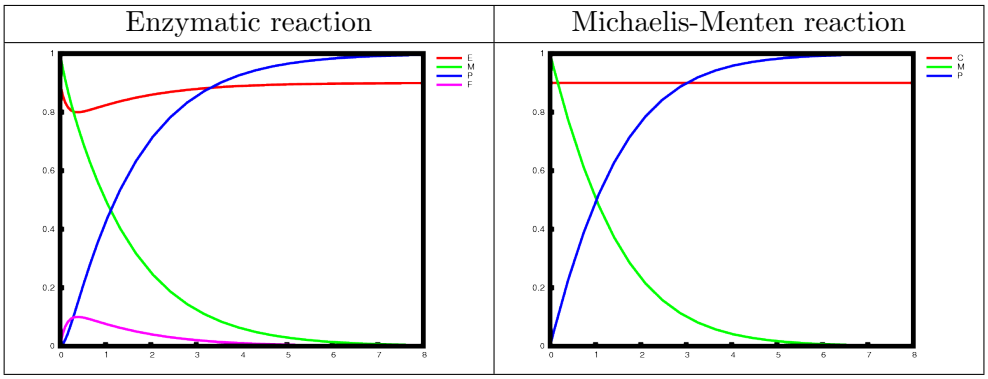
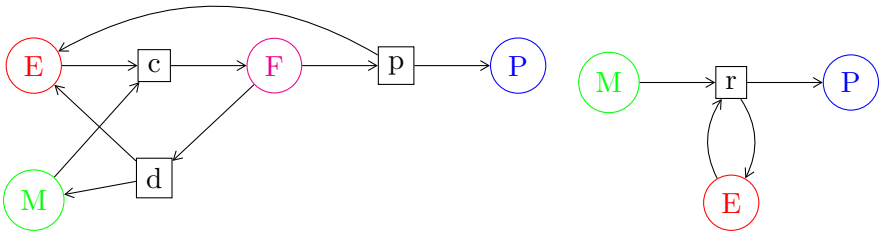


Figure 1.2: On the left, the catalytic mechanism from Fig.1.1, with its 3 reactions and 4 species. On the right, the Michaelis-Menten reduced version, with 1 reaction and 3 species.



The focus is the action of the mathematical reduction on the structure of a model. Keeping Michaelis-Menten as our running example, consider Fig.1.2: these are the graphs of the extended and reduced catalytic mechanisms.

A set of chemical reactions can be represented as bipartite graphs, where circle vertices are chemical species, and square vertices chemical reactions. What graph operations can transform the full mechanism into the reduced one?

The whole process could be considered as one graph transformation that we could apply locally in realistic systems biology models, but generalizing this to every reduction technique would lead to accumulating graph transformations, so that the theory at the graph abstraction level would be too complicated for computation purposes, and also too unstable mathematically, i.e. adding one kind of reduction could make mathematical properties of the framework turn from true to false. We need a simple yet robust foundation, on which computation is feasible, yet accurate enough to capture mathematical model reductions.

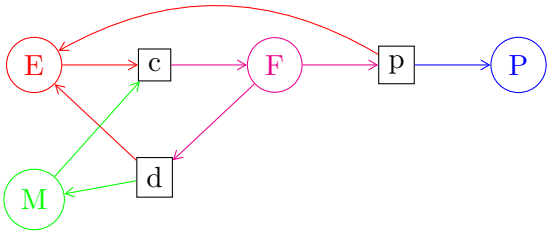
A natural graph operation in this case is *vertex deletion*, since removing a new molecule and its associated reactions from a graph should yield a system where the molecule is considered absent. This operation is well-studied, since asking whether M' can be reached from M by vertex deletions is equivalent to asking whether there is a subgraph isomorphism (SISO) from M to M', and the SISO problem is a classic NP-complete problem [16]. Yet SISO is not satisfying, since it does not capture the omnipresent graph transformation underlying the Michaelis-Menten reduction in Fig.1.2. However, we observe that adding *vertex merging* is enough, as shown in Fig.1.3.

This thesis is a study on using vertex deletion and vertex merging as a relaxation of biochemical model reductions that is still a good approximation. Given M and M', we can answer to the question of reducibility by using this framework. If M' is not reachable from M by using deletion and merging, then there is no way to reduce M to M' by any combination of the mathematical reductions captured by the framework. When M is a knowledge map and M' is a model of some new interaction, this means that the new interaction cannot be seen as a consequence of some model reduction, it is genuinely new. If M' is reachable from M, then this does not systematically mean that M can really be reduced to M' in some valid way, however it can give some insight as of how much a model is similar to another, and the framework is unexpectedly accurate in this use, as shown in Chapter 10.

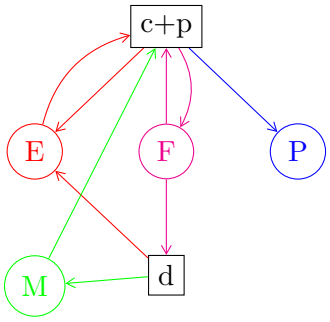
By systematically applying comparisons between all models to repositories such as BioModels, that contain hundreds of models with no given relation, the framework can organize existing models as hierarchies of refined to reduced, without having to add more information to the models.

Outline. The plan of this thesis is the following. Since using delete/merge operations was not considered for our purpose before, we will first outline the theory behind this particular kind of graph transformations. In **Chapter 2**, we will define our graph transformations and identify properties that are useful for computation purposes: specifically, the fact

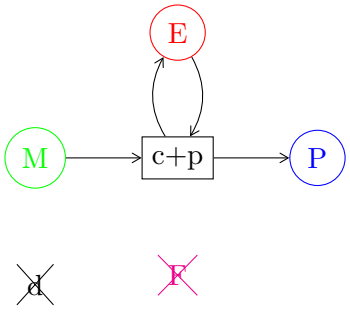
Figure 1.3: Model reduction can be approximated as deletion/merging graph operations: e.g. constant concentrations as species deletion, proportional concentrations as species merging, zero rate as reaction deletion, proportional rates as reaction merging.



Merge c with p



Delete d, Delete F



that delete/merge strings can be seen as a kind of graph morphisms, namely subgraph epimorphisms, or SEPI for short. In **Chapter 3**, we explore further notions associated to the partial order generated by our operations: the SEPI greatest lower bounds correspond to maximal common subgraphs, which can be a way to see structure shared between models by way of model reduction, the SEPI least upper bounds correspond to minimal common supergraphs, which can be seen as a way to couple models so that the original models are reduced versions of the coupled one, and the SEPI distance between two graphs.

Then we will see that despite the NP-completeness of the SEPI decision problem, even in the case of our bipartite graph application, as shown in **Chapter 4**, computing a SEPI is still feasible in practice. **Chapter 5** presents a way to decide the existence of a SEPI using constraint programming. **Chapter 6** presents a way to decide the existence of a SEPI and to compute SEPI glb and lub using a SAT-solving approach.

Finally we will return to our motivating application. **Chapter 7** shows one major mathematical framework in systems biology, ordinary differential equations, the natural structure underlying it, reaction models, and the abstraction we use for this thesis, reaction graphs. It will also show how graph operations and model reductions are linked. **Chapter 8** will show how to import real-life models from the BioModels repository in our framework: indeed, the extraction of a reaction graph from an SBML file is not always straightforward. **Chapter 9** is a benchmark of the previous methods on reaction graphs extracted from systems biology models. **Chapter 10** then evaluates the framework on the BioModels repository and shows that our framework yields meaningful results.

The conclusion will recapitulate the limits of the current approach, and discuss how to improve it, both from the computational and biological points of view.

Part I

A Theory of Subgraph Epimorphisms

Chapter 2

Graphs, Reduction Operations and Morphisms

A directed graph, or graph for short in this section, is a pair (V, A) such that V is a finite set of vertices and $A \subseteq V \times V$ is a set of arcs. Defined this way, a graph may have loops, and cannot have more than one arc from some vertex to another vertex. The cardinality of a set S is denoted $|S|$. The size of a graph $G = (V, A)$, denoted $|G|$, is $|G| = |V|$. When not explicitly defined, G and G' denote graphs, with $G = (V, A)$ and $G' = (V', A')$.

2.1 Graphs, Merge and Delete, Pointed Graphs

2.1.1 *merge* operation

The *merge* operation intuitively makes a new vertex using two vertices in the source graph: the new vertex inherits the arcs of the two vertices, and these vertices are then removed.

Definition 2.1 (Merge)**.** *For all $v, w \in V$ the merge operation $m_{v,w}$ on v and w yields the graph $m_{v,w}(G) = (V', A')$ where*

$$\begin{aligned} V' &= V \setminus \{u, v\} \uplus \{uv\} \\ A' &= A \cap (V' \times V') \\ &\quad \cup \{(uv, x) \mid (u, x) \in A \text{ or } (v, x) \in A\} \\ &\quad \cup \{(x, uv) \mid (x, u) \in A \text{ or } (x, v) \in A\} \\ &\quad \cup \{(uv, uv) \mid (u, v) \in A \text{ or } (v, u) \in A\} \end{aligned}$$

and uv is a fresh symbol.

We write $G \to_m G'$ whenever $\exists u, v, G' = m_{u,v}(G)$, and $G \to_m^* G'$ whenever a graph can be obtained using zero, one or several merges, i.e.,

$$G \to_m G_1 \to_m \ldots \to_m G_n \ldots \to_m G'$$

Example 2.2. *Fig. 2.1 shows how merging is applied on some graphs, Fig. 2.2 shows graphs that cannot be related with a merge.*

Different merging strings may yield the same result independently of the graph itself. Indeed, since this operation is local, mergings of distinct pairs of vertices do not affect one another. Since the operation is basically an union of vertices and their arcs, they enjoy, as set union, a commutative-associative-like property:

Proposition 2.3. *Let u, v, w, x be distinct vertices of G. Then*

- $m_{u,v}(G) = m_{v,u}(G)$
- $m_{u,v}(m_{w,x}(G)) = m_{w,x}(m_{u,v}(G))$
- $m_{u,vw}(m_{v,w}(G)) = m_{v,uw}(m_{u,w}(G)) = m_{w,uv}(m_{u,v}(G))$

Example 2.4. *Fig. 2.3 illustrates the associative-commutative-like property of merging on a small graph.*

2.1.2 *delete* operation

The *delete* operation removes a vertex from a graph and every arc connected to it.

Definition 2.5 (Delete)**.** *Let $v \in V$. The result of the deletion of v in G is the graph $d_v(G)=(V', A')$ where*

$$V' = V \setminus \{v\}$$
$$A' = A \cap (V' \times V')$$

We write $G \to_d G'$ whenever $\exists u, G' = d_u(G)$, and $G \to_d^* G'$ whenever a graph can be obtained using zero, one or several deletes, i.e.,

$$G \to_d G_1 \to_d \ldots \to_d G_n \ldots \to_d G'$$

Example 2.6. *Fig. 2.4 shows how delete operations applied on some graphs, Fig. 2.5 shows graphs that cannot be related with one deletion.*

As mergings, deletions have a local action, so that they enjoy commutation properties.

Proposition 2.7. *Vertex deletions commute: $\forall u \neq v, d_u(d_v(G)) = d_v(d_u(G))$.*

Figure 2.1: Four examples of merging. From top to bottom: a disconnected graph gets connected, merging cannot make a double arc appear, merging can make a pair of inverse arcs appear, a loop gets created when connected vertices are merged.

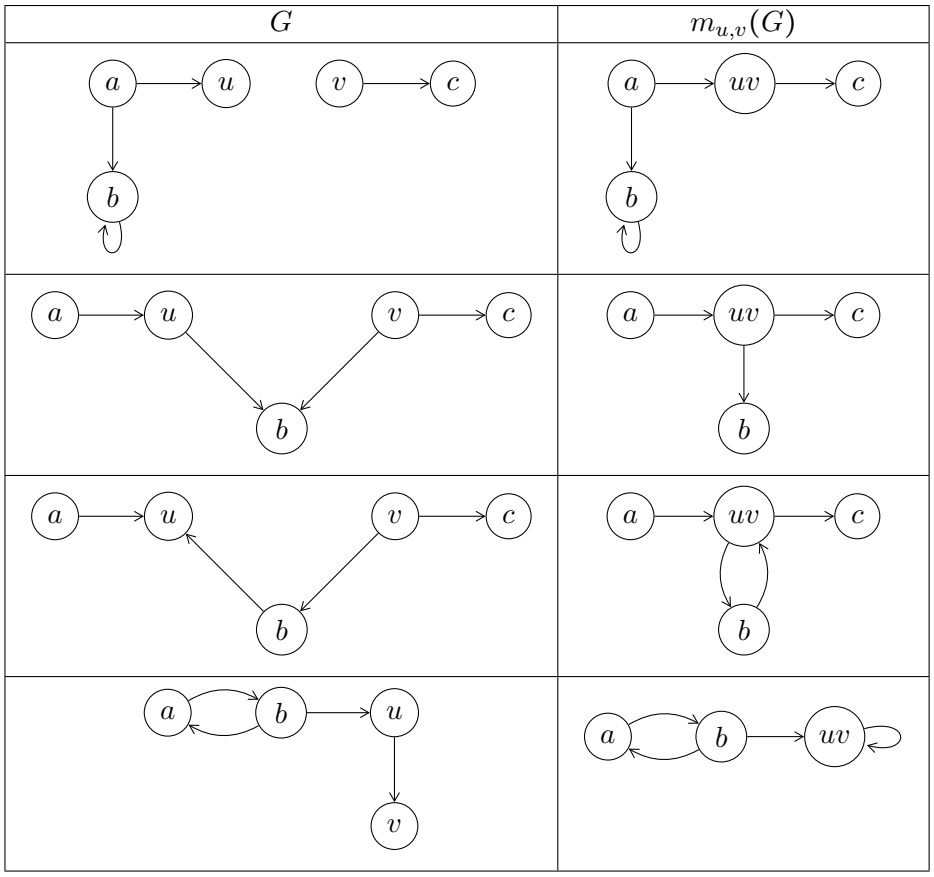


Figure 2.2: Four G, G' pairs such that $G \longrightarrow_m G'$ does not hold. From top to bottom, G' has too many vertices; G' has too few vertices; every merging but $m_{a,c}$ and $m_{b,d}$ create loops, but these two do not create inverse arcs; finally the image of the 3-circuit *bed* should be either a 3-circuit or a 2-circuit with a loop, but there is neither in G'.

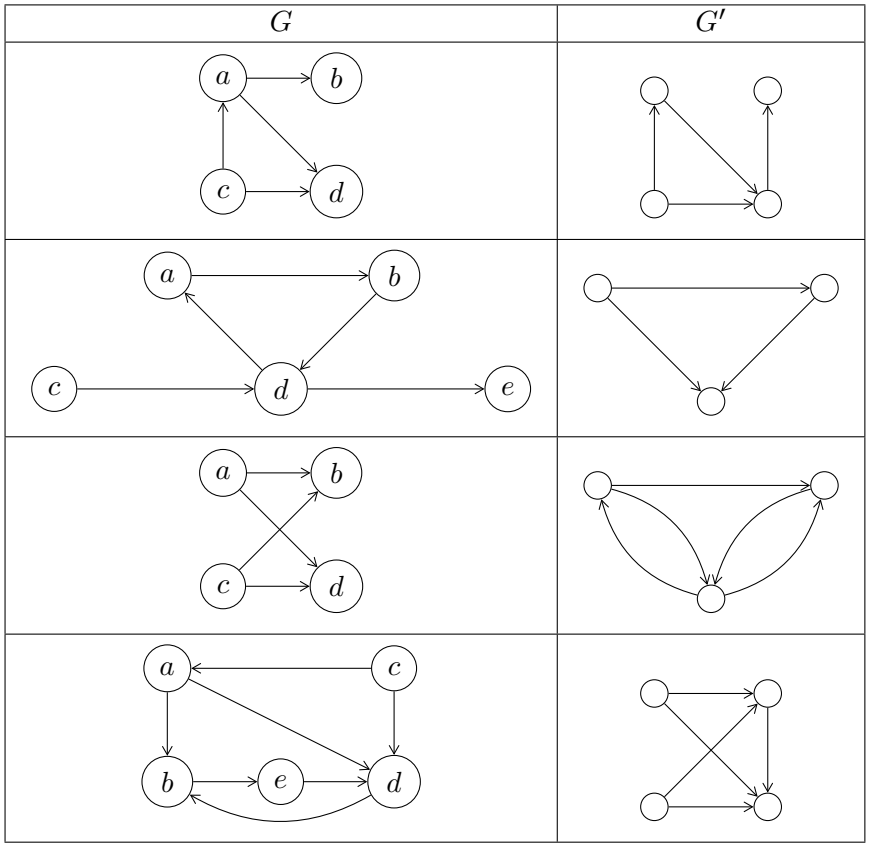


Figure 2.3: An illustration of the commutation property of Proposition 2.3. When merging a set of vertices, the end result does not depend on the order of the mergings. Here merging the set $\{u, v, w\}$ in the top graph yields the bottom graph, whatever the order of application.

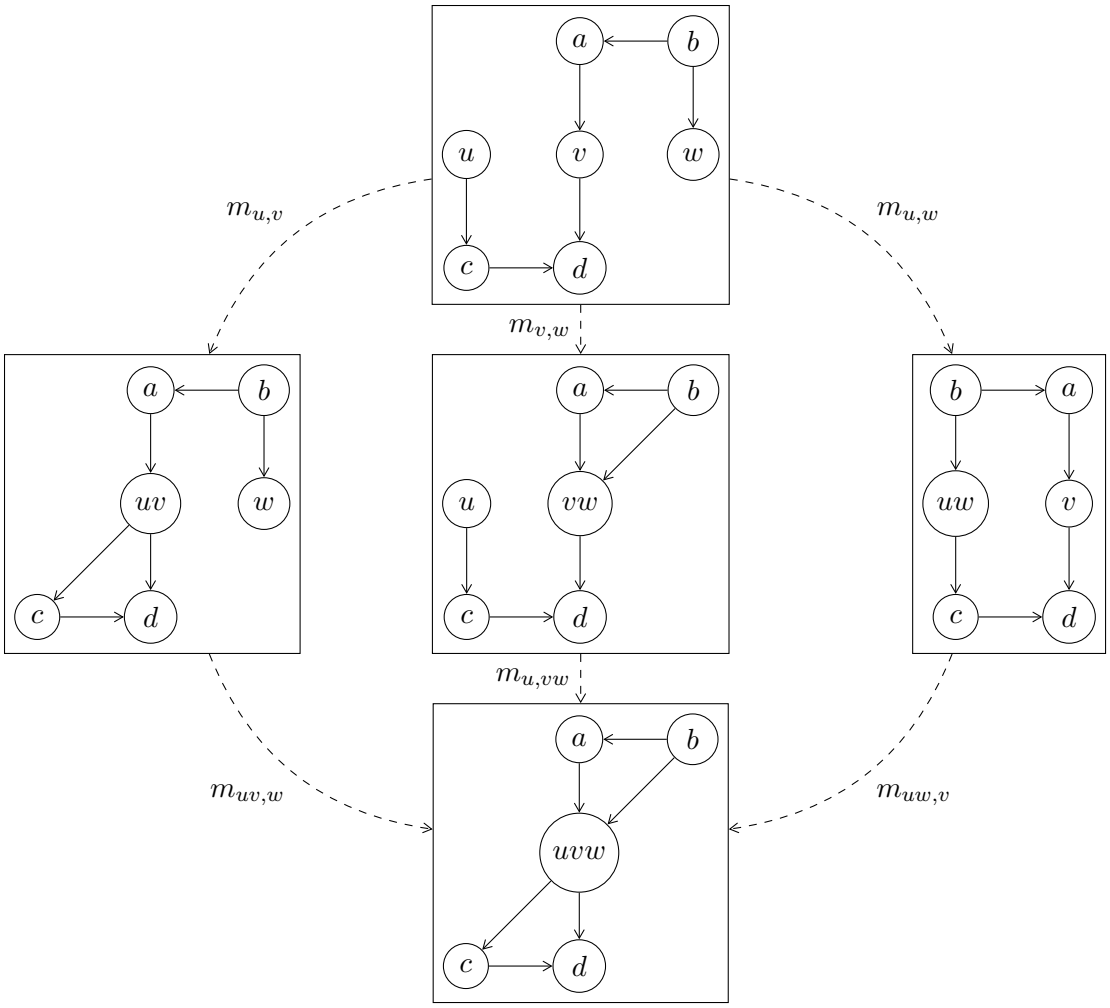


Figure 2.4: Two examples of vertex deletion. Vertex deletion decreases the number of vertices and arcs, more specifically the number of incoming and outcoming arcs for each vertex.

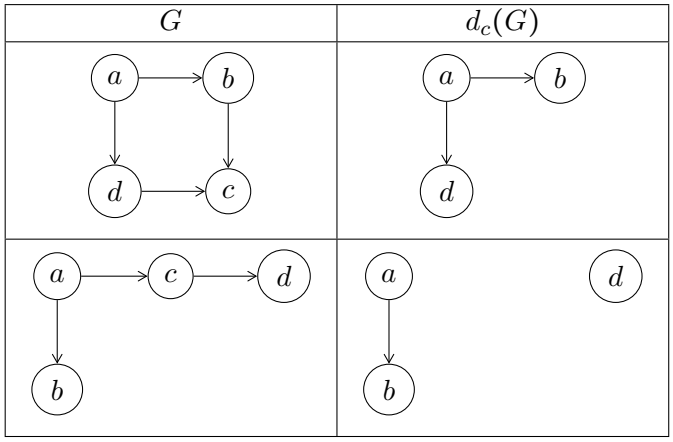
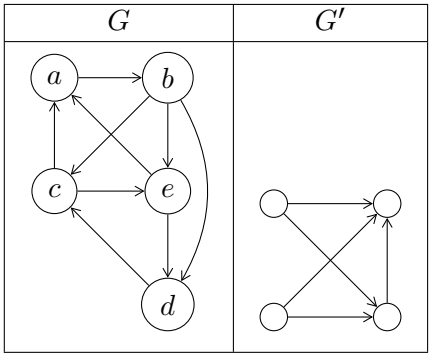


Figure 2.5: An example where $G \longrightarrow_d G'$ does not hold. G' has a vertex with 3 incoming arcs, but G has no vertex with more than 2 incoming arcs Since degree decreases through vertex deletion, $G \longrightarrow_d G'$ cannot hold.



2.1.3 Combining merge and delete operations

We write $G \to_{md} G'$ whenever $G \to_m G'$ or $G \to_d G'$, and $G \to^*_{md} G'$ whenever a graph can be obtained using merges and/or deletes, i.e.,

$$G \to_{md} G_1 \to_{md} \ldots \to_{md} G'$$

Proposition 2.8. *Some identites emerge from mixing merge and delete operations:*

- $d_u(m_{v,w}(G)) = m_{v,w}(d_u(G))$
- $d_{uv}(m_{u,v}(G)) = d_v(d_u(G))$

Example 2.9. *Fig. 2.6 shows additional symmetries that emerge when mixing merge and delete operations.*

Merging and deleting do not simulate one another directly, in the sense that $\exists G', G \to_m G' \wedge G \not\to'_d G$ and $\exists G', G \to_d G' \wedge G \not\to'_m G$. This is shown in Fig. 2.7.

However, using a simple encoding allows the simulation of deletes by merges:

Definition 2.10 (Pointed Graphs)**.** *Let $\bot$ be a fresh symbol.*

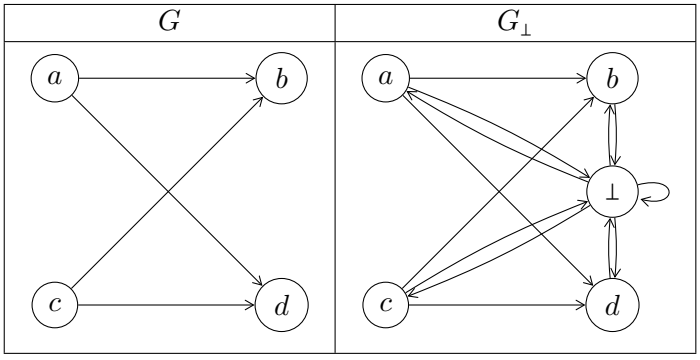
Let $\cdot_\bot : G \in \mathcal{G} \longrightarrow G_\bot = (V_\bot, A_\bot)$, where $V_\bot = V \uplus \bot$ and $A_\bot = A \uplus (V \times \{\bot\}) \uplus (\{\bot\} \times V) \uplus \{(\bot, \bot)\}$.

We call dummy vertex *of a graph G one that has all possible arcs to/from the other vertices of G and to itself.*

In $G_\bot$, $\bot$ is always a dummy vertex, we call $G_\bot$ the pointed graph *of G.*

We write the set of pointed graphs $\mathcal{G}_\bot = \{G_\bot \mid G \in \mathcal{G}\}$. We extend the merge operation on $\mathcal{G}_\bot$ with no special treatment for $\bot$: a priori, the image of $\bot$ can be any vertex, and the antecedents of $\bot$ can be any vertex.

Example 2.11. *On the left, a graph G with 4 nodes. On the right, the graph $G_\bot$.*



Proposition 2.12. *$G \to^*_d G'$ iff $G_\bot \to_m G'_\bot$.*

Proof. By coding deletion as a merge with $\bot$. □

Corollary 2.13. *$G \to^*_{md} G'$ iff $G_\bot \to^*_m G'_\bot$.*

Figure 2.6: Symmetries appear when using both merge and delete operations on the top graph: deleting u and deleting v amounts to merging u with v and deleting the resulting vertex uv.

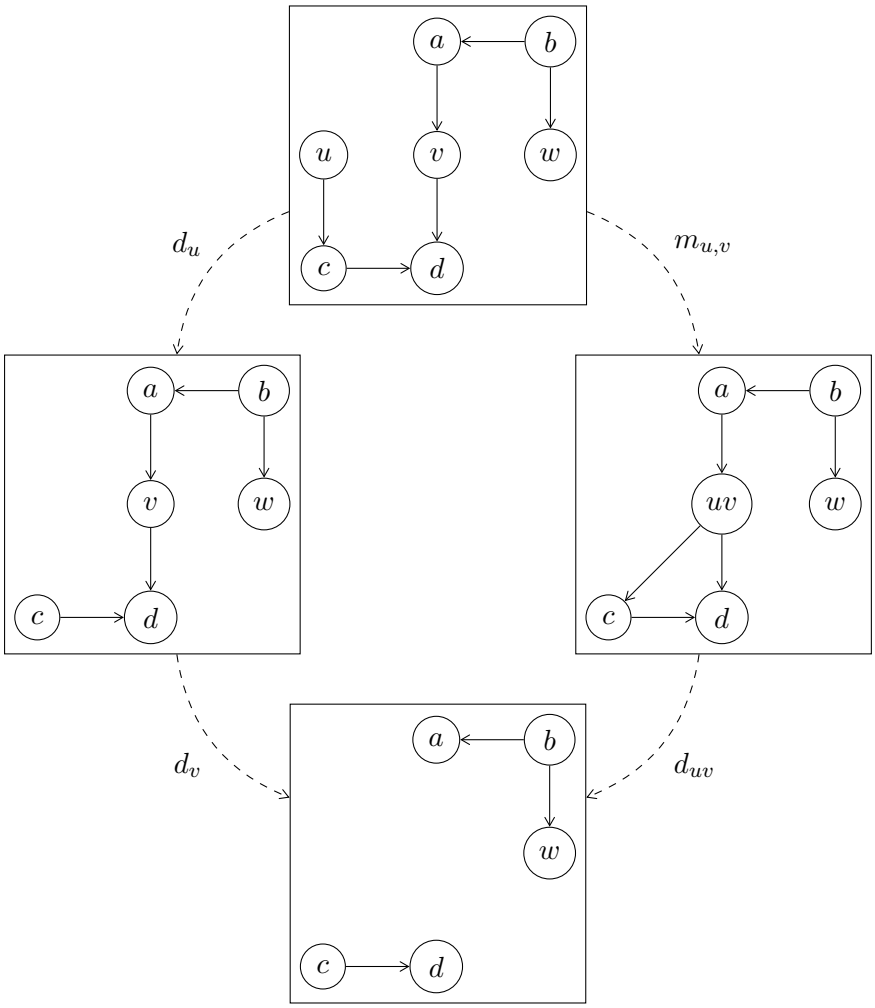
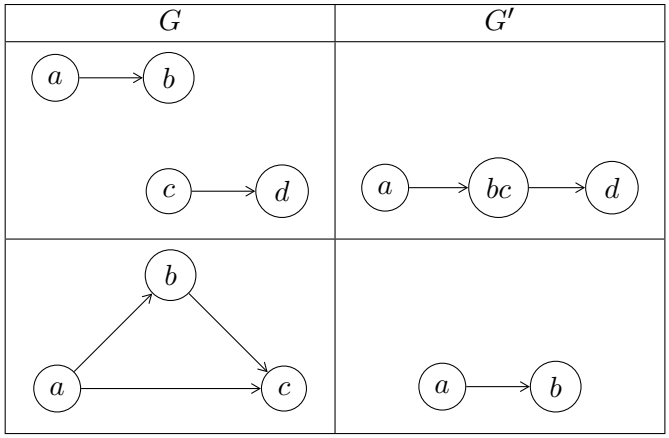


Figure 2.7: In the first row, $m_{b,c}(G) = G'$, but any deletion from G would lose an arc, so $G \longrightarrow_d G'$ is impossible. In the second row, $d_c(G) = G'$, but since there is an arc between any two vertices, every merge would make a loop, so $G \rightarrow_m G'$ does not hold.



2.2 Graph Morphisms

Given G and G', is there a sequence of operations that transform G into G'? To answer this question, the particular order of the operations is not important, so we can use commutation properties 2.3, 2.7 and 2.8 to reduce the search space. These results were already used in our first publication about SEPI in [31].

Doing this amounts to changing formalisms: for each chain of operations, there is a corresponding notion of graph morphism.

We first introduce classical notions of graph morphisms, and then introduce a notion of morphism that captures merge/delete strings.

2.2.1 Homomorphisms

Definition 2.14 (Graph homomorphism)**.** *A homomorphism from G to G' is a* total *function from V to V' that preserves the arcs of the G,* i.e.*, for all $u, v \in V$, if $(u, v) \in A$ then $(f(u), f(v)) \in A'$.*

2.2.2 Isomorphism

Definition 2.15 (Graph isomorphism)**.** *An isomorphism from G to G' is a bijective total function $f : V \to V'$ such that $(u, v) \in A$ iff $(f(u), f(v)) \in A'$.*

Two graphs G and G' are isomorphic when there exists a graph isomorphism from G

to G'. Graph isomorphism is an equivalence relation on directed graphs: we note $\mathcal{G}$ the set of all graphs quotiented by this equivalence relation.

2.2.3 Subgraph isomorphisms are deletion strings

The order in which deletions are applied to a graph do not matter, as seen in Prop. 2.7. When deciding whether G' can be obtained from G by deletions, using this property allows us to look for a morphism instead of a string of deletions. This basically removes symmetries from the search space.

Definition 2.16 (Subgraph isomorphism)**.** *A subgraph isomorphism from G to G' is an injective partial function $f : V \to V'$ such that $\forall (u, v)$ s.t. u and v are in $\mathrm{dom}(f)$, $(u, v) \in A$ iff $(f(u), f(v)) \in A'$.*

This is sometimes called *induced* subgraph isomorphism, because the subgraph of G induced by the vertices over which f are defined is isomorphic to G'.

Note that in our formalism, the morphism goes from the "big" graph to the "small" graph, not the other way around as in many other texts on (sub)graph (iso)morphisms [36]. Choosing this definition makes more sense in our context, as we will see in the next section: it allows merge to be captured by surjection.

Theorem 2.17. *There is a subgraph isomorphism from G to G' iff $G \to_d^* G'$.*

Proof. If $G' = d_u(G)$, then there is an obvious subgraph isomorphism which is defined everywhere but on u. Since subgraph isomorphisms compose (the composed function of two subgraph isomorphism is a subgraph isomorphism), by iteration we get the direct implication.

For the converse implication, if m is a subgraph isomorphism from G to G', then deleting from G the vertices on which m is not defined in any order yields G', thus $G \to_d^* G'$. □

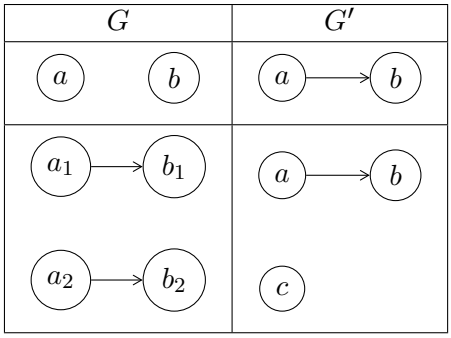
The subgraph isomorphism problem, i.e. determining whether there is a subgraph isomorphism from G to G' given G and G', is NP-complete, as shown in Cook's paper [16].

2.2.4 Epimorphisms are merge strings

When deciding whether G' can be obtained from G by mergings, Prop. 2.3 allows us to look for morphisms instead of strings of mergings. Once again this removes some symmetries from the search space.

Definition 2.18 (Epimorphism)**.** *An epimorphism from G to G' is a total function $f : V \to V'$ such that $(u, v) \in A \Rightarrow (f(u), f(v)) \in A'$, f surjective (onto) on V', and f onto on A', i.e.: $\forall (u', v') \in A', \exists (u, v) \in A, u' = f(u) \wedge v' = f(v)$.*

Figure 2.8: In the first row, there is a morphism from G to G' surjective on vertices, but none surjective on arcs. In the second row, there is a morphism from G to G' surjective on arcs, but none surjective on vertices.



In categorical lexicon (see [41]) an epi is an arrow h such that $g_1 \circ h = g_2 \circ h \Rightarrow g_1 = g_2$; the meaning in our case is that through an epimorphism, the source graph should completely capture the structure of the target graph; every node (arc) in the target graph comes from some vertex (arc) in the source graph.

Note that surjectivity on vertices does not imply surjectivity on arcs, and surjectivity on arcs does not imply surjectivity on vertices either, as shown in Fig. 2.8.

However, surjectivity on arcs implies surjectivity on vertices if there are no isolated vertices:

Theorem 2.19 ([36])**.** *Let $f : G \longrightarrow G'$ be a morphism surjective on arcs. Then f not surjective on $V' \Rightarrow \exists u' \in V', \forall (v', w') \in A', u' \notin \{v', w'\}$.*

This result will be useful later, since we will consider connected graphs with at least an arc.

Theorem 2.20. *There exists an epimorphism from G to G' iff $G \rightarrow_m^* G'$.*

Proof. If $G' = m_{u,v}(G)$, then there is an obvious epimorphism which has $m(u) = m(v)$. Since epimorphisms compose, by iteration we get the backwards implication.

For the direct implication, if m is an epimorphism from G to G', then merging the elements of $m^{-1}(u')$ for every u' vertex of G' yields a graph isomorphic to G', thus $G \rightarrow_m^* G'$. □

Proposition 2.21. *If $G \rightarrow_m^* G'$, then:*

- *G has no vertices iff G' has no vertices*
- *G has no arcs iff G' has no arcs*

Proof. The vertices case is obvious. Regarding the arcs case, by theorem 2.20, arc surjectivity implies G has more arcs than G', hence the forward implication. For the converse, if G' has no arcs, then the arc codomain of an epimorphism on G' is empty, since epimorphisms are defined everywhere, the domain must also be empty. □

The Ph.D. thesis of Narayan Vikas [68] provides an extensive study of graph epimorphisms, which are graph compaction in his nomenclature.

2.2.5 Subgraph Epimorphisms are merge/deletion strings

There are notions of morphism corresponding to $\to_d^*$ and $\to_m^*$. Instead of looking for strings of mergings and deletions, we can compress strings in one notion of morphism:

Definition 2.22 (Subgraph Epimorphism)**.** *A subgraph epimorphism from G to G' is a function $f : V \to V'$ such that $\forall (u, v)$ with u and v are in* $\mathrm{dom}(f)$*, $(u, v) \in A \Rightarrow (f(u), f(v)) \in A'$, f surjective (onto) on V' and A'.*

Theorem 2.23. *There exists a subgraph epimorphism from G to G' iff $G \to_{md}^* G'$.*

Proof. Let us prove the backwards implication first. We can use Prop. 2.8 to reorder operations so that deletions happen first, and then only mergings occur: $G \to_d^* G_1 \to_m^* G'$. Suppose the set of deleted vertices is V_1, then the mergings go from $d_{V_1}(G)$ to G_1, and from Thm. 2.20, there is a graph epimorphism μ from G_1 to G. This same μ is a subgraph epimorphism from G to G'.

The direct implication can be proved easily: from a subgraph epimorphism μ from G to G', first delete the set V_1 of all vertices on which μ is not defined, then merge the remaining vertices according to the partition defined by μ^{-1}. □

2.2.6 Coding Subgraph Epimorphisms with Epimorphisms of pointed graphs

Following property 2.12, we can encode subgraph epimorphisms with epimorphisms and pointed graphs:

Theorem 2.24. *There exists a subgraph epimorphism from G to G' iff there exists an epimorphism from $G_\perp$ to $G'_\perp$.*

This encoding property will be useful in both theory and practice, since it can be used to transfer properties of epimorphisms to subgraph epimorphisms, and in CP/SAT models to extend EPI decision procedures to SEPI ones.

Chapter 3

The SEPI Partial order

As an alternative to using SISO, SEPI can be used to answer the yes/no question "is graph G' a reduced version of G?". However, since these relations are not total orders, some pairs of graphs are incomparable.

Instead of using a yes/no question, one might want to compare graphs using a notion of distance. There exists two main ways for defining the distance between two graphs G and G':

(i) it may be defined in a denotational way, by means of the size of a largest subgraph common to G and G', i.e. by using the underlying partial order;

(ii) it may be defined in an operational way, by means of the minimum cost sequence of graph edit operations that should be performed to transform G into G', i.e. by using the state graph induced by allowed edition operations.

In [9], Bunke has connected these two definitions in the context of edit operations that generalize deletion, by introducing a special cost function for the graph edit distance, and by showing that under this cost function, the graph edit distance problem is equivalent to the maximum common subgraph computation.

Connecting an edition distance into a partial order distance reduces the search space, in the same way translating strings of operations into morphisms does.

While it is always possible to transform one graph into another by performing a sequence of vertex insertion and deletion operations, the size of such a sequence may not be representative of the similarity of the two graphs. Indeed, in some applicative contexts, it is more relevant to measure the distance between two graphs not only by means of vertex insertion and deletion operations, but also by means of vertex merge and split operations, thus leading to the extended graph edit distance [1, 13].

Here we introduce a graph edit distance which also allows merge operations. This distance is a simplified case of the extended graph edit distance introduced in [1], which has been defined for labeled graphs and which is parametrized by edit costs. Here we provide

a means to compute this distance using a notion akin to maximal subgraph computation, but in the SEPI setting. The result is one of the contributions of [29].

In the first part of this chapter, we will see that SEPI can also be used to formalize the notion of graph contained in both G and G', and of graph containing both G and G'.

In the second part of this chapter, we study edition distances, and relate them to the partial orders SEPI, EPI, and SISO.

For this chapter, $G = (V, A)$ and $G' = (V', A')$ are graphs, and o is an operation set among $\{m\}$, $\{d\}$ and $\{m, d\}$.

3.1 Partial Order: intersection, union

The arrows $\rightarrow_o^*$ defined in the previous chapter are binary relations on $\mathcal{G}$; since they are transitive, reflexive and antisymmetric, they are also partial orders on $\mathcal{G}$. We are interested in studying the partially ordered set (poset) $(\mathcal{G}, \rightarrow_{md}^*)$.

Some definitions first are in order. Let $(E, <)$ be a poset, and X a subset of E. All following notions are defined in the context of this order on this set ; when comparing orders on the same set, the context will have to be specified first.

A *minimal element* of X is an element $m \in X$ s.t. $\forall y \in X, y \not< x$. If there is a unique minimal element m in X, we call it the *minimum* of X. We define *maximal* and *maximum* elements similarly.

A *lower bound* of X is an element $b \in E$ such that $\forall x \in X, b < x$. If the set of lower bounds of X has a maximum, it is called the *greatest lower bound* (glb) of X. We define *upper bound* and *least upper bound* (lub) similarly.

Definition 3.1. *We will use specific notations for the sets of bounds of G and G'. We write:*

- $G \cap_o G' = \{H \mid G \rightarrow_o^* H \wedge G' \rightarrow_o^* H\}$,
- $G \cup_o G' = \{H \mid H \rightarrow_o^* G \wedge H \rightarrow_o^* G'\}$,
- $\overline{G \cap_o G'}$ *the set of* $\rightarrow_o^*$-maximal *elements of* $G \cap_o G'$, *and*
- $\underline{G \cup_o G'}$ *the set of* $\rightarrow_o^*$-minimal *elements of* $G \cup_o G'$.

$G \cap_o G'$ and $G \cup_o G'$ embody the elements that capture shared and specific structure of G and G', $\overline{G \cap_o G'}$ and $\underline{G \cup_o G'}$ are the elements that do so optimally.

3.1.1 Emptyness

The first question that arises for every operation we consider is: are bound sets empty?

In non-trivial cases, the lower bound set is never empty:

Proposition 3.2. *$G \cap_d G'$ and $G \cap_{md} G'$ are not empty.*
If G and G' have at least a vertex and an arc, $G \cap_m G'$ is not empty.

Proof. The empty graph $(\varnothing, \varnothing)$ is in $G \cap_d G'$ and $G \cap_{md} G'$.
If both G and G' have at least an arc, then $(\{1\}, \{(1,1)\})$ is in $G \cap_m G'$. □

In non-trivial cases, the upper bound set is never empty either. In order to deal with $G \cup_m G'$, we use a graph construction:

Definition 3.3 (Product Graph)**.** *The* product graph *(see for example [36]) of G and G' is:*

$$G \times G' = (V \times V', \{((u,u'),(v,v')) \mid (u,v) \in A \wedge (u',v') \in A'\})$$

Note that $G \times G'$ is isomorphic to $G' \times G$.

Example 3.4. *When G, G' are simple enough, we can represent G and G' as one-dimensional projections of $G \times G'$. This is shown in Fig. 3.1.*

Proposition 3.5. *If G and G' have at least one vertex and one arc, then $G \times G' \to_m^* G$ and $G \times G' \to_m^* G'$.*

Proof. For instance $\pi_1 : \big((u,v) \longrightarrow u\big)$ is an epimorphism from $G \times G'$ to G. Indeed, since G' has at least a vertex, every node of G has an antecedent by π_1 in $G \times G'$, similarly every arc of G has an antecedent by π_1 in $G \times G'$. By construction it is obvious that π_1 is a morphism. □

Proposition 3.6. *$G \cup_d G'$ and $G \cup_{md} G'$ are not empty.*
If G and G' have at least one vertex and one arc, then $G \cup_m G'$ is not empty.

Proof. The disjoint union of G and G', $G \uplus G' = (V \uplus V', A \uplus A')$, is in both $\cdot \to_d^* G$ and $\cdot \to_{md}^* G$, since deleting the vertices of G' from $G \uplus G'$ yields G. Similarly, $G \uplus G'$ is in $\cdot \to_d^* G'$ and $\cdot \to_{md}^* G'$,
If both G and G' have vertices and arcs, then from property 3.5, $G \times G' \in G \cup_m G'$. □

3.1.2 Finiteness

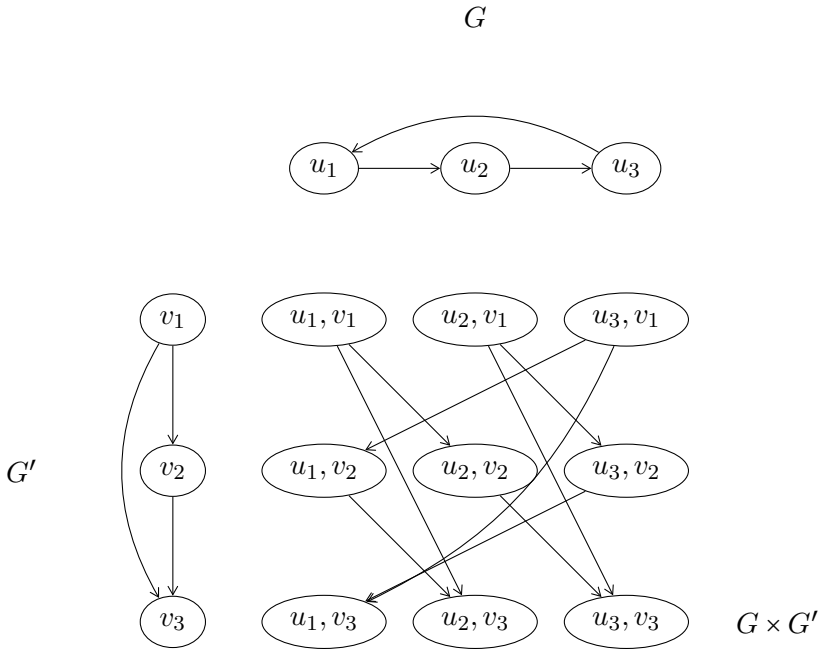
When bounds exist, we may want to compute them. In order to deal with computation issues, we should first answer basic questions about the size of bounds, especially extremal bounds.

Proposition 3.7. *$G \cap_o G'$ is finite.*

Proof. Trivial, since delete and merge reduce the number of vertices, and graphs with less than $\min(|G|, |G'|)$ vertices are in finite number. □

Proposition 3.8. *$\underline{G \cup_o G'}$ is finite.*

Figure 3.1: Depiction of a product graph $G \times G'$ as a 2-dimensional structure. Note how merging vertices of $G \times G'$ on the same row yields G', and merging vertices of $G \times G'$ on the same column yields G.



Proof. Let $K \in G \cup_o G'$, with g and g' corresponding morphisms. Suppose $\exists u, v \in K$ s.t. $(g(u), g'(u)) = (g(v), g'(v))$. Then obviously $m_{u,v}(K) \in G \cup_o G'$ and $d_u(K) \in G \cup_o G'$, so K is not minimal, whether $o = m$, $o = d$ or $o = md$.

Thus, two vertices of a minimal element K^* of $G \cup_o G'$ cannot have the same pair of images, and since there are $(|G|+1)(|G'|+1)$ different possible pairs of images (taking "not defined" as a possible image), a minimal graph has size at most $(|G|+1)(|G'|+1)$. Which means there are a finite number of such graphs. □

Of course, when $G \cup_o G'$ is not empty, it is infinite: for example, if $U \in G \cup_o G'$, then the graph nU that contains n disjoint copies of U is in $G \cup_o G'$.

3.1.3 Lattice structure

When compared to the inclusion order on sets, which yield a unique intersection and unique union, the SEPI order on graphs yield a more irregular structure: $(\mathcal{G}, \to_o^*)$ is not a lattice, not even a semi-lattice. Indeed, for every operation set o, we exhibit counterexamples proving that there are G, G' for which $\overline{G \cap_o G'}$ is not a singleton, and there are G, G' for which $\underline{G \cup_o G'}$ is not a singleton.

This is also the case for other formalisms, such as for linear algebra, where minimal generating families of vector spaces and maximal free families of vector spaces are not uniquely defined, but at least they all have the same size. Given o, we could expect elements of $\overline{G \cap_o G'}$ to have the size size, or elements of $\underline{G \cup_o G'}$ to have the same size. This is not the case either.

The counterexamples are summed up in the following table:

	$\overline{\cap_o}$	$\underline{\cup_o}$
	$\neq \{x\}$, $\lvert E\rvert \neq C$	$\neq \{x\}$, $\lvert E\rvert \neq C$
$\{d\}$	Fig. 3.2	Fig. 3.4
$\{m\}$	Fig. 3.3	Fig. 3.5
$\{md\}$	Fig. 3.2	Fig. 3.4

3.2 Partial Order Distance

If we are to define a distance using the partial order structure, there are two ways to do it: either using greatest lower bounds or least upper bounds.

Definition 3.9 (Partial order distances)**.** *For $o \in \{d, m, md\}$,*

- $d_o^{\cap}(G, G') = \min\{(|G| - |I|) + (|G'| - |I|) \ \ s.t. \ \ I \in G \cap_o G'\}$
- $d_o^{\cup}(G, G') = \min\{(|U| - |G|) + (|U| - |G'|) \ \ s.t. \ \ U \in G \cup_o G'\}$

Figure 3.2: $\overline{G \cap_o G'}$ is not always a singleton, and its elements do not always have the same size, for $o = d$ and $o = md$. Here $H, H' \in \overline{G \cap_o G'}$, but the have different sizes.

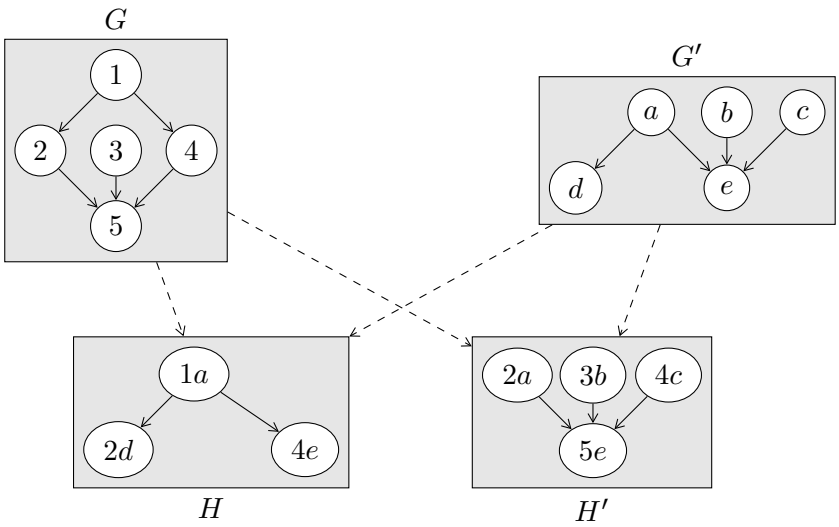


Figure 3.3: $\overline{G \cap_o G'}$ is not always a singleton, and its elements do not always have the same size, for $o = m$. Here $H, H' \in \overline{G \cap_o G'}$, but the have different sizes.

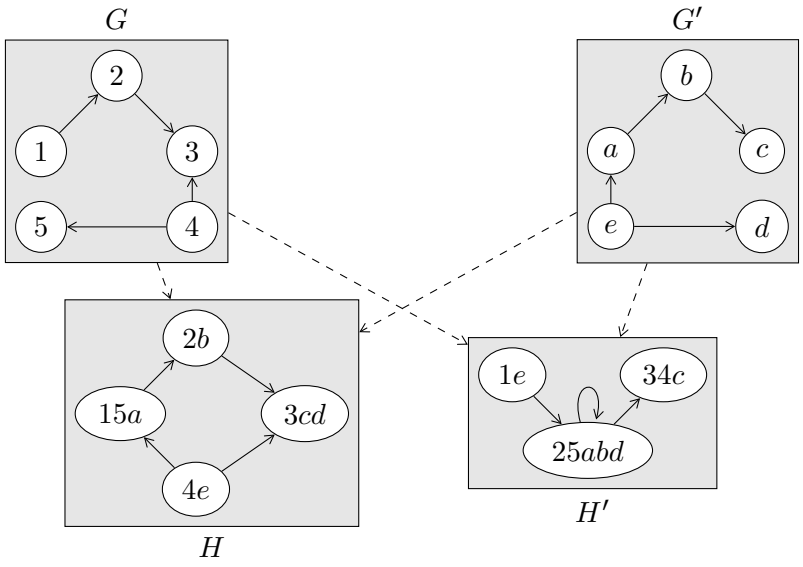
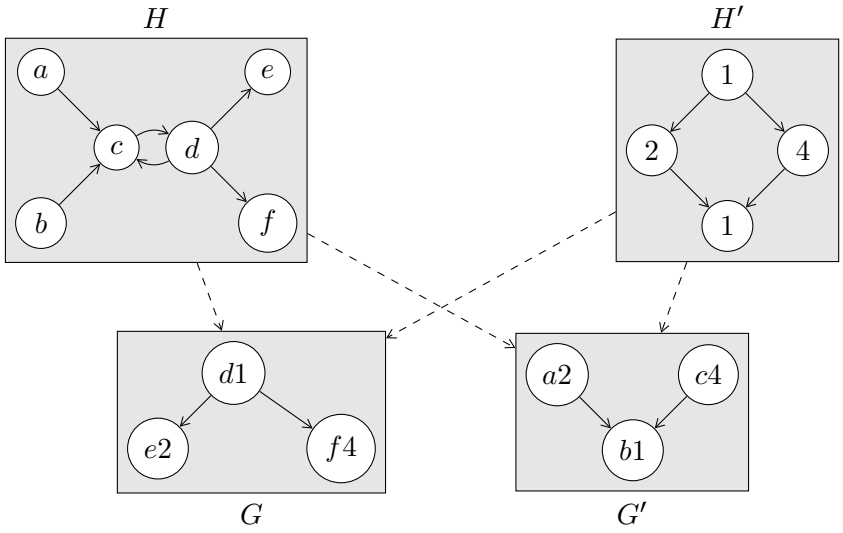


Figure 3.4: $\underline{G \cup_o G'}$ is not always a singleton, and its elements do not always have the same size, for $o = d$ and $o = md$. Here $H, H' \in \underline{G \cup_o G'}$, but the have different sizes.



with $\min \varnothing = +\infty$.

Note that a common graph that realizes the distance will always be an extremal graph, i.e. either in $\overline{G \cap_o G'}$ for $d_o^{\cap}(G, G')$ or in $\underline{G \cup_o G'}$ for $d_o^{\cup}(G, G')$.

Before proving that these are distances, let us compare the functions for a given set of operations o. When $o = d$, the functions coincide:

Theorem 3.10. $\forall G, G', d_d^{\cap}(G, G') = d_d^{\cup}(G, G')$

Proof. First, the functions are never infinite, thanks to Prop. 3.2. We will use a construction to show $d_d^{\cup}(G, G') \leq d_d^{\cap}(G, G')$.

Suppose I realizes the minimum for $d_d^{\cap}(G, G')$, and let $\mu : G \to I$ and $\mu' : G' \to I$ be the corresponding morphisms. For $(u, v) \in A$, let $\mu_\perp((u, v))$ be a function that conserves only deleted arcs, by setting $\mu_\perp((u, v)) = (u, v)$ if μ not defined on both u and v, $(u, \mu(v))$ if μ not defined on u, $(\mu(u), v)$ if μ not defined on v, and $\mu_\perp((u, v))$ not defined otherwise. We define $\mu'_\perp((u, v))$ the same way.

Let U be a graph with vertices $V(I) \uplus \mu^{-1}(\perp) \uplus \mu'^{-1}(\perp)$ and arcs $A(I) \uplus \mu_\perp(A(G)) \uplus \mu'_\perp(A(G'))$. Then $\nu : u \in \mu'^{-1}(\perp) \to u$ and $u \in V(I) \to \mu^{-1}(u)$ is a SISO from U to G, and we can build a similar function from U to G'. So the graph U is a witness that $d_d^{\cup}(G, G') \leq d_d^{\cap}(G, G')$.

The inequality $d_d^{\cup}(G, G') \geq d_d^{\cap}(G, G')$ can be proved by starting from a graph U, for

Figure 3.5: $\underline{G \cup_o G'}$ is not always a singleton, and its elements do not always have the same size, for $o = m$. Here $H, H' \in \underline{G \cup_o G'}$, but the have different sizes. Notice that $H' = G \times G'$.

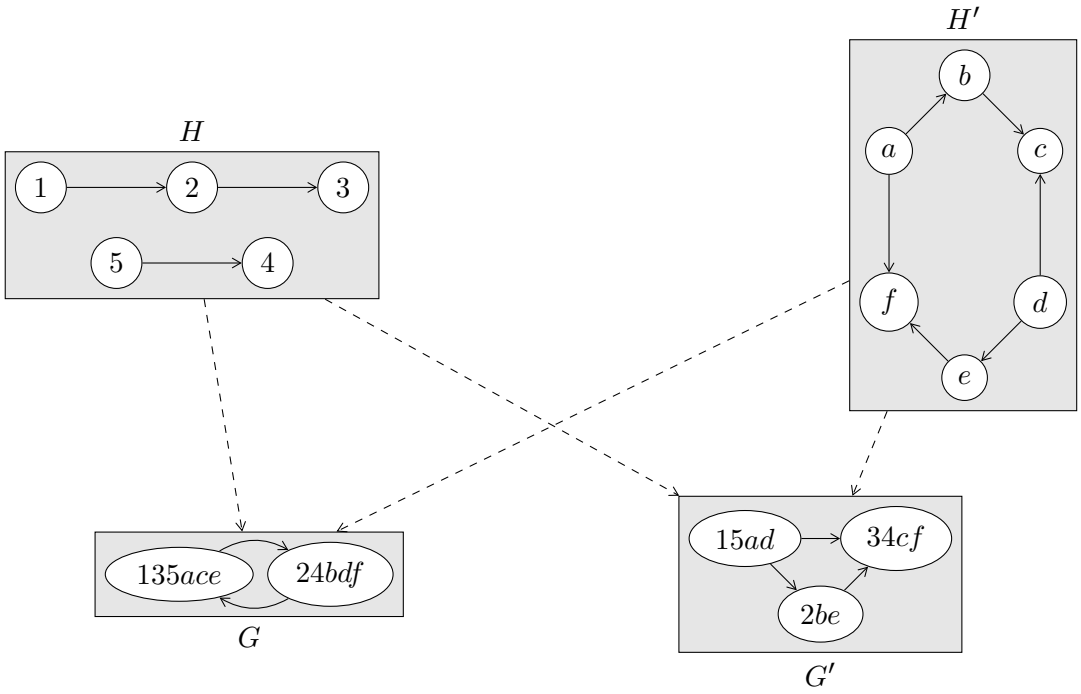
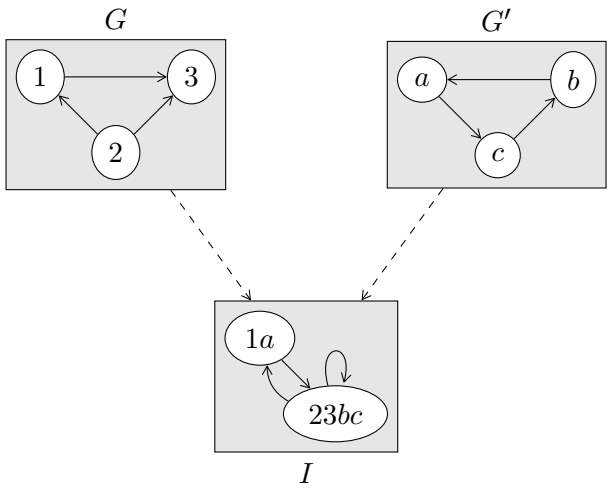


Figure 3.6: An example of graphs such that $d_m^{\cap}(G, G') < d_m^{\cup}(G, G')$. Graph I proves that $d_m^{\cap}(G, G') = 2$, but $d_m^{\cup}(G, G') > 2$.



the "union", that realizes $d_d^{\cup}(G, G')$, and then building a witness I for the "intersection" is even simpler. □

When $o = m$, we can exhibit a counter-example to the equality.

Example 3.11. $\exists G, G', d_m^{\cup}(G, G') \neq d_m^{\cap}(G, G')$. *Fig. 3.6 shows an example for which* $d_m^{\cap}(G, G') = 2$, *but* $d_m^{\cup}(G, G') > 2$. *Let us prove this by showing that there is no graph* U *of size* 4 *such that* $U \to_m^* G$ *and* $U \to_m^* G'$.

First, if there is such a graph, then the morphisms to G *and* G' *have to be chains of exactly one merge. Let the vertices be* $\{\alpha, \beta, \gamma, \alpha', \}$, *and the epimorphism to* G' *be* $m_{\alpha\beta}$. *Thanks to the symmetry of* G', *we can assume the image of* α *and* α' *to be* a. *Renaming allows us to assume that* β *has image* b, *and* γ *has image* c.

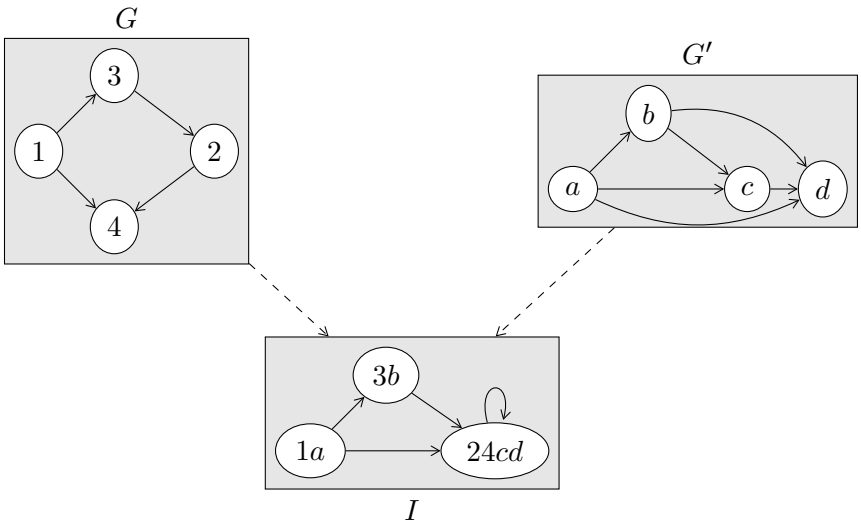
The arc surjection forces ac *to have an antecedent in* U, *either* $\alpha\gamma$, $\alpha'\gamma$ *or both. Renaming allows us to assume that* $\alpha\gamma$ *is an arc in* U. *Then* $\gamma\beta$ *is also an arc, and either* $\beta\alpha$ *or* $\beta\alpha'$. *In any case, there is a path of length* 3 *in* U.

However, the image of this path cannot fit anywhere in G. *So there cannot be a graph of size* 4 *in* $G \cup_m G'$, *which proves* $d_m^{\cup}(G, G') > 2$.

When $o = md$, despite the deletion part for which the functions coincide, we can also exhibit a counter-example to $d_{md}^{\cup}(G, G') = d_{md}^{\cap}(G, G')$:

Example 3.12. $\exists G, G', d_{md}^{\cup}(G, G') \neq d_{md}^{\cap}(G, G')$. *Fig. 3.7 shows an example for which* $d_{md}^{\cap}(G, G') = 2$, *but* $d_{md}^{\cup}(G, G') > 2$.

Figure 3.7: An example of graphs such that $d^{\cap}_{md}(G,G') < d^{\cup}_{md}(G,G')$. Graph I proves that $d^{\cap}_{md}(G,G') = 2$, but $d^{\cup}_{md}(G,G') > 2$.



Let us prove this. Once again, suppose there exists U of size 5 with $U \to^*_{md} G$ and $U \to^*_{md} G'$. Then the morphisms μ and μ' have to be a single merge or a single delete.

If μ' is a single delete, then U contains a copy of G' as an induced subgraph. The image of this copy through mu, whether μ is a delete or a merge, has to contain a T_3 (i.e. a graph isomorph to abc in G'). However G contains no T_3, so μ' cannot be a delete.

If μ' is a single merge, it merges two vertices α, α' in one of a, b, c or d. The other vertices always form a T_3. So μ cannot be d_α, $d_{\alpha'}$ or $m_{\alpha\alpha'}$, or G would contain a T_3. This means α and α' are mapped to different vertices in G. These vertices cannot be adjacent, or there would be an arc $\alpha\alpha'$ in U, and the image of this arc through μ' would be a loop in G', impossible. The images of $\{\alpha, \alpha'\}$ is either $\{1,2\}$ or $\{3,4\}$. The remaining vertices in U have to cover the rest, so they have to cover either $\{1,2\}$ or $\{3,4\}$. But there is always an arc between two of the remaining vertices, which would force an arc between $\{1,2\}$ or $\{3,4\}$, impossible.

So there is no graph of size 5 in $G \cup_{md} G'$, which proves $d_{md}(G,G') > 2$.

There are no examples where $d^{\cap}_o(G,G') > d^{\cup}_o(G,G')$: actually, we will prove that $d^{\cap}_o(G,G') \leq d^{\cup}_o(G,G')$ in Prop. 3.35. Since that result will be used to prove an even stronger one, we delay its exposition until then.

We will show that computing $d^{\cap}_o$ and $d^{\cup}_o$ is feasible in Chap. 6. However, we still do not know if these functions are distances, and whether we should favor $d^{\cap}_o$ or $d^{\cup}_o$. In order to have a better view on the problem, we will now study edition distances.

3.3 Edition distance

When working with graph edition operations, the notion of edition distance is a natural way to *define* a distance.

Let us first define the edition graph which associates a vertex with every graph equivalence class of $\mathcal{G}$, and which associates an arc with every pair (G_1, G_2) such that one can transform any graph of G_1 into any graph of G_2 by applying one operation. We define three different edition graphs, depending on the set of operations o.

Definition 3.13 (Edition graph)**.** *Let $o \in \{m, d, md\}$. The* edition graph $\mathcal{E}_o$ *is the pair $(\mathcal{G}, \mathcal{A}_o)$ such that $\mathcal{A}_o = \{(G_1, G_2) \in \mathcal{G} \times \mathcal{G} \mid G_1 \to_o G_2\}$.*

These edition graphs are not strongly connected. For example, there is no path from any graph G to any graph G' which has more vertices than G as for every arc $G_1 \to_o G_2$, we have $|G_2| = |G_1| - 1$. This prevents us from defining a metric using paths. However, there is a natural definition using the *walks* of $\mathcal{E}_o$. If a path from s to t only crosses arcs forwards, walks extend paths by allowing to cross arcs forwards and backwards:

Definition 3.14 (Walk of a graph)**.** *Let $G = (V, A)$ be a graph, and u, v be vertices of G. A* walk *w from u to v is a finite sequence $w = (a_1 \dots a_n)$ of arcs of A such that $\exists x_0 \dots x_n \in V, x_0 = u, x_n = v$, and $\forall 1 \leq i \leq n, a_i = (x_{i-1}, x_i) \vee a_i = (x_i, x_{i-1})$.*

The length *of w is $|w| = n$.*

Let us now define a graph edit distance as the length of a shortest walk.

Definition 3.15 (Distance)**.** *Let $o \in \{m, d, md\}$, and $G_1, G_2 \in \mathcal{G}$. The distance $d_o : \mathcal{G} \to \mathbb{N} \cup \{+\infty\}$ is*

$$d_o(G_1, G_2) = \min\{|w| \text{ s.t. } w \text{ is a walk of } \mathcal{E}_o \text{ from } G_1 \text{ to } G_2\}$$

with $\min \varnothing = +\infty$.

Example 3.16. *On the graphs G and G' of the Michaelis-Menten reduction given in the introduction, we have*

$$\begin{aligned} d_{md}(G, G') &= 3 \\ d_m(G, G') &= 3 \\ d_d(G, G') &= 5 \end{aligned}$$

For $o \in \{d, md\}$, the distance $d_o(G, G')$ is never $+\infty$, as there always exists a walk from G to G' which goes through the empty graph. However, when $o = m$, it may happen that $d_o(G, G') = +\infty$. This is the case, for example, when G has no arcs whereas G' has at least one arc.

Compared to functions $d_o^\cap$ and $d_o^\cup$, the fact that d_o is a metric on $\mathcal{G}$, i.e.that it satisfies the non-negativity, symmetry, separability and triangular inequality properties, is easy to check.

$d_o^\cap$, $d_o^\cup$ and d_o can be related straightforwardly:

Proposition 3.17. *$d_o(G,G') \leq d_o^{\cap}(G,G')$ and $d_o(G,G') \leq d_o^{\cup}(G,G')$.*

Proof. When $I \in G \cap_o G'$, there is a walk $G \to_o^* I \leftarrow_o^* G'$; when $U \in G \cup_o G'$, there is a walk $G \leftarrow_o^* U \to_o^* G'$. Minimum sized walks will be even shorter, hence the property. □

The three distances d_o can be compared among themselves:

Proposition 3.18. *Let $G, G' \in \mathcal{G}$. Then:*

$$\begin{array}{rcl} d_{md}(G,G') & \leq & d_m(G,G') \\ d_{md}(G,G') & \leq & d_d(G,G') \\ d_m(G,G') & \leq & 3d_d(G,G') \\ d_{md}(G,G') & \leq & 3d_d(G,G') \end{array}$$

Proof. The two first inequalities can be proved using $\mathcal{E}_{md} \supseteq \mathcal{E}_m$ and $\mathcal{E}_{md} \supseteq \mathcal{E}_d$. The third and fourth can be proved by simulating every merge operation by the deletion of both vertices to be merged, and addition (undeletion) of the merged vertex. □

In [9], Bunke has shown that the graph edit distance which only considers vertex deletions (i.e., d_d) is related to the size of the maximum common subgraph. In the next sections, we extend this result to graph edit distances which consider vertex merges (i.e., d_o when $o \in \{m, md\}$) by relating them to EPI and $SEPI$.

3.4 Quotient graphs

In this section, we define quotient graphs, which are another way to describe merging operations. This general construction entails cardinality results that would be less clear if shown in ad-hoc proofs.

We equate merge operations to quotienting by equivalence relations, then link the number of merges with the dimension of equivalence relations.

3.4.1 Equivalence relations

In this section, S is a finite set. A binary relation α over S is called an *equivalence relation* over S iff it has the following properties:

- reflexivity: $\forall x \in S, (x,x) \in \alpha$
- symmetry: $\forall x \in S, \forall y \in S, (x,y) \in \alpha \Rightarrow (y,x) \in \alpha$
- transitivity: $\forall x \in S, \forall y \in S, \forall z \in S, (x,y) \in \alpha \wedge (y,z) \in \alpha \Rightarrow (x,z) \in \alpha$

Definition 3.19. *Let α be an equivalence relation over a set S and $x \in S$. The class of x modulo α, denoted $[x]_\alpha$, is $[x]_\alpha = \{y \in A \mid (x,y) \in \alpha\}$.*

The set of classes modulo α, denoted S/α, is $S/\alpha = \{[x]_\alpha \mid x \in S\}$.

Definition 3.20 (Transitive closure)**.** *Let $\alpha \subseteq X \times Y, \beta \subseteq Y \times Z$. The* composition *of α and β is $\alpha \cdot \beta = \{(x,z) \mid \exists y, (x,y) \in \alpha \wedge (y,z) \in \beta\}$.*

The transitive closure *of α is the relation $\alpha^+ = \cup_{i=1}^{\infty} \alpha^i$ with $\alpha^1 = \alpha$ and $\forall i \geq 2, \alpha^{i+1} = \alpha \cdot \alpha^i$.*

The reflexive transitive symmetric closure *of α is the relation*

$$\alpha^{\equiv} = \{(x,y) \mid (x,y) \in \alpha \vee (y,x) \in \alpha \vee x = y\}^+.$$

For any α, $\alpha^{\equiv}$ is an equivalence relation, the smallest containing α.

Definition 3.21. *Let α, β be equivalence relations over S. The* product of equivalence relations *is $\alpha * \beta = (\alpha \cup \beta)^+$. It is an equivalence relation, the smallest (inclusion-wise) containing both α and β.*

3.4.2 Dimension of equivalence relations

For the remainder of this section, let α and β be equivalence relations over S.

Definition 3.22. *Let s be a binary relation over S.*

s is called a spanning *of α iff $s^{\equiv} = \alpha$.*

s is called a free family *(or just* free *for short) iff it has no loops and no cycles.*

As subsets of $S \times S$, spannings are ordered by inclusion. The minimal spannings are free, analogously to minimal generating families in vector spaces. Minimal spannings share a common size, which enables the definition of dimension:

Proposition 3.23. *Let s be a spanning of α. Then s is minimal iff s is free.*

In this case, $|s| = |E| - |E/\alpha|$.

Proof. Suppose s is minimal and has a cycle $e_1 \dots e_n$. Since $e_n \in (s - \{e_n\})^{\equiv}$, $s - \{e_n\}$ is a spanning of α, so s is not minimal, which is contradictory. Likewise, s has no loops.

Now suppose s is free. Let us prove $|s| = |E| - |E/\alpha|$. Let $n = |E/\alpha|$. Notice that $n \geq 1$. When $n = 1$, the undirected version of s ($s \cup s^{-1}$) is a tree, so if it covers $k \geq 1$ vertices, it has $k - 1$ arcs.

We have $s \subseteq \bigcup_{[x]_\alpha \in E/\alpha} [x]_\alpha \times [x]_\alpha$, so $s = \coprod_{[x]_\alpha \in E/\alpha} s \cap ([x]_\alpha \times [x]_\alpha)$.

Since $s \cap ([x]_\alpha \times [x]_\alpha)$ is a free spanning of $[x]_\alpha \times [x]_\alpha$, the argument for $n = 1$ gives $|s \cap ([x]_\alpha \times [x]_\alpha)| = |[x]_\alpha| - 1$.

So $|s| = \sum_{[x]_\alpha \in E/\alpha} (|[x]_\alpha| - 1) = |E| - |E/\alpha|$.

To conclude, every free spanning has cardinality $|E| - |E/\alpha|$. If $s_0 \subseteq s$ with s_0 minimal, since s_0 is also free, $|s_0| = |s|$, so $s = s_0$ and s minimal. □

From Property 3.23, one can define the dimension of an equivalence relation as follows:

Definition 3.24. *The* dimension $dim(\alpha)$ *of* α *is the the size of its minimal spannings* $|E| - |E/\alpha|$.

One can then show a theorem analogous to the incomplete basis theorem:

Theorem 3.25. *Let* $s \subseteq \alpha$ *be a free family. Then there is a minimal spanning* t *of* α *that contains* s.

Proof. Let us build an increasing sequence of free families of α that contain s. Let $s_0 = s$. If s_n doesn't span α, then for any $(x_n, y_n) \in \alpha - s_n^{\equiv}$, $s_n \cup \{(x_n, y_n)\}$ is free. In this case we define $s_{n+1} = s_n \cup \{(x_n, y_n)\}$. This sequence grows strictly within a finite set, so has to stop at some m. Then s_m has to span α. Since s_m is free, it is minimal. □

Next, we show that maximal free families are generating families.

Proposition 3.26. *Let* $s \subseteq \alpha$ *be free.*
If s *is a maximal free family, then* s *is a spanning of* α.
If s *has size* $dim(\alpha)$*, then* s *is a spanning of* α.

Proof. For the first assertion, suppose s doesn't span α, that is, $\exists (x, y) \in \alpha - s^{\equiv}$. Then $x \neq y$, since $s^{\equiv}$ is reflexive. Since $s \cup \{(x, y)\}$ is not free, it must have a cycle, which must contain (x, y), with every other arc of the cycle in s. This last statement means that $(x, y) \in s^{\equiv}$, which is absurd.

For the last assertion, suppose free family $s \subseteq \alpha$ has size $dim(\alpha)$. Using Theorem 3.25, let t be a minimal spanning of α that contains s. Since $dim(\alpha) = |t| \geq |s| = dim(\alpha)$, we have $s = t$. □

Proposition 3.27.

$$dim(\alpha * \beta) \leq dim(\alpha) + dim(\beta)$$

Proof. Let s, t be minimal spannings of α, β. $s \cup t$ is a spanning of $\alpha * \beta$, with $|s \cup t| \leq |s| + |t|$; a minimal spanning will be even smaller. □

The equality case gives an interesting result: in this case, for every s, t minimal spannings of α, β, $s \cup t$ has no cycle.

3.4.3 Quotients of graphs by equivalence relations

In this section, $G = (V, A)$ is a directed graph (i.e. $A \subseteq V \times V$), and α, β are equivalence relations over V.

Definition 3.28 (Quotient Graph)**.** *The* quotient *of* G *by* α *is:*
$G/\alpha = (\{[x]_\alpha \mid x \in V\}, \{([x]_\alpha, [y]_\alpha) \mid (x, y) \in A\})$.

Notice that G/α has $|V/\alpha|$ vertices.

The notions of graph epimorphisms and graph quotients are strongly related, in the sense conveyed by the following theorem:

Theorem 3.29. *There exists an epimorphism f from G to G' iff there exists an equivalence relation α over V such that G/α is isomorphic to G'.*

Proof. We prove the theorem using direct constructions:

$\Rightarrow$ Let $\alpha = \{(x, y) \in V^2 \mid f(x) = f(y)\}$. Then G/α is isomorphic to G'.

$\Leftarrow$ Let $f : V \to \mathcal{P}(V)$ such that $\forall u \in V, f(u) = [u]_\alpha$. Then f is an epimorphism from G to G/α. □

We shall use a classical theorem about equivalence classes.

Theorem 3.30. *Let γ be an equivalence relation with $\alpha \subseteq \gamma$. Then γ/α is an equivalence relation over V/α, and $dim(\gamma/\alpha) = dim(\gamma) - dim(\alpha)$.*

Proof. Showing that γ/α is an equivalence class over V/α is easy. The result on dimensions is less well-known.

Let s be a minimal spanning of α. From Property 3.23 s is a free family of γ, so by Property 3.25, there is a $t \supseteq s$ that is a minimal spanning of γ. t being a graph, we can consider quotienting it by α. One can show that t/α is actually a spanning of γ/α.

Now t/α generally contains loops and may not be a minimal spanning. Let $t_0 = t - s$. The arcs s/α are exactly the loops of t/α, which means $(t_0/\alpha)^{\equiv} = (t/\alpha)^{\equiv} = \gamma/\alpha$

One can prove that t_0/α is free, and its size is $|t| - |s|$, which allows us to conclude on the dimension of γ/α. □

Proposition 3.31. *Let γ an equivalence relation such that $\alpha \subseteq \gamma$. Then $(G/\alpha)/(\gamma/\alpha)$ is isomorphic to G/γ.*

Proof. Notice how the vertices of G/γ are subsets of V, and the vertices of $(G/\alpha)/(\gamma/\alpha)$ are subsets of subsets of V. It is easy to check that $m : X \in V/\alpha \longrightarrow \bigcup_{x \in X} x$ is an isomorphism from the latter graph to the former. □

3.5 Intersection theorem

In this section, we finally prove that $d_o = d_o^\cap$. This both makes $d_o^\cap$ the natural distance, and provides a practical way to cut the search space when computing d_o.

Let us begin by a simple property of our operations.

Proposition 3.32. *If there is a sequence of* merge *and* delete *operations that transform G into G', then this sequence has $|G| - |G'|$ operations.*

Proof. Each *merge* or *delete* operation decrements the number of vertices of G by 1. □

3.5.1 Delete distance

Let us show that in $\mathcal{E}_d$, there is always a short walk with a down-up pattern. This could actually be proven earlier using Thm. 3.10. This case is the simplest of the three.

Proposition 3.33. *Let $G, G_1 = (V_1, A_1), G_2 = (V_2, A_2) \in \mathcal{G}$. If there is a walk $w = G_1 \leftarrow_d^* G \rightarrow_d^* G_2$, then there exists a graph G' and a walk $w' = G_1 \rightarrow_d^* G' \leftarrow_d^* G_2$, with $|w'| \leq |w|$.*

Proof. Let $V' = V_1 \cap V_2$, and $G' = G_{\downarrow V'}$. There is a deletion string from G_1 (resp. G_2) to G', by deleting vertices $V_1 - V_2$ (resp. $V_2 - V_1$).

G contains vertices $V_1 \cup V_2$, but it also has the vertices that have been deleted on both paths to G_1 and to G_2, so that $|G| \geq |V_1 \cup V_2|$.

Using Property 3.32, w has length $|G| - |G_1| + |G| - |G_2| \geq |V_1 \cup V_2| - |V_1| + |V_1 \cup V_2| - |V_2| = |V_1 \cup V_2| - |V_1 \cap V_2|$, and w' has length $|V_1| - |V_1 \cap V_2| + |V_2| - |V_1 \cap V_2| = |V_1 \cup V_2| - |V_1 \cap V_2|$.

Thus $|w'| \leq |w|$. □

Theorem 3.34. *If there is a walk w of $\mathcal{E}_d$ from G_1 to G_2, then there is a graph G_c and a walk $w' = G_1 \rightarrow_d^* G_c \leftarrow_d^* G_2$ not longer than w.*

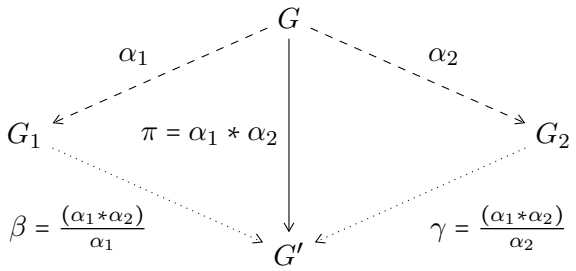
Proof. By recursion on the number of maximal 'peaks' of w, i.e. the number of maximal subwords of $w \in \rightarrow_d \cdot \leftarrow_d$: using Property 3.33 decreases the number of maximal peaks, and at each step the resulting walk from G_1 to G_2 is shorter or has the same length. □

3.5.2 Merge distance

To show the same kind of properties for merge operations, graph quotients come into play.

Proposition 3.35. *Let $G, G_1, G_2 \in \mathcal{G}$. If there is a walk $w = G_1 \leftarrow_m^* G \rightarrow_m^* G_2$, then there exists a graph G' and a walk $w' = G_1 \rightarrow_m^* G' \leftarrow_m^* G_2$, with $|w'| \leq |w|$.*

Proof. This illustrates the construction:



Using Theorem 3.29, we see a string of merge operations as one graph quotient. Let α_1 and α_2 such that $G_1 = G/\alpha_1$ and $G_2 = G/\alpha_2$. Let $\pi = \alpha_1 * \alpha_2$.

With $\beta = \pi/\alpha_1$ and $\gamma = \pi/\alpha_2$, we have, using Property 3.31, $G/\pi = G_1/\beta = G_2/\gamma$ (in other words, $\frac{G}{\alpha_1 * \alpha_2} = \frac{G}{\alpha_1}/\frac{(\alpha_1 * \alpha_2)}{\alpha_1} = \frac{G}{\alpha_2}/\frac{(\alpha_1 * \alpha_2)}{\alpha_2}$).

So, by transitivity, G' has epimorphisms from G_1 and G_2. And since β and γ have respectively smaller dimensions than α_1 and α_2 (by Properties 3.27 and 3.30), the dotted lines in the figure above are shorter than the dashed lines, hence the property.

□

Theorem 3.36. *If there is a walk w of $\mathcal{E}_m$ from G_1 to G_2, then there is a graph G_c and a walk $w' = G_1 \to_m^* G_c \leftarrow_m^* G_2$ not longer than w.*

Proof. Same proof as Theorem 3.34, using Property 3.35. □

3.5.3 Merge-Delete Distance

Using a pointed graph coding, we can extend the distance result to the merge and delete operations at the same time: in short, vertex deletion can be simulated by merging with a dummy vertex.

Definition 3.37 (Dummy vertices, pointed graphs)**.** *Let $\perp$ be a fresh symbol that is not a vertex of any graph in $\mathcal{G}$.*

Let $\cdot_\perp : G \in \mathcal{G} \longrightarrow G_\perp = (V_\perp, A_\perp)$, where $V_\perp = V \uplus \perp$ and $A_\perp = A \uplus (V \times \{\perp\}) \uplus (\{\perp\} \times V) \uplus \{(\perp, \perp)\}$.

We call dummy vertex *of a graph G one that has all possible arcs to/from the other vertices of G and to itself. In $G_\perp$, $\perp$ is always a dummy vertex.*

We write the set of pointed graphs $\mathcal{G}_\perp = \{G_\perp \mid G \in \mathcal{G}\}$. We extend the merge operation on $\mathcal{G}_\perp$ with no special treatment for $\perp$: a priori, the image of $\perp$ can be any vertex, and the antecedents of $\perp$ can be any vertex.

Proposition 3.38. *$\cdot_\perp$ is an isomorphism from $\mathcal{E}_m$ to $(\mathcal{E}_\perp)_m$:*
$G \to_{md} G'$ if and only if $G_\perp \to_m G'_\perp$.

Proof. The left to right implication is straightforward. Let $\mu : G \longrightarrow G'$ the function corresponding to the operation, merging or deletion. If the operation is a merging, then sending $\perp$ to $\perp$ is valid. If it is a deletion, sending $\perp$ to $\perp$ and sending the deleted vertex to $\perp$ makes the operation a merging. So $\mu_\perp : \begin{pmatrix} v \in \mathrm{dom}(\mu) & \longrightarrow & \mu(v) \\ v \in V - \mathrm{dom}(\mu) & \longrightarrow & \perp \\ \perp & \longrightarrow & \perp \end{pmatrix}$ is a merging from $G_\perp$ to $G'_\perp$.

The converse implication should be done carefully. Let us call $\mu_\perp : G_\perp \longrightarrow G'_\perp$ the function corresponding to the merging. Notice that $\mu_\perp(\perp)$ is not necessarily $\perp$. However

$\mu_\perp(\perp)$ is necessarily a dummy vertex, so that $\omega : \begin{pmatrix} v \notin \{\mu_\perp(\perp), \perp\} & \longrightarrow & v \\ \mu_\perp(\perp) & \longrightarrow & \perp \\ \perp & \longrightarrow & \mu_\perp(\perp) \end{pmatrix}$ is a graph isomorphism of $G_\perp$.

Let $\rho = \omega \circ \mu_\perp$: it is a merging from $G_\perp$ to $G'_\perp$ with $\rho(\perp) = \perp$. One can check that $\mu : \begin{pmatrix} v & \longrightarrow & \rho(v) & \text{if } \rho(v) \neq \perp \end{pmatrix}$ is either a merging (when $\rho^{-1}(\perp) = \{\perp\}$), or a deletion of u (when $\rho^{-1}(\perp) = \{\perp, u\}$). □

Theorem 3.39. *If there is a walk w of $\mathcal{E}_{md}$ from G_1 to G_2, then there is a graph G_c and a walk $w' = G_1 \to^*_{md} G_c \leftarrow^*_{md} G_2$ not longer than w.*

Proof. Combining Theorem 3.36 and Property 3.38 yields the result. □

Let us recapitulate the results of this section:

Corollary 3.40. *Let $G, G' \in \mathcal{G}$.*

- $d_d(G, G') = |G| + |G'| - 2\max\{|G_c| \text{ s.t. } G_c \in G \cap_d G'\}$
- $d_m(G, G') = |G| + |G'| - 2\max\{|G_c| \text{ s.t. } G_c \in G \cap_d G'\}$
- $d_{md}(G, G') = |G| + |G'| - 2\max\{|G_c| \text{ s.t. } G_c \in G \cap_d G'\}$

Proof. Use theorems from section 2.2 to transpose Theorems 3.34, 3.36 and 3.39 to morphisms, then use Property 3.32 for cardinalities. □

The first equality is already known: the deletion distance between G and G' is the number of deletions to the maximum common induced subgraph G_c [9], which is *the* greatest lower bound of G and G' for the SISO partial order.

The other two equalities are new: G_c is a glb of maximal cardinality for the EPI (resp. SEPI) partial orders. Note that there may be several common graphs of maximum cardinality.

3.6 Comparison to graph minors

SEPI is the relation defined by vertex deletion and vertex merging. It may look similar to graph minors [55], which are defined by vertex deletion, arc deletion and merging vertices linked by an arc, but it is quite different.

The graph minor theory has been extensively studied. The "minor of" binary relation between graphs, which we will write $\to_{\text{minor}}$, enjoys the well-quasi-order property (WQO) [57]. That property entails that for every set S of graphs stable by the operations considered, this set can be characterized by a finite obstruction set $\mathcal{O}(S)$, i.e., a set of graphs such that if $G \notin S$, then G reduces to some $O \in \mathcal{O}(S)$.

In turn, for every given graph O, there is a polynomial time algorithm $\mathcal{A}_O$ which determines whether its input graph has O as a minor [56], which means that for every family stable by the minor operations, there is a polynomial-time algorithm that determines membership.

Is there such a property for SEPI? We will see that SEPI is not a WQO. Actually, this means that there is no simple way to encode SEPI in terms of minor. Let us see how to prove this. First, let us define the notion of well-quasi-order:

Definition 3.41 (Well-Quasi-Order)**.** *Let X be a set and $\leq$ be a subset of $X \times X$. R is a* well-quasi order *iff:*

1. $\forall x \in X, x \leq x$
2. $\forall x, y, z \in X, x \leq y \wedge y \leq z \Rightarrow x \leq z$
3. $\forall (x_i)_{i \in \mathbb{N}}$ *sequence of* X, $\exists i < j, x_i \leq x_j$

Even though SEPI is not monotonic in terms of degree of the vertices, number of connected components and the such, we observe that the number of *non-neighbors* always decreases:

Definition 3.42 (Non-neighbors)**.** *The set of outgoing (resp. incoming)* non-neighbors *of a vertex u in a graph G is the set of vertices $ONN(u, G) = \{v \in V \mid (u, v) \notin A\}$ (resp. $INN(u, G) = \{v \in V \mid (v, u) \notin A\}$).*

Proposition 3.43. *Suppose there is a SEPI f from G to G'. Then, for any vertex $x \in \mathrm{dom}(f)$,*
$f(ONN(x, G)) \supseteq ONN(f(x), G')$, and $f(INN(x, G)) \supseteq INN(f(x), G')$.

Proof. If $ONN(x', G') = \varnothing$, the case is immediately proved.

Suppose $ONN(x', G') \neq \varnothing$, and let $y' \in ONN(x', G')$. By surjectivity of f, let y such that $f(y) = y'$. We have $(x', y') \notin A'$, so, since f morphism, $(x, y) \notin A$. Thus $y \in ONN(x, G)$, which proves $y' \in f(ONN(x, G))$.

The proof for INN is similar. □

Comment 1. *The property for every vertex of G to have at most k outgoing (resp. incoming) non-neighbors is preserved by SEPIs. This property is used to show that SEPI is not a well-quasi-order.*

Proposition 3.44. *If f is a subgraph epimorphism from G to G', then $\forall u \in \mathrm{dom}\, f$, $f(ONN(u, G)) \supseteq ONN(f(u), G') \leq k$ (resp. $|INN(u, G)| \leq k$), then $\forall u' \in V', |ONN(u', G')| \leq k$ (resp. $|INN(u', G')| \leq k$).*

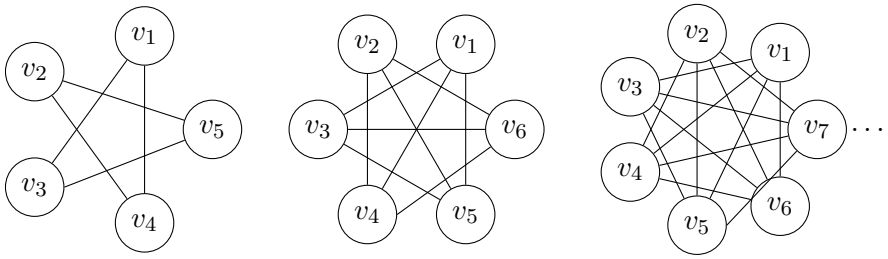
Proof. Let us prove the property for ONN by showing that for every vertex $u \in V$ such that $u \in \operatorname{dom} f$, we have $f(ONN(u,G)) \supseteq ONN(f(u),G')$, so that $|ONN(u,G)| \geq |ONN(f(u),G')|$. If $ONN(f(u),G') = \varnothing$, the case is proved. Suppose $ONN(f(u),G') \neq \varnothing$ and let $v' \in ONN(f(u),G')$. By surjectivity of f, let v such that $f(v) = v'$. We have $(f(u),v') \notin E'$, so, since f is an (epi)morphism, $(u,v) \notin E$. Thus $v \in ONN(u,G)$, which proves $v' \in f(ONN(u,G))$.

The proof for INN is similar. □

Proposition 3.45. *$SEPI$ is not a Well-Quasi-Order.*

Proof. In definition 3.41 of a well-quasi order, there are three items: items 1 and 2 is the definition of quasi-order (preorder), item 3 means there is no infinite descending chain and no infinite antichain.

One can exhibit an infinite antichain of graphs for $SEPI$. Let $G_n = (V_n, A_n)$ for $n \geq 5$, with $V_n = \{1...n\}$ and $A_n = \{(i,j)$ such that $|i-j| > 1[n]\}$.



This family is an infinite antichain for $SEPI$. First, one can easily check that $\forall n, \forall u \in V_n, |ONN(x,G_n)| = 3$ (in absence of loop a vertex is in its own ONNset). Let f be a $SEPI$ from G_n to G_m, with $n \geq m$.

Second, f does not delete any vertex. Suppose $D = \{i \mid i \notin \operatorname{dom} f\}$ is not empty. D cannot be V_n either, or G_m would not have any vertex. So there exists i such that one vertex in $\{i, (i+1)[n]\}$ is deleted and the other is not. Without loss of generality, say i is not deleted and $(i+1)[n]$ is. Then, from Prop. 3.43, $f(i)$ has at most 2 outgoing non-neighbors in G_m: itself and, if defined, the image of $(i-1)[n]$. Which is impossible, since every vertex of G_m has exactly 3 outgoing non-neighbors.

Next, f does not merge any vertices. Suppose f merges i with at least another vertex j. i and j have at most two outgoing non-neighbors in common. Indeed, remind that G_n is defined for $n \geq 5$: they have either two outgoing neighbors in common (when $|i-j| = 1[n]$), one (when $|i-j| = 2[n]$), or none (when $|i-j| > 2[n]$). Merging i and j makes them lose their outgoing non-neighbors not in common, thus they lose at least one outgoing non-neighbor, which is once again impossible.

Finally, f must be the identity so that $n = m$. □

Corollary 3.46. *EPI and SISO are not Well-Quasi-Orders.*

Proof. Just remark that an antichain for SEPI is also an antichain for EPI and SISO. □

This shows that, despite the similarity of the operations, the SEPI and minor relations are fundamentally different.

An important corollary is that there is no way to encode SEPI into minor as an order morphism. We mean this in the following way:

Proposition 3.47. *There is no function $c : \mathcal{G} \longrightarrow \mathcal{G}$ such that $G \to^*_{md} G' \Leftrightarrow c(G) \to_{minor} c(G')$.*

Proof. Suppose there is such a c. Let G_n be the infinite SEPI antichain in the proof of proposition 3.45. Then $(c(G_n))_{n\geq 5}$ is an infinite antichain for the minor relation, which is impossible [57]. □

This does not mean that we cannot encode SEPI problems into minor problems: to do this, we only need a function $c' : \mathcal{G} \times \mathcal{G} \longrightarrow \mathcal{G} \times \mathcal{G}$ such that $(G, G') \in$ SEPI iff $c'((G, G')) \in$ minor relation. This is easy: just map SEPI to one pair (H, H') in the minor relation and $\mathcal{G} \times \mathcal{G} \setminus$ SEPI to one pair out of the minor relation. The SEPI problem would be trivial to decide. However, it is NP-complete, and the goal of this encoding question is to transfer polynomial algorithms that can sometimes be achieved for minor decisions to the SEPI case. So the real question is: is there an encoding c' that is computable in polynomial time? This remains to be answered.

Part II

Subgraph Epimorphism: Computational Aspects

Chapter 4

The SEPI Decision Problem

In order to use the SEPI formalism on our application, we need some algorithms to decide SEPI comparability and compute SEPI glb/lub.

However, the induced subgraph isomorphism problem, which is the decision problem related to SISO, is known to be NP-complete [28]. The decision problem related to EPI is also NP-complete [68].

To obtain a similar result for SEPI, we cannot use the pointed graph encoding of Prop. 3.38 and the NP-completeness result for EPI: indeed, we would need an NP-completeness result for EPI restricted to pointed graphs. Moreover, we also need a complexity result for SEPI in the bipartite graphs case, which are of interest to our application.

In this chapter, we give short proofs of NP-completeness for the SEPI decision problem in the general and in the bipartite case. This result is one of the contributions of [29].

4.1 NP-completeness for general graphs

Since the SISO and EPI decision problems are NP-complete, computing d_{SISO} or d_{EPI} is NP-hard, indeed, $d_o(G, G') = |G| - |G'|$ if and only if $G \to_o^* G'$, by Property 3.32 and Theorem 3.40.

This is also the case for SEPI. Let us first define the problem formally:

Definition 4.1 (SEPI decision problem)**.** *The* subgraph epimorphism problem *is the decision problem:*

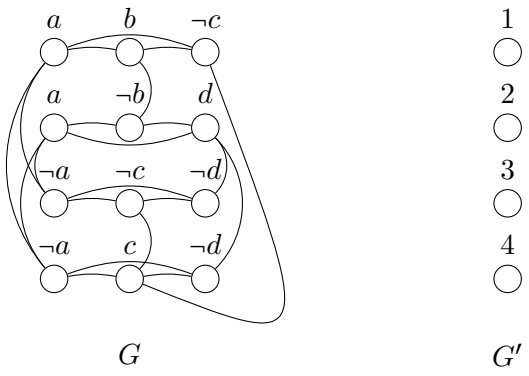
Instance: *Two graphs G, G'.* Question: $G \overset{SEPI}{\rightsquigarrow} G'$ *?*

NP-completeness of the SEPI decision problem can be proved by reduction of SAT [16]:

Theorem 4.2. *The subgraph epimorphism problem is NP-complete.*

The following proof was found by Thierry Martinez.

Figure 4.1: Reduction from the SAT instance $\phi = (a \vee b \vee \neg c) \wedge (a \vee \neg b \vee d) \wedge (\neg a \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee \neg d)$ to SEPI



Proof. The subgraph epimorphism problem is NP: given graphs $G = (V, A)$, $G' = (V', A')$ and partial function $f : V \to V'$, checking whether f is a subgraph epimorphism or not can obviously be done in polynomial time.

Let ϕ be a boolean formula in conjunctive normal form: $\phi = c_1 \wedge \cdots \wedge c_n$, $c_i = \ell_{i,1} \vee \cdots \vee \ell_{i,n_i}$, and $\ell_{i,j} \in X \cup \neg X$, with $X = \{x_1, \ldots, x_m\}$ and $\neg X = \{\neg x_1, \ldots, \neg x_m\}$.

Let $C_\phi = \{i \in \mathbb{N} \mid 1 \leq i \leq n\}$ and $L_\phi = \left\{(i,j) \in \mathbb{N}^2 \mid 1 \leq i \leq n, 1 \leq j \leq n_i\right\}$. Let $G' = (C_\phi, \varnothing)$, and

$$G = \left(L_\phi, \left\{\left((i,j),(i',j')\right) \subseteq L_\phi \mid (i = i' \wedge j \neq j') \vee \ell_{i',j'} = \neg \ell_{i,j}\right\}\right)$$

This construction is depicted in Fig. 4.1.

We now prove that ϕ satisfiable $\Leftrightarrow G \overset{SEPI}{\rightsquigarrow} G'$:

$\Rightarrow$ Suppose ϕ is satisfiable, and let ν be a valuation that satisfies it. Then $\forall i, \exists j, \ell_{i,j}$ is satisfied in valuation ν. Let j_i be the minimal such j for every i, and let $K = \{(i, j_i) \in L_\phi\}$.

Since ν is a valuation, it satisfies at least a literal per clause, so the first projection $\pi_1 : \left((i,j) \in L_\phi \longrightarrow i \in C_\phi \text{ if } (i,j) \in K\right)$ is surjective.

There are no arcs in the subgraph induced by K: first there cannot be any distinct (i, j_1) and (i, j_2) in $K \times K$ since there is only one minimal j_i, then since ν is a valuation, it cannot contain both literals ℓ and $\neg\ell$.

So π_1 is a subgraph epimorphism from G to G'.

$\Leftarrow$ Suppose there is a subgraph epimorphism μ is a SEPI from G to G'. Let $K = \mu^{-1}(C_\phi)$, and $\nu = \{\ell_{i,j} \mid (i,j) \in K\}$. Because of the structure of G', there can be no arcs in $K \times K$. So there are no $\ell \in K$ such that $\neg\ell \in K$, which means K is a valid valuation.

Since μ is surjective, there are at least n elements in K. However, there cannot be two distinct elements (i, j_1) and (i, j_2) in K, or the arc $((i, j_1), (i, j_2))$ would be in the subgraph induced by K. So there is exactly one j_i for every i, which means ν satisfies clause c_i with literal j_i, hence ϕ is satisfiable. □

It is worth noticing that in this proof, $G \overset{SEPI}{\rightsquigarrow} G' \Leftrightarrow G \overset{SISO}{\rightsquigarrow} G'$, so that this reduction shows the NP-hardness of both SEPI and SISO.

4.2 NP-completeness for reaction graphs

In our application, we consider bipartite graphs with labeled vertices, which we call reaction graphs. Even with this additional structural constraint, the SEPI problem can be proved NP-complete. Let us define the terms first:

Definition 4.3 (Reaction Graph)**.** *A* reaction graph G *is a triple* $G = (V, A, t)$*, where* $t : V \longrightarrow \{s, r\}$ *labels the type of vertices:* $S = t^{-1}(s)$ *is the set of* species *vertices,* $R = t^{-1}(r)$ *is the set of* reaction *vertices, and* $A \subseteq S \times R \bigcup R \times S$*.*

It is equivalent to write a reaction graph by separating species and reaction: we will also write (S, R, A)*.*

When considering reaction graphs, merge operations can only be done on vertices of the same type. This leads to a specialized definition of subgraph epimorphisms:

Definition 4.4 (Subgraph Epimorphism on reaction graphs)**.** *A subgraph epimorphism on reaction graphs from* $G = (V, A, t)$ *to* $G' = (V', A', t')$ *is a subgraph epimorphism from* (V, A) *to* (V', A') *that preserves labeling, i.e* $\forall u$ *s.t.* $f(u)$ *is defined,* $t'(f(u)) = t(u)$*.*

We can state the main result:

Theorem 4.5. *The subgraph epimorphism for reaction graphs is NP-complete.*

The following proof was found by Christine Solnon.

Proof. We reduce the exactly-k-set-covering problem to SEPI. This problem is defined by:
Instance. A set E, subsets $U \subseteq \mathcal{P}(E)$, and an integer $k \leq |U|$.
Question. Is there a family $U^* \subseteq U$ such that $|U^*| = k$ and $\bigcup_{s \in U^*} s = E$?

The more well-known k-set-covering problem asks for $|U^*| \leq k$, it is obvious that the variant we use here is also NP-complete.

Let (E, U, k) be an instance of exactly-k-set-covering, $E = \{e_1 \dots e_n\}$ and $U = \{u_1 \dots u_m\}$.

Let $G = (S, R, A)$ with $S = \{u_1 \dots u_m\} \cup \{e_1 \dots e_n\}$, $R = \{r_1 \dots r_m\}$, and $A = \{(u_i, r_i) \mid 1 \leq i \leq m\} \cup \{(r_i, e_j) \mid e_j \in u_i\}$.

Let $G' = (S', R', A')$ $S' = \{s'_1 \dots s'_m\} \cup \{e_1 \dots e_n\}$, $R' = \{r'\}$, and $A' = \{(s'_i, r') \mid 1 \leq i \leq k\} \cup \{(r', e_j) \mid 1 \leq j \leq n\}$.

We prove that the exactly-k-set-covering problem has a positive answer iff there is a reaction graph SEPI from G to G'.

$\Rightarrow$ Suppose $U^* = \{u_1^* \dots u_k^*\}$ is a solution of (E, U, k). Let $U \setminus U^* = \{u_{k+1}^* \dots u_m^*\}$. Then $\mu : u_j = u_i^* \longrightarrow s_i', r_i \longrightarrow r', e_i \longrightarrow e_i$ is a reaction graph SEPI from G to G'.

$\Leftarrow$ Suppose there is a reaction graph SEPI μ from G to G'. Then, with $\mu^{-1}(r') = \{r_{i_1} \dots r_{i_k}\}$ and $U^* = \{u_{i_1} \dots u_{i_k}\}$, U^* is a solution of the covering problem.

First, U^* covers E. Indeed, let $e \in E$. Then the arc $(r', \mu(e))$ is covered in G', by some arc (r_i, e_1) in G. $r_i \in \mu^{-1}(r')$, but is e_1 always the vertex e? Yes: since there are the same number of s-vertices in G and G' $(n+m)$, μ induces a bijection between s-vertices of G and G', so $e_1 = e$. Which proves that e is covered by the subset corresponding to r_i.

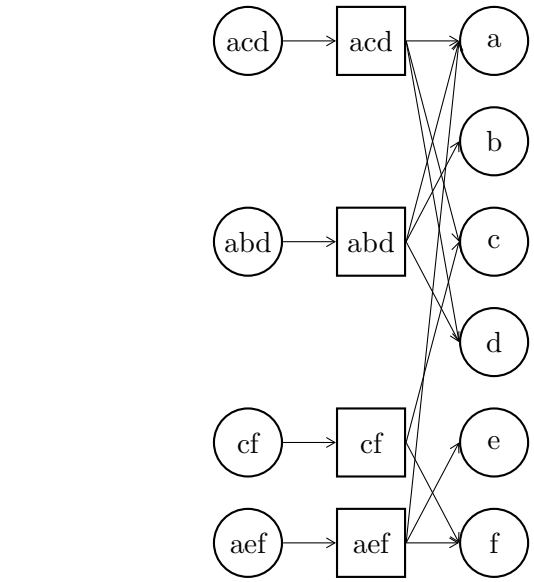
Next, U^* has k elements. Vertex types force arcs (u_i, r_i) to be the only ones that can cover the (s_i', r'). So the preimages of the s_i' are some u_i, and there must be exactly k such u_i because of the bijection on s-vertices. U^* is exactly those u_i.

The coding being clearly polynomial, this concludes the proof of reduction from exactly-k-set-covering to the reaction graph SEPI decision problem, and the proof of NP-completeness. □

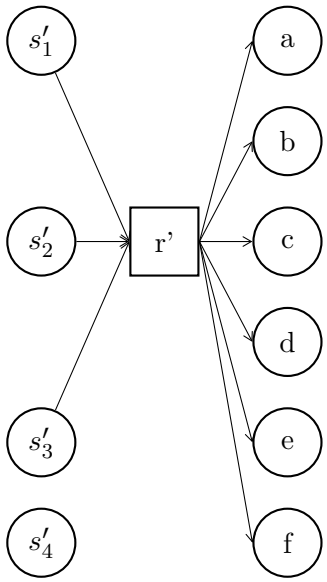
Example 4.6. *Let us consider an instance of the set covering problem* (E, U, k) *such that* $E = \{a, b, c, d, e, f\}$ *and* $U = \{\{a, c, d\}, \{a, b, d\}, \{c, f\},$
$\{a, e, f\}\}$. *Fig. 4.2(a) displays the corresponding source graph, fig. 4.2(b) and the target graph corresponding to* $k = 3$, *and 4.2(c) the target graph corresponding to* $k = 2$.

The graph 4.2(a) may be transformed into graph 4.2(b) by deleting the r-vertex associated with $\{a, c, d\}$ *and by merging the three other r-vertices, respectively associated with* $\{a, b, d\}$, $\{c, f\}$ *and* $\{a, e, f\}$. *This corresponds to the solution of the set covering problem such that the 3 selected subsets are* $\{a, b, d\}$, $\{c, f\}$ *and* $\{a, e, f\}$.

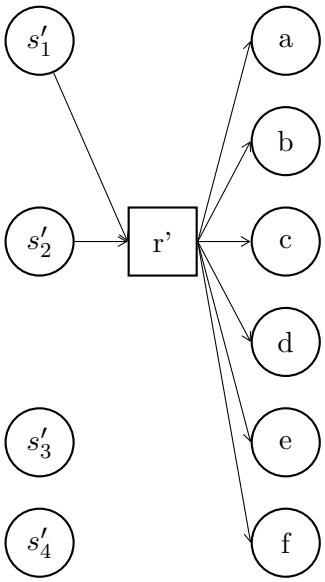
However, graph 4.2(a) cannot be transformed into graph 4.2(c), as the set covering problem instance has no solution for $k = 2$.



(a)



(b)



(c)

Figure 4.2: Reaction graphs for $E = \{a, b, c, d, e, f\}$ and $U = \{\{a, c, d\}, \{a, b, d\}, \{c, f\}, \{a, e, f\}\}$. s-vertices are displayed in circles, r-vertices in squares. (a) the source graph G; (b) the target graph G' defined for $k = 3$; (c) the target graph G' defined for $k = 2$.

Chapter 5

Constraint Programming Model

We have shown that the SEPI decision problem is NP-complete. It means that solving it may be, in the worst case, done in exponential time.

In this chapter, we present a *constraint programming* approach to the SEPI decision problem. Graph matching problems can be easily modeled as constraint satisfaction problems [42], and in this context, constraint programming has experimentally proven to be as competitive as, and in some cases to outperform, dedicated approaches such as Vflib [70, 64]. The model we present here is a refinement of the one used in our first publication about the subject in [31].

In the next chapter, we will present a *SAT solver*-based approach to the SEPI decision problem, and to SEPI glb and lub computations. The performance of these methods is evaluated in chapter 9.

5.1 Constraint Programming

The SEPI decision problem may be NP-complete, this does not mean that our instances are intractable in practice, even for large sizes.

The naive approach to NP-complete problems is known as *generate-and-test*: this algorithm searches for an answer exhaustively by an iteration that generates a candidate and tests whether it satisfies the problem; a candidate that satisfies the problem is a witness to the satisfiability, if no witness is found, the exhaustivity of the algorithm guarantees that the decision problem is not satisfiable. This is a naive example of a complete algorithm.

Depending on available computing power, there is an upper bound on the size of instances that such algorithms cannot overcome. The goal of complete methods is to use the structure of the problem to delay this intractability bound to higher sizes, and/or remove it for some families of instances.

Instead of *generate-and-test*, constraint programming uses *constrain-and-search*. This approach works in a fashion similar to a greedy algorithm: it iterates from a partial instan-

tiation and tries to build a solution to the problem incrementally. The approaches differ in choice and commitment: a greedy algorithm chooses the best way to extend the partial instantiation locally at each iteration and commits to this choice, a constraint program removes parts of the search space where finding a witness is impossible, splits the search space and iteratively searches for an extension inside every subspace.

In the worst case, with no filtering, the search procedure is equivalent to generate-and-test. In the best case, the combination of filtering and search heuristics yield a solution on the first try, i.e. never backtracks.

To remove parts of the search space, constraint programming deduces information from the structure of the problem to solve, the problem has to be described as a constraint satisfaction problem:

Definition 5.1 (CSP)**.** *A* constraint satisfaction problem *(CSP for short) is a triple* $\mathcal{P} = (V, D, C)$*, where:*

- V *is a set of variables, the* decision *variables.*
- D *is a family of domains indexed by variables from* V*:* $\forall X \in V$*,* D_X *is a finite set, the* domain *of* X*.*
- C *is a set of constraints, each* $c \in C$ *is defined by its arity* $\text{arity}(c) \in \mathbb{N}$*, a tuple of variables* $\vec{X}(c) \in V^{\text{arity}(c)}$*, and a relation* $R(c) \subseteq \prod_{i=1}^{\text{arity}(c)} D_{\vec{X}_i(c)}$*.*

A noticeable point is that CSPs formulate decision problems as conjunctions of elementary problems, the constraints.

A particular formulation of a decision problem as a CSP is an *encoding* of the problem. There may be several encodings of the same problem, with various properties.

Example 5.2. *The existence of an injective function* $f : A \to B$ *can be encoded with:*

- *Variables* X_a *with* $a \in A$
- *Domains* $D_{X_a} = B$ *for every* $a \in A$
- *For every distinct* $a, a' \in A$*, constraints* $c_{a,a'}$ *usually written* $X_a \neq X_{a'}$*, with*
  - $\text{arity}(c_{a,a'}) = 2$
  - *Variables* $(X_a, X_{a'})$
  - *Relation* $\{(b, b') \in B^2 \mid b \neq b'\}$

Example 5.3. *The existence of an injective function* $f : A \to B$ *can also be encoded with:*

- *Variables* X_a *with* $a \in A$
- *Domains* $D_{X_a} = B$ *for every* $a \in A$

- *A constraint c with*
  - $\text{arity}(c) = |A|$
  - *Variables* $(X_{a_1}, \ldots, X_{a_{|A|}})$
  - *Relation* $\{(b_1, \ldots b_{|A|}) \in B^{|A|} \text{ s.t. } |\{b_1 \ldots b_{|A|}\}| = |A|\}$

One formulation decomposes the problem in many constraints, and the other has a monolithic constraint at its core; still, they are equivalent, i.e. they have the same set of solutions:

Definition 5.4 (Solution of a CSP)**.** *An assignment $\eta : X \in V \longrightarrow \eta(X) \in D_X$ is a solution of $\mathcal{P}$ when $\forall c \in C, (\eta(X_1), \ldots, \eta(X_n)) \in R(c)$, with $\vec{X}(c) = (X_1, \ldots, X_n)$.*

If a CSP has a solution, it is called satisfiable*, if it has none, it is* unsatisfiable*.*

We shall use the following symbolic constraints:

- $\texttt{element}(I, F, Y)$ constrains a list of *variables* F describing a function f from domain $1 \ldots |F|$ to $\mathbb{N}$ to have Y as the image of I. The constraint $c = \texttt{element\_var}(I, F, Y)$, where $I, Y \in V$ and $F = (F_1, \ldots, F_n) \in V^n$, is defined by
  - $\text{arity}(c) = n + 2$
  - $\vec{X}(c) = (X, F_1, \ldots, F_n, Y)$
  - $R(c) = \{(i, \eta_1, \ldots, \eta_n, y) \in D_X \times (\prod_{i=1}^{n} D_{F_i}) \times D_Y \mid \eta_i = y\}$
- $A = a \Rightarrow B = b$ conditions the value of B on the value of A.
  - $\text{arity}(c) = 4$
  - $\vec{X}(c) = (A, B)$
  - $R(c) = (\mathbb{N} \times \mathbb{N}) \setminus (\{a\} \times \{y \in \mathbb{N} \mid y \neq b\})$
- $A = a \Rightarrow B \leq b$ also conditions the value of B on the value of A.
  - $\text{arity}(c) = 4$
  - $\vec{X}(c) = (A, B)$
  - $R(c) = (\mathbb{N} \times \mathbb{N}) \setminus (\{a\} \times \{y \in \mathbb{N} \mid y > b\})$

5.2 A CP model for SEPI

In this section, we expose a CP model for the existence of a SEPI from G to G' ; the choices made in this modelisation are explained in the next section. First, we show the CSP model, then the strategy used to solve it.

5.2.1 Constraint Model

To differentiate mathematical variables and CP variables, we write CP variables in bold font (as in $\mathbf{X}$ opposed to X) ; $[a, b, c]$ denotes the list of the three elements a, b, c ; π_1 and π_2 are the first and second projection functions.

Let G and G' be two graphs, with $G = (V, A)$, $G' = (V', A')$, and $V = \{v_1 \dots v_n\}$, $A = \{a_1 \dots a_k\}$, $V' = \{v'_1 \dots v'_{n'}\}$, $A' = \{a'_1 \dots a'_{k'}\}$, $A'_\perp = A' \cup \{(x, y) \in (V' \cup \{\perp\})^2 | x = \perp \vee y = \perp\} = \{a'_1 \dots a'_{k'}, a'_{k'+1} \dots a'_{k'_\perp}\}$.

The following CSP is a model for the existence of a SEPI from G to G':

Variables. Variables are associated to the vertices and arcs of G and G':

- Morphism variables
  - $\mathbf{X}_v$ for $v \in V$, with $D(\mathbf{X}_v) = V' \cup \perp$.
  - $\mathbf{A}_a$ for $a \in A$, with $D(\mathbf{A}_a) = \{1, \dots, |A'_\perp|\}$.
- Antecedent variables
  - $\mathbf{X'}_{v'}$ for $v' \in V'$, with $D(\mathbf{X'}_{v'}) = V$.
  - $\mathbf{A'}_{a'}$ for $a' \in A'$, with $D(\mathbf{A'}_{a'}) = A$.

Constraints . The role of morphism and antecedent variables is enforced with the following constraints:

I. Morphism constraints

i. $\forall a \in A, \texttt{element}(\mathbf{A}_a, [\pi_1(a'_1) \dots \pi_1(a'_{k'_\perp})], \mathbf{X}_{\pi_1(a)})$

ii. $\forall a \in A, \texttt{element}(\mathbf{A}_a, [\pi_2(a'_1) \dots \pi_2(a'_{k'_\perp})], \mathbf{X}_{\pi_2(a)})$

II. Minimal antecedent constraints

i. $\forall v \in V, \forall v' \in V', \mathbf{X'}_{v'} = v \Rightarrow \mathbf{X}_v = v'$

ii. $\forall v \in V, \forall v' \in V', \mathbf{X}_v = v' \Rightarrow \mathbf{X'}_{v'} \leq v$

iii. $\forall a \in A, \forall a' \in A', \mathbf{A'}_{a'} = a \Rightarrow \mathbf{A}_a = a'$

iv. $\forall a \in A, \forall a' \in A', \mathbf{A}_a = a' \Rightarrow \mathbf{A'}_{a'} \leq a$

III. Global surjection constraints

i. $\texttt{gsurjection}([\mathbf{X}_{v_1} \dots \mathbf{X}_{v_n}], V')$

ii. $\texttt{gsurjection}([\mathbf{A}_{a_1} \dots \mathbf{A}_{a_k}], A')$

This model uses reified constraints and the `element` constraint.

It also uses a constraint `gsurjection` that has a list of variables and a list of values as arguments. It constrains the list of variables to contain at least once each value in the second list, enforcing a global surjection.

Proposition 5.5. *The CP model $\mathcal{P}$ associated to graphs G, G' has a solution if and only if there exists a subgraph epimorphism from G to G'.*

5.2.2 Search Strategy

Although the search strategy was not the main focus of this work, we did try different variants. The best we found is to enumerate first the antecedent variables for the arcs, $\mathbf{A'}_{a'}$, then the antecedent variables for the vertices, $\mathbf{X'}_{x'}$, and finally the morphism variables.

The strategy uses a min to max value order, which is standard but proved more efficient than the first fail heuristics.

5.3 Modelling Choices

As asserted in 5.1, a decision problem can be modeled in different manners, and different models have different characteristics. Of course, computation time is our first worry here, with composability and simplicity coming in second. In this section, we explain how the model was built.

Keep in mind that this model is shown without an assumption to be the best possible. Unlike more classic algorithms, constraint programming does not allow us to give absolute guarantees on the performance of a program on an instance. However, techniques that have become classics in the field can be used to make models that prove better than other models. The following explains where this know-how has been used in the model.

5.3.1 SEPI as an assignment

As stated in the first part of this text, there is an equivalence between strings of merge/delete, subgraph epimorphisms and graph quotients. Thus, a CP model could be based on any of these three formalisms.

SEPI vs String of Operations. The advantage of a SEPI model over a string of operations is straightforward: the symmetries of merge and delete operations shown in chapter 2 are completely flattened by using a single function for all operations.

Knowing this, there may be another way to break symmetries in a string of operations model, e.g. by putting deletions before merging and ordering operations in lexicographical order. This is more complicated than using an assignment from source vertices to target vertices, where symmetry breaking is canonically in the representation.

SEPI vs Graph Quotient. A graph quotient model would be based on computing α such that $G_\perp/\alpha \sim G'_\perp$.

Representing an isomorphism between $G_\perp/\alpha$ and $G'_\perp$ is already representing a morphism, so why not represent the SEPI directly? Still, there could be an advantage in representing an equivalence relation, since it would allow one to reason on informations such as $\mu(a) \neq \mu(b)$ or $\mu(a) = \mu(b)$.

This could be a refinement of the CP model. However, for such (dis)equality information to prove useful, there should be some additional constraint that uses it, or the search strategy should somehow take advantage of the supplementary variables.

To compare, this model knows that $\mu(a) \neq \mu(b)$ only when their domain is disjoint, e.g. when $\mu(a) = a'$ and $a' \notin \mathrm{dom}(\mu(b))$.

5.3.2 Dual Modeling

Surjection is a constraint that does not have a domain-arc-consistent propagator in the model, because our CP framework, GNU-Prolog does not have any. As such, it does more checking than propagating, and since the surjection constraints affects most variables of the CSP, checking can be very punitive, i.e., it can force backtracks to occur deep in the search tree. This is dealt with in two steps: in the encoding of surjection, and then in the search strategy.

We encode surjection with *dual variables* channeled with constraints IIi and IIiii. This forces every value to have an antecedent by μ, hence μ has to be a surjection.

However, since a value can have several antecedents in a given solution, this encoding introduces representation symmetries. Constraints IIii and IIiv break these representation symmetries by choosing minimal antecedents.

This minimal antecedents encoding both improves performance and contrary to a simple antecedent encoding, ensures that a given surjection can appear at most once in the list of solutions.

5.3.3 Search Strategy

In order to deal with the late-failure aspect of surjection checking, labeling antecedent variables first limits backtrackings due to surjection higher in the search tree. This effect can be formalized as follows:

Proposition 5.6. *The SEPI CP model yields a solution iff variables* $(\mathbf{X'}_{v'})_{v' \in V'}$ *and* $(\mathbf{A'}_{a'})_{a' \in A'}$ *can be successfully instantiated.*

Proof. Obviously, if enumerating antecedent variables fails, there is no SEPI from the source graph to the target graph.

Conversely, if the enumeration on antecedent variables succeeded, then the corresponding $(\mathbf{X}_v)_{v \in V}$ and $(\mathbf{A}_a)_{a \in A}$ have singleton domains, thanks to domain-arc-consistency of

`element` constraints II. The induced subgraph formed by the source vertices and arcs that correspond to these variables are sufficient to cover G', and the morphism constraints I ensure that the variables code a morphism. Giving the $\perp$ value for every remaining morphism variable yields a SEPI from the source graph to the target graph. □

Therefore in the CP model, the morphism variables need not be instantiated to decide the existence of a SEPI, only antecedent variables do. Morphism variables can be instantiated after antecedent variables, in order to compute a SEPI without further backtracking.

5.3.4 Global surjection constraint

Global constraints are a powerful tool that can be used to get better deductions by considering global properties of the CSP. In the SEPI CP model, we implemented a constraint `gsurjection` that uses cardinality reasoning to detect cases where a partially instantiated assignment that cannot cover its codomain.

This global propagator prunes more than arc-consistent propagators on the antecedent variables, leading to performance improvements.

The cardinality reasoning is based on the fact there has to be enough variables to cover target values. More formally, let $\mathbf{D} = [\mathbf{D}_1 \dots \mathbf{D}_d]$ be a list of variables and $T = [T_1 \dots T_t]$ of sorted list of integers.

Definition 5.7 (Covered values, Committed variables)**.**

$$\mathit{covered}(\mathbf{D}, T) = \{T_i \mid \exists j, \mathit{dom}(\mathbf{D}_j) = \{T_i\}\}$$

$$\mathit{committed}(\mathbf{D}, T) = \{\mathbf{D}_j \mid \mathit{dom}(\mathbf{D}_j) \subseteq \mathit{covered}(\mathbf{D}, T)\}$$

Informally, $\mathit{covered}(\mathbf{D}, T)$ is the set of elements of T that are taken by at least one of the variables in $\mathbf{D}$, and $\mathit{commited}(\mathbf{D}, T)$ is the set of variables which cannot cover any uncovered variable.

Then $\texttt{gsurjection}(\mathbf{D}, T)$ enforces $|T| - |\mathit{covered}(T)| \leq |\mathbf{D}| - |\mathit{commited}(\mathbf{D})|$ by failing when this is not the case.

As a consequence, when all $\mathbf{D}_j$ are ground, $\mathbf{D}$ has to be a surjection on the elements of T.

Implementing a propagator with a linear number of domain operations (union, intersection, complement) for this constraint is straightforward: in a first pass, accumulate covered elements in a domain, then count the number of uncommitted variables, i.e. the variables whose domain is not a subset of the covered values.

While these constraints are *redundant* with the minimal antecedent coding, using both minimal antecedents to guide the search and a global constraint to detect failures earlier leads to performance improvement.

Note that there was an alternative to this global constraint: using a domain-arc-consistent `alldifferent` constraint on dual variables has even more pruning power than

`gsurjection`. In a prototype, this proved to be an efficient encoding. However, such a constraint was not available in our framework GNU-Prolog, and it is more complicated to implement than the linear propagator of `gsurjection`: the simplest way to take advantage of `alldifferent` would be to change frameworks.

5.3.5 Channeling constraints

In earlier implementations, morphism constraints where encoded using the table `relation` constraint of GNU-Prolog, so that $\mathbf{A_a}$ variables where not available to channel antecedent constraints. A more complex channeling had to communicate information between morphism and antecedent constraints.

However, in GNU-Prolog, `relation` is internally coded as `element` constraints, which means that there were pairs of variables, one in the channeling and the other inside `relation` , that always had the same value in solutions!

Decomposing `relation` as two `element` constraints makes the $\mathbf{A_a}$ available, allowing to share these variables between morphism and antecedent constraints. This leads to not only simpler channeling, but also performance improvement, since information propagation is effectively rerouted in a single variable. This effect has more than a simple twofold computation time improvement, most likely due to chaotic propagation effects.

5.3.6 Other considerations

Interestingly, the $\perp$ vertices used to reduce SEPI to EPI in the proof of Proposition 3.38 are also used in this coding of subgraph epimorphism as a CSP: without the dummy vertices, we would have a CSP encoding of the EPI problem.

Note also that our application has reaction graphs, which are bipartite graphs with vertices separates as either species or reaction. In order to specialize the CP model to reaction graphs, the domains of reaction vertex variables can be restricted to reaction vertices and species to species.

Chapter 6

Boolean Models of SEPI

Coding problems into SAT instances and using a SAT solver to find whether they are satisfiable or not is another successful approach to solve NP-complete problems. We presented this approach in a specialized workshop [30].

In this chapter, we present CNF (Conjunctive Normal Form) encodings of the SEPI problem and of the decision variants of the SEPI glb (resp. lub) problems, i.e. the existence of a graph of bounded size in $G \cap_{md} G'$ (resp. $G \cup_{md} G'$).

A SAT instance can be described as a pair (X, C), where X is a set of variables, and C is a set of clauses $c_1 \dots c_r$ with $c_i = \bigvee l_{i,j}$, and finally $l_{i,j}$ is either x or $\bar{x}$, with $x \in X$. A SAT instance can be described more shortly as a boolean formula in conjunctive normal form (CNF).

The SAT framework is similar to the CP framework: SAT "variables" have only two values and SAT "constraints" can only be disjunctions of literals. In the most successful SAT framework (Conflict Driven Clause Learning solvers), deduction only comes from a mechanism called *unit propagation*.

The principle of unit propagation is the following: when a clause has only one literal l_i, then this literal has to be true for the clause to be true. The corresponding variable v_i is set accordingly, and the whole CNF is rewritten with v_i to its new value. This may shorten clauses to length one, in which case unit propagation is re-triggered. We may take advantage of this mechanism in our models, by choosing encodings that have smaller clauses, and adding redundant clauses expressing deductions that would not be trivial for the algorithm.

The success of the CDCL framework comes from learning mechanisms that we will not explicitly take advantage of. Moreover, even simple modifications such as changing the order in which clauses are passed to the solver have an influence on computation time, but since they depend heavily on implementation details of the solver, we will not rely on them.

The boolean formulae given in this chapter are transformed into clauses using an obvious

normalization procedure: implications $a \Rightarrow (b \wedge c)$ are broken into $a \Rightarrow b$ and $a \Rightarrow c$, implication $a \Rightarrow b$ is coded in $\neg a \vee b$; no further transformations are done. We write $\mathrm{cl}(f)$, where f is a boolean function, to denote the clauses passed to the SAT solver.

6.1 CNF encoding of the SEPI Decision Problem

In this section, we will describe, for a given SEPI problem (G, G'), an encoding of the problem into boolean clauses. This encoding has been implemented, and the evaluation will be made in the next section.

We split the description of the coding into two main parts: first how to code a partial surjective function, then adding graph constraints to code a subgraph epimorphism.

6.1.1 Partial Surjective Function Coding

A SEPI m from G to G' is also a partial surjective function from V to V'.

Definition 6.1 (Partial Surjective Function)**.** *A binary relation $R \subseteq E \times E'$ is a* partial surjective function *if the following conditions are fulfilled:*

- $\forall x \in E, x'_1 \in E', x'_2 \in E', ((x, x'_1) \in R \wedge (x, x'_2) \in R) \Rightarrow x'_1 = x'_2$
- $\forall x' \in E', \exists x \in E, (x, x') \in R$

The elements $x \in E$ do *not* have to be covered by some $(x, x') \in R$, hence the qualifier *partial*; we write $R(x) = x'$ when $x \in E$ is covered by x', $R(x) = \perp$ when x is not covered.

Variables. m is encoded as a binary relation on $V \times (V' \cup \{\perp\})$. The elements of $V' \cup \{\perp\}$ are put in a total order $v'_0 = \perp < v'_1 < \ldots < v'_{n'}$.

- $\forall (v, v') \in V \times (V' \cup \{\perp\}), \mathbf{m}_{v,v'} = 1$ iff $m(v) = v'$.
- $\forall (v, v') \in V \times (V' \cup \{\perp\}), \mathbf{m}^{<}_{v,v'} = 1$ iff $m(v) < v'$.

Clauses. The following clauses are used to force variables to match their description:

I. Left Totality. $\forall v \in V$, $\mathrm{cl}(\bigvee_{v' \in V' \cup \{\perp\}} \mathbf{m}_{v,v'})$

II. Functionality. $\forall (v, v'_j) \in V \times (V' \cup \{\perp\})$,

   i. $\mathrm{cl}(\mathbf{m}_{v,v'_j} \Rightarrow \mathbf{m}^{<}_{(v,v'_{j+1})})$

   ii. $\mathrm{cl}(\mathbf{m}^{<}_{v,v'_j} \Rightarrow \mathbf{m}^{<}_{(v,v'_{j+1})})$

   iii. $\mathrm{cl}(\mathbf{m}^{<}_{v,v'_j} \Rightarrow \neg \mathbf{m}_{(v,v'_j)})$

III. Right Totality. $\forall v' \in V'$, $\mathrm{cl}(\bigvee_{v \in V} \mathbf{m}_{v,v'})$

The function is encoded as a matrix of booleans: this is the so-called direct encoding. An alternative would be to code the image of each vertex as a number in binary form, but such encodings are not especially beneficial when the codomain of the function is small, which is the case in our application; binary encodings are also better suited to arithmetic constraints, but ours are symbolic constraints.

Functionality could be encoded straightforwardly as well, with something like:
$\forall (v, v'_1, v'_2) \in V \times (V' \cup \{\bot\})^2$ with $v'_1 \neq v'_2, \mathrm{cl}(\neg(\mathbf{m}_{v,v'_1} \wedge \mathbf{m}_{v,v'_2}))$

This encoding has $|V| \cdot |V' \cup \{\bot\}|^2$ clauses, which is a problem in practice. The coding above, so-called ladder encoding, achieved by using the order on $|V' \cup \{\bot\}|$ to force the image of $v \in V$ to be minimal, only has $O(|V| \cdot |V' \cup \{\bot\}|)$ clauses. See [7] for general information on encodings.

6.1.2 Subgraph Epimorphism Coding

Let us build on the previous part to constrain the function to represent a SEPI.
Variables. Additional variables are used to constrain SEPI:

- Non deleted arcs. $\forall (a, a') \in A \times A', \mathbf{m}_{a,a'}$ iff $m(a) = a'$.
- Deleted arcs. $\forall a \in A, \mathbf{is\_dummy}(\mathbf{m}_a) = 1$ iff $m(a) = \bot$

Clauses.

I. Left Totality on Arcs. $\forall a \in A$, $\mathrm{cl}(\mathbf{is\_dummy}(\mathbf{m}_a) \vee \bigvee_{a' \in A'} \mathbf{m}_{a,a'})$

II. Right Totality on Arcs. $\forall a' \in A'$, $\mathrm{cl}(\bigvee_{a \in A} \mathbf{m}_{a,a'})$

III. Graph Morphism. $\forall (u, v) \in A, (u', v')) \in A'$,

  i. $\mathrm{cl}(\mathbf{m}_{(u,v),(u',v')} \Rightarrow \mathbf{m}_{u,u'})$
  ii. $\mathrm{cl}(\mathbf{m}_{(u,v),(u',v')} \Rightarrow \mathbf{m}_{v,v'})$
  iii. $\mathrm{cl}((\mathbf{m}_{v,v'} \wedge \mathbf{m}_{v,v'}) \Rightarrow \mathbf{m}_{(u,v),(u',v')})$

IV. Subgraph Morphism. $\forall (u, v) \in A$,

  i. $\mathrm{cl}(\mathbf{is\_dummy}(\mathbf{m}_{(u,v)}) \Rightarrow \mathbf{m}_{u,\bot} \vee \mathbf{m}_{v,\bot})$
  ii. $\mathrm{cl}(\mathbf{m}_{u,\bot} \Rightarrow \mathbf{is\_dummy}(\mathbf{m}_{(u,v)}))$
  iii. $\mathrm{cl}(\mathbf{m}_{v,\bot} \Rightarrow \mathbf{is\_dummy}(\mathbf{m}_{(u,v)}))$

V. Redundant Morphism Propagation. $\forall (u, v) \in A, (u', v') \in (N' \times N') \setminus A'$,

  i. $\mathrm{cl}(\neg \mathbf{m}_{u,u'} \vee \neg \mathbf{m}_{v,v'})$

The encoding of the morphism constraint follows from the encoding of the function. Notice that the redundant morphism clauses V take advantage of unit propagation to remove forbidden values. Indeed, suppose the image of u is u' and $(u, v) \in A$. Without the redundant morphism clauses, unit propagation on clauses I and III can deduce the clause $\bigvee_{(u',v')\in A'} \mathbf{m}_{\mathbf{v},\mathbf{v}'}$, but no clause put the $\mathbf{m}_{\mathbf{v},\mathbf{v}'}$ for $v' \in (u', v') \in (N' \times N') \setminus A'$ to false, even though it would be correct from the logic point of view. Clauses V help the SAT solver deduce this correct inference, which speeds up the solving.

The model can be specialized to reaction graphs by restricting domains, i.e. by setting $\mathbf{m}_{v,v'}$ to false when v and v' are not of the same type.

6.1.3 Surjectivity and Sorting Networks

The `gsurjection` propagation idea can be imitated with boolean clauses. This improves performance a little. The idea is to introduce minimal antecedents, and then use cardinality networks to force the number of minimal antecedents to be greater than the number of targets.

Cardinality networks use boolean sorting networks to have some consistency using only unit clause propagation.

Since this is not central to the model and yields only small gains compared to the complexity required, we refer to publications that describe how sorting networks can be implemented and what can be gained from them. For a full exposition, [19] compares different approaches to coding integers in boolean clauses, [6] uses such networks on decompositions of cardinality-related constraints, [15] shows that Parberry's odd-even networks behave better than Batcher's merge networks for this purpose, [40] efficiently solves MAXSAT by coding the cardinality constraints with sorting networks coded in boolean clauses.

6.2 SAT SEPI GLB model

Since the SEPI decision problem is NP-complete, the SEPI glb decision problem (Is there a glb of size $\geq k$?) is also an NP-complete problem ; indeed the SEPI decision problem "Is there a SEPI from G_1 to G_2" can be encoded as "Is there a SEPI glb of G_1 and G_2 of size $|G_2|$?".

This is why we will use a SAT solver to solve SEPI glb instances.

6.2.1 SAT coding

Let G_1, G_2 be two graphs. The goal of this section is to formulate the existence of a graph G with k vertices or more such that $G_1 \to^*_{md} G$ and $G_2 \to^*_{md} G$.

Contrary to the SEPI SAT model, we do not know how many vertices there are, neither if vertices are species or reactions.

In order to overcome this, we model the graph G by a graph of variable size, and modify the coding of SEPI functions μ_1 from G_1 to G and μ_2 from G_2 to G. The size of G can go from lower bound k to $m' = \min(|G_1|, |G_2|)$.

First, we code a partial surjection on a set of size between k and m'. Then, we will take arcs into account.

Coding a Partial surjective function on a Set of Variable Size

As the coding is the same for μ_1 and μ_2, we describe the coding for μ_i.

Variables. μ_i is encoded as a binary relation on $V_i \times \{0..m'\}$.

- $\forall (v, v') \in V_i \times \{0 \dots m'\}, \mathbf{m}^i_{v,v'} = 1$ iff $\mu_i(v) = v'$.
- $\forall (v, v') \in V_i \times \{0 \dots m'\}, \mathbf{m}^{i,<}_{v,v'} = 1$ iff $\mu_i(v) < v'$.
- $\forall v' \in \{1 \dots m'\}, \mathbf{e}_{v'} = 1$ iff $v' \in V'$

Once again, the value $\mu_i(v) = 0$ means $\mu_i(v) = \perp$ or v is deleted by μ_i. Notice how left totality and functionality clauses take 0 into account ($v' \in \{0 \dots m'\}$) but covering and existing set clauses do not have to ($v' \in \{1 \dots m'\}$).

Clauses. The following clauses are used to force variables to match their description:

I. Left Totality. $\forall v \in V_i$, $\mathrm{cl}(\bigvee_{v' \in \{0 \dots m'\}} \mathbf{m}^i_{v,v'})$

II. Functionality. $\forall (v, v'_j) \in V_i \times \{0 \dots m'\}$,

   i. $\mathrm{cl}(\mathbf{m}^i_{v,v'_j} \Rightarrow \mathbf{m}^{i,<}_{(v,v'_{j+1})})$

   ii. $\mathrm{cl}(\mathbf{m}^{i,<}_{v,v'_j} \Rightarrow \mathbf{m}^{i,<}_{(v,v'_{j+1})})$

   iii. $\mathrm{cl}(\mathbf{m}^{i,<}_{v,v'_j} \Rightarrow \neg \mathbf{m}^i_{(v,v'_j)})$

III. Covering of Existing Vertices.
$\forall v' \in \{1 \dots m'\}, \mathrm{cl}(\mathbf{e}_{v'} \Leftrightarrow \bigvee_{v \in V_i} \mathbf{m}^i_{v,v'})$

IV. Existing Set Normalization.

   i. $\forall v' \in \{2 \dots m'\}, \mathrm{cl}(\mathbf{e}_{v'} \Rightarrow \mathbf{e}_{v'-1})$

   ii. $\forall v' \in \{1 \dots k\}, \mathrm{cl}(\mathbf{e}_{v'})$

Left totality does not change, and we use the ordering to express functionality in quadratic size again.

Right totality is modified: vertices are covered iff they exist. This is where communication between the two functions occur: when the covering clauses of both functions are put together, these clauses read $\bigvee_{v\in V_1} \mathbf{m}^1_{v,v'} \Leftrightarrow e_{v'} \Leftrightarrow \bigvee_{v\in V_2} \mathbf{m}^2_{v,v'}$. Describing $\bigvee_{v\in V_1} \mathbf{m}^1_{v,v'} \Leftrightarrow \bigvee_{v\in V_2} \mathbf{m}^2_{v,v'}$ directly would lead to a linear number of clauses of linear size, amounting to quadratic size. The solution of using $\mathbf{e}_{v'}$ variables uses a linear number of clauses of constant size ($\mathbf{m}^i_{v,v'} \Leftarrow \mathbf{e}_{v'}$) and 2 clauses of linear size ($\mathbf{e}_{v'} \Leftarrow \bigvee_v \mathbf{m}^i_{v,v'}$).

Finally, the set of existing vertices has symmetries in its representation. Clauses IVi force existing vertices to be the smallest in $\{1 \dots m'\}$. They break symmetries in the representation of the existing set; they allow only one representation of a set with r elements instead of $\binom{m'}{r}$. By pushing existing vertices to the left, they allow clauses IVii to enforce the lower bound of k existing vertices straightforwardly. There are a linear number of clauses of constant size, hence linear size.

Coding surjective functions on a variable set is done in quadratic size.

Coding a Subgraph Epimorphism

The "subgraph" part has been taken care of by using the value 0 for deleted vertices. In the additional constraints, we simply ignore the value 0.

A graph morphism can be seen as a morphism of relation systems [36]. This will prove useful to encode species and reactions.

Definition 6.2 (Relation System)**.** *A* relation system *is a set E equipped with a set of labels $\mathcal{L}$, where a label is a triple $(name, arity, R)$, with $arity \in \mathbb{N}$ and $R \subseteq E^{arity}$.*

Two labels cannot both have the same name and the same arity.

Example 6.3. *In this formalism, labels can be of any arity, so arcs can be coded as labels: $G = (V, A)$ can be coded as $G = (V, \{(arcs, 2, A)\})$.*

Definition 6.4 (Morphism of relation systems)**.** *Let $(E, \mathcal{L})$ and $(E', \mathcal{L}')$ be relation systems.*

A function $\mu : E \longrightarrow E'$ is a morphism of relation systems *if $\forall L = (n, a, R) \in \mathcal{L}, \exists L' = (n, a, R') \in \mathcal{L}'$, with $\forall (u_1 \dots u_a) \in R, (\mu(u_1) \dots \mu(u_a)) \in R'$*

For pure subgraph epimorphisms, we can restrict to one type of label: arcs of arity 2. Then the morphism requirement on relation systems is exactly to send arcs on arcs.

Let us constrain SEPI using this generalized formalism.

Variables. We use variables for the image of tuples.

These are the variables for a label with name n and arity a: Suppose there is a label $L = (n, a, R)$ in the first relation system and $L' = (n, a, R')$ in the second.

- $\forall (v'_1, \ldots, v'_a) \in \{1 \ldots m'\}^a, \mathbf{l}^n_{v'_1,\ldots,v'_a} = 1 \Leftrightarrow (v'_1, \ldots v'_a) \in R'$.
- $\forall (v_1, \ldots, v_a) \in R, (v'_1, \ldots, v'_a) \in \{1 \ldots m'\}^a$,
  $\mathbf{m}^i_{(v_1,\ldots,v_a),(v'_1,\ldots,v'_a)} = 1 \Leftrightarrow \mu_i(v_1) = v'_1 \wedge \ldots \wedge \mu_i(v_a) = v'_a$.

Clauses.

I Tuple Images. $\forall l = (v_1, \ldots, v_a) \in R, \forall l' = (v'_1, \ldots, v'_a) \in \{1 \ldots m'\}^a$,
$\mathrm{cl}((\bigwedge_{c\in\{1,\ldots,a\}} \mathbf{m}^i_{v_c,v'_c}) \Leftrightarrow \mathbf{m}^i_{l,l'})$

II Label Epimorphism. $\forall l' \in \{0, \ldots, m'\}^a$, $\mathrm{cl}(\bigvee_{l\in R} \mathbf{m}^i_{l,l'} \Leftrightarrow \mathbf{l}_{l'})$

We use the factorization trick of the previous section again: the clauses II code for $\bigvee_{l\in R} \mathbf{m}^1_{l,l'} \Leftrightarrow \bigvee_{l\in R} \mathbf{m}^2_{l,l'}$. This factorization saves an order of magnitude for the encoding size, so that for labels of arity up to 2, the coding has cubic size.

Symmetry breaking

Suppose two SEPIs μ_1 and μ_2 are found from G_1 and G_2 to G' of size n'. Then, composition with a permutation of the n' vertices yields another way to describe G' as SEPI lb of G_1 and G_2. Such equality up to permutation is an equivalence relation of the pairs (μ_1, μ_2). We can force the tuple $(\mu(1) \ldots \mu(n))$ to be lexicographically minimal among all the possible $\sigma \circ \mu$ where σ is a permutation.

The following scheme is described in [69]: associate a bounding function $M : V_i \longrightarrow \{1, \ldots, n_i\}$, with $M(1) = 1$, $\forall v \in V_i \mu_i(v) \leq M(v)$, and $M(v+1) = \max(M(v), \mu_i(v) + 1)$. This is called a *precedence* constraint. When listing the numbers that appear for the first time when reading $(\mu(1) \ldots \mu(n))$ from left to right, this precedence constraint forces the first-time value sequence to be increasing and to have no gap, which makes it the smallest for the lexicographical ordering.

Example 6.5. *The tuple* $(1, 2, 1, 4, 2, 3)$ *has first-time value sequence* $(1, 2, 4, 3)$*, which does not comply with the constraints.*

The permutation-equivalent sequence $(1, 2, 1, 3, 2, 4)$ *complies.*

This scheme cannot be applied on both SEPIs at the same time, it yields best performance when applied on the SEPI from the graph with smallest size. We describe how to apply the constraints on $\mu : V \longrightarrow V'$:

Variables

- $\mathbf{M}_{v,b} = 1$ iff $M(v) \leq b$

Clauses

I $M(1) = 1$: $\mathrm{cl}(\mathbf{M}_{1,1})$

II M exists: $\forall v \in V, \mathrm{cl}(\bigvee_{v' \in \{1,\ldots,n'+1\}} \mathbf{M}_{v,v'})$

III $\mu \leq M$: $\forall v \in V, v' \in \{1, \ldots, n' - 1\}$,

(a) $\mathrm{cl}(\mathbf{M}_{v,v'} \Rightarrow \mu^{<}_{v,v'+1})$

(b) $\mathrm{cl}(\mathbf{M}_{v,v'} \Rightarrow \mathbf{M}_{v,v'+1})$

IV $M(v+1) = \max(M(v), \mu_i(v) + 1)$: $\forall v \in V - \{n\}, v' \in \{1, \ldots, n'\}$

(a) $\mathrm{cl}(\mathbf{M}_{v,v'} \Rightarrow \mathbf{M}_{v+1,v'+1})$

(b) $\mathrm{cl}(\neg\mu_{v,v'} \vee \neg\mathbf{M}_{v+1,v'})$

(c) $\mathrm{cl}((\mu^{<}_{v,v'} \wedge \mathbf{M}_{v,v'}) \Rightarrow \neg\mathbf{M}_{v+1,v'})$

This encoding of the precedence constraint is of quadratic size.

The precedence constraint could force value in a top-down order instead of bottom-up, but this would conflict with the normalization of existing vertices, and force all vertices to be existing. Choosing a bottom-up order keeps the symmetry breakings compatible.

Specialization to Biochemical Networks

For biochemical networks, we encode species and reactions as labels. This forces the targets of reactions to be reactions and species to be species. However, this does not prevent vertices in the SEPI lb from being both reactions and species! We add some constraints to prevent this from happening:

Clauses

I $\forall v' \in \{1, \ldots, m'\}, \mathrm{cl}(\neg\mathbf{l}^{\text{reaction}}_{v'} \vee \neg\mathbf{l}^{\text{species}}_{v'})$

Notice that we do not constrain 0: since both species and reactions may be deleted, we may need vertex 0 to be both a reaction's image and a species'.

6.3 SAT SEPI LUB model

6.3.1 SAT coding

Let G_1, G_2 be two graphs. The goal of this section is to formulate the existence of a graph G with k vertices or less such that $G \to^*_{md} G_1$ and $G \to^*_{md} G_2$.

The model we use is dual to the one for SEPI glb.

Coding a Partial surjective function on a Set of Variable Size

As the coding is the same for μ_1 and μ_2, we describe the coding for μ_i.

Variables. μ_i is encoded as a binary relation on $\{0..k\} \times V_i$.

- $\forall (v, v') \in \{1 \dots k\} \times \{0\} \cup V_i, \mathbf{m}^i_{v,v'} = 1$ iff $\mu_i(v) = v'$.
- $\forall (v, v') \in \{1 \dots k\} \times \{0\} \cup V_i, \mathbf{m}^{i,<}_{v,v'} = 1$ iff $\mu_i(v) < v'$.
- $\forall v \in \{1 \dots k\}, \mathbf{e}_v = 1$ iff $v \in V$ iff $\mu_1(v) \neq 0 \vee \mu_2(v) \neq 0$

Left totality and functionality clauses take 0 into account, covering and existing set clauses still do not have to.

Clauses. The following clauses are used to force variables to match their description:

I. Left Totality. $\forall v \in V$, $\mathrm{cl}(\bigvee_{v' \in \{0 \dots n_i\}} \mathbf{m}^i_{v,v'})$

II. Functionality. $\forall (v, v'_j) \in V \times \{0 \dots n_i\}$,

   i. $\mathrm{cl}(\mathbf{m}^i_{v,v'_j} \Rightarrow \mathbf{m}^{i,<}_{(v,v'_{j+1})})$

   ii. $\mathrm{cl}(\mathbf{m}^{i,<}_{v,v'_j} \Rightarrow \mathbf{m}^{i,<}_{(v,v'_{j+1})})$

   iii. $\mathrm{cl}(\mathbf{m}^{i,<}_{v,v'_j} \Rightarrow \neg \mathbf{m}^i_{(v,v'_j)})$

III. Covering of Existing Vertices.
$\forall v' \in \{1 \dots n_i\}, \mathrm{cl}(\bigvee_{v \in V} \mathbf{m}^i_{v,v'})$

IV. Existing Set Normalization.

   i. $\forall v \in \{2 \dots k\}, \mathrm{cl}(\mathbf{e}_v \Rightarrow \mathbf{e}_{v-1})$

   ii. $\forall v \in \{1, \dots, k\}, \mathrm{cl}(\neg \mathbf{e}_v \Leftrightarrow \mu^1_{v,0} \wedge \mu^2_{v,0})$

   iii. $\forall v \in \{1 \dots \max(n_1, n_2)\}, \mathrm{cl}(\mathbf{e}_{v'})$

Left totality does not change, and we use the ordering to express functionality in quadratic size again.

Right totality does not take existence into account. There are a linear number of clauses of linear size, amounting to quadratic size.

The set of existing vertices has again its symmetries broken, and this time communication between the two functions occur here, obviously. Coding this is done in linear size.

Coding surjective functions from a variable set is done in quadratic size.

Coding a Subgraph Epimorphism

We use relation systems again to code the graph morphism part, but this time we cannot ignore the value 0. Indeed, with SEPI glbs, if some arc has to exist in the glb because of the first graph, it has to be covered by the second, which can be coded without mentioning bottom vertices. In the lub case, if some arc has to exist in the lub because of the first graph, its image by μ_2 does not have to be in the arcs of the second graph, since it may be deleted. There will be additional variables to help solve this problem.

Variables. We use variables for the image of tuples.

These are the variables for a label with name n and arity a: Suppose there is a label $L = (n, a, R)$ in the first relation system and $L' = (n, a, R')$ in the second.

- $\forall l = (v_1, \ldots, v_a) \in \{1 \ldots k\}^a, \mathbf{l}^n_l = 1 \Leftrightarrow l \in R$.
- $\forall l = (v_1, \ldots, v_a) \in \{1 \ldots k\}^a, l' = (v'_1, \ldots, v'_a) \in R'$
  - $\mathbf{m}^i_{l,l'} = 1 \Leftrightarrow \mu_i(v_1) = v'_1 \wedge \ldots \wedge \mu_i(v_a) = v'_a$.
  - $\mathbf{lm}^i_{l,l',n} = 1 \Leftrightarrow \mu_i(v_1) = v'_1 \wedge \ldots \wedge \mu_i(v_a) = v'_a \wedge l \in R$
- $\forall l = (v_1, \ldots, v_a) \in \{1 \ldots k\}^a, \mathbf{d}^n_l = 1 \Leftrightarrow \bigvee_{i \in 1 \ldots a} \mu(v_i) = 0$

Clauses.

I Tuple Images. $\forall l = (v_1, \ldots, v_a) \in \{1 \ldots k\}^a, \forall l' = (v'_1, \ldots, v'_a) \in R'$, $\mathrm{cl}((\bigwedge_{c \in \{1,\ldots,a\}} \mathbf{m}^i_{v_c, v'_c}) \Leftrightarrow \mathbf{m}^i_{l,l'})$

II Labeled Tuple Images. $\forall l = (v_1, \ldots, v_a) \in \{1 \ldots k\}^a, \forall l' = (v'_1, \ldots, v'_a) \in R'$, $\mathrm{cl}(\mathbf{m}^i_{l,l'} \wedge \mathbf{l}^n_l \Leftrightarrow \mathbf{lm}^i_{l,l',n})$

III Deleted Tuples . $\forall l = (v_1, \ldots, v_a) \in \{1 \ldots k\}^a$, $\mathrm{cl}((\bigvee_{c \in \{1,\ldots,a\}} \mathbf{m}^i_{v_c,0}) \Leftrightarrow \mathbf{d}^i_l)$

IV Label Morphism. $\forall l' \in R'$, $\mathrm{cl}(\bigvee_{l \in \{1 \ldots k\}^a} \mathbf{lm}^i_{l,l',n})$

V Label Epi. $\forall l = (v_1, \ldots, v_a) \in \{1 \ldots k\}^a, \mathrm{cl}((\mathbf{l}^n_l \wedge \mathbf{d}^i_l) \Rightarrow \bigvee_{l' \in R'} \mathbf{m}^i_{l,l'})$

Once again we use the factorization trick, which yield a cubic size coding for the same reasons.

Symmetry breaking

We use the same symmetry breaking technique on the graph representation. This time it constrains the SEPI lub, but the principle and execution are the same, however, it has best performance when applied on the SEPI from the graph with *largest* size.

Specialization to Biochemical Networks

For biochemical networks, we do the same as last section, and encode species and reactions as labels, and then force vertices to have either a species label or a reaction label.

6.4 Performance

Chapter 9 presents a performance evaluation of the CP and SAT approaches on graphs from our application.

Part III

Application: Detection of Model Reductions in Collections of Biochemical Models.

Chapter 7

Reaction Model Reduction

In this part, we will use the structure of models to relate them by graph operations that approximate model reduction.

This chapter introduces the link between models and their structure, by first introducing ODE models, then the more structured reaction models with underlying reaction graphs, and show how some model reductions at the ODE level are linked to the graph operations we presented earlier.

By doing so, we provide a theoretical link between model reduction in systems biology and our graph approach, aiming to support the view that SEPI is an important relaxation of model reduction.

The practical trials of chapter 10 will later confirm this claim.

7.1 Reaction Models

Systems of Ordinary Differential Equations are a popular tool for modelization of physical phenomena. Among the tools in our framework, they are the most commonly used, so we start the exposition of our framework by a definition of ODEs:

Definition 7.1 (System of Ordinary Differential Equations)**.** *Let* $\vec{X} = (x_1 \dots x_n)$ *be a vector of variables. A system of* ODE *over* $\vec{X}$ *is a system of* n *equations:*

$$\mathcal{O} = \begin{cases} \dot{x_1} &= f_1(x) \\ & \dots \\ \dot{x_n} &= f_n(x) \end{cases}$$

Most often, the intent behind systems biology ODE models is to describe a network of biochemical reactions between molecular species, so that $S = \{x_1, \dots, x_n\}$ are molecular species. Using ODEs is quite a crude way to describe a set of chemical reactions. Let us define formally a *reaction model* as a set of reactions, which are processes that takes in some *reactants*, and outputs some *products* at a certain *rate*:

Definition 7.2 (Reaction model)**.** *A* reaction model R *is a finite set of* m *reactions, written*

$$\mathcal{M} = \{\ e_i \texttt{ for } \ r_i \ \texttt{=>} \ \ p_i\ \}_{i=1,\ldots,m}$$

where e_i *is a formal mathematical expression over molecular species concentrations (possibly involving symbolic parameters),* r_i *and* p_i *multisets of molecular species. The* r_i *represent the reactants of the reaction, and* p_i *its products.*

The species that are both reactants and products in a reaction are called catalysts. For a multiset r of molecular species, i.e. a function $S \longrightarrow \mathbb{N}$, we denote by $r(x)$ the multiplicity of x in r, i.e. $r(x) = 0$ if x does not belong to r, and $r(x) \geq 1$ if x belongs to r, which is also written $x \in r$. The empty multiset is written $\_$. A multiset r will also be sometimes denoted by the linear expression with integer stoichiometric coefficients $\sum_{i=1}^{m} r(x_i) * x_i$.

We can derive an ODE system $\mathcal{O}(\mathcal{M})$ from $\mathcal{M}$:

Definition 7.3. *The set of equations associated to* $\mathcal{M}$ *is*

$$\mathcal{O}(\mathcal{M}) = \{\dot{x_j} = \sum_{i=1}^{n} e_i * (p_i(x_j) - r_i(x_j))\}_{j \in S}$$

By interpreting species as positive real numbers (their concentration), we can associate a dynamics to a reaction model through its derived ODE system.

Hence, reaction models can be used as a modelling tool, that have more structure than ODE systems. We will explore further a method to infer reaction models from ODE models in Chap. 8.

Example 7.4. *The enzymatic mechanism*

$$E + M \rightleftarrows_{k_d}^{k_c} F \rightarrow_{k_p} E + P$$

can be modelled by the ODE system:

$$\mathcal{O} = \begin{cases} \dot{M} &= -k_c \cdot E \cdot M + k_d \cdot F \\ \dot{E} &= -k_c \cdot E \cdot M + k_d \cdot F + k_p \cdot F \\ \dot{F} &= k_c \cdot E \cdot M - k_d \cdot F - k_p \cdot F \\ \dot{P} &= k_p \cdot F \end{cases}$$

The more structured reaction model uses the intended reactions:

$$\mathcal{M} = \begin{Bmatrix} k_c \cdot E \cdot M & \texttt{for} & E + M & \texttt{=>} & F \\ k_d \cdot F & \texttt{for} & F & \texttt{=>} & E + M \\ k_p \cdot F & \texttt{for} & F & \texttt{=>} & E + P \end{Bmatrix}$$

This reaction model's associated ODE system $\mathcal{O}(\mathcal{M})$ *is exactly* $\mathcal{O}$*.*

An ODE system can be seen as a reaction model from which the reaction structure has been removed. Now if we removed kinetic information from a reaction model, we would get a Petri Net []. By going even further and removing stoichiometric information, leaving only reactant and product information about reactions, we get our abstraction of interest, reaction graphs.

Formally, a reaction graph G is a bipartite directed graph, where some vertices are *species vertices* and the others are *reaction vertices*. The set of arcs describes how species interact through reactions.

Definition 7.5 (Reaction Graph)**.** *A* reaction graph G *is a triple* $G = (V, A, t)$*, where* $t : V \longrightarrow \{species, reaction\}$ *labels the type of vertices:* $S = t^{-1}(species)$ *is the set of* species *vertices,* $R = t^{-1}(reaction)$ *is the set of* reaction *vertices, and* $A \subseteq S \times R \cup R \times S$*.*

It is equivalent to write a reaction graph by separating species and reaction: we will also write (S, R, A)*.*

There is an arc (s, r) (resp. (r, s)) if s is a reactant (resp. product) of r. Both arcs can be present, e.g. if s is a catalyst of r. Arcs can only appear between a reaction and a species, never between vertices of the same type.

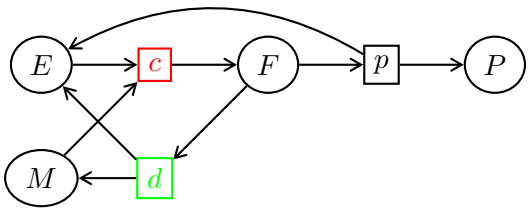
A reaction graph is derived from a reaction model as follows:

Definition 7.6. *Let* $\mathcal{M} = \{\ e_i$ `for` r_i `=>` $p_i\ \}_{i=1,\dots,m}$*. Then the derived reaction graph is*

$$G(\mathcal{M}) = (S, R, A)$$

where $R = \{1, \dots, m\}$ *disjoint from* S*, and* $A = \{(s, r) \in S \times R \mid p_r(s) > 0\} \cup \{(r, s) \in R \times S \mid r_r(s) > 0\}$*.*

Example 7.7. *The following reaction graph expresses the enzymatic mechanism from ex. 7.4.*



The species are represented here by ellipse vertices, they are enzyme E*, substrate* M*, complex* F*, and produce* P*. The reactions are represented by rectangle vertices, they are complexation* c*, decomplexation* d*, and production* p*. So* $S = \{E, M, F, P\}, R = \{c, d, p\}$*, and* $A = \{(M, c), (E, c), (c, F),\ (d, M), (d, E), (F, d),\ (p, P), (p, E), (F, p)\}$*.*

7.2 Reduction Operations on Reaction Graphs

The purpose of this section is to show how the delete and merge graph operations relate to model reduction.

We give examples of reduction operations on $\mathcal{M}$ that leave $\mathcal{O}(\mathcal{M})$ invariant (up to equivalence), while inducing a delete or merge operation on $G(\mathcal{M})$; in other words, the graph operations *capture* the model reductions.

Several facts need to be stated to measure the importance of these examples:

- these model reductions seem too perfect to be applicable in real systems, however reductions are often approximations, and some are approximations of these examples ;
- these reductions are fundamental enough that they *should* be captured by any framework claiming to capture biochemical model reductions ;
- many model reductions can be decomposed into steps featuring the examples.

Moreover, some complex model reductions may not be decomposed into these particular reductions, but the graph transformation they induce may still be decomposed into merge/delete chains.

The framework could capture too many operations and be able to relate any graph to any other, however we will see in chapter 10 that it is surprisingly accurate.

In this section, we suppose we have a reaction model $\mathcal{M} = \{\ e_i \text{ for } r_i \Rightarrow p_i\ \}_{i\in\{1\ldots m\}}$.

7.2.1 Constant concentration

If the concentration of some species s is constant, $\mathcal{M}$ can be reduced to the same model where the variable s is replaced by its value: this leaves $\mathcal{O}(\mathcal{M})$ invariant. The effect on the underlying graph is to delete this species vertex.

The typical case where this can be applied is when a species is in excess: then the variation of its concentration is small compared to its absolute value, which can be approximated by considering the variation as zero, hence the concentration is considered constant.

7.2.2 Zero rate reaction, Trivial reaction

If the rate of some reaction is exactly 0, this reaction can be removed from the reaction model. Indeed, it has no effect on the concentrations of molecular species, hence removing it from the ODE system yields a system which has the same semantics.

The effect on the underlying graph is removing the reaction vertex.

If a reaction has the same multiset of reactants and product, i.e. $r_i = p_i$, then its action is nullified in ODEs. This can happen after species a and b have been merged: then reactions $a + r$ `=>` $b + r$ become trivial.

Another case where this kind of model reduction is made is when a reaction is deemed negligible compared to another: then its rate may be approximated to 0.

7.2.3 Proportional concentrations

If two species have proportional concentrations, substituting one by the other times a correcting factor in all expressions yields a model M' with the same semantics. Merging the two vertices in G yields the same reaction graph G'.

Proposition 7.8. *If two species x_{j_1} and x_{j_2} have proportional concentrations, then there exists a reaction model R' with reaction graph $m_{x_{j_1},x_{j_2}}(G)$ that has equivalent ODEs.*

Proof. Suppose $x_{j_2} = \alpha x_{j_1}$. The set of equations associated to the model is

$$\mathcal{O} = \left\{ \dot{x_j} = \sum_{i=1}^{n} (p_i(x_j) - r_i(x_j)) * e_i \right\}_{j \in S}$$

and it can be rewritten

$$\mathcal{O}' = \left\{ \dot{x_j} = \sum_{i=1}^{n} ((p_i(x_j) - r_i(x_j)) * e_i)[x_{j_2}/\alpha x_{j_1}] \right\}_{j \in S - \{j_2\}}$$

where $t[x/y]$ is the term t where every occurrence of x has been replaced with y. This ODE system is defined with expressions that take the same values as the expressions in $\mathcal{O}$ everywhere, thus it is equivalent. Moreover, $\mathcal{O}'$ is exactly the ODE system of the reaction model

$$R' = \{\ e_i[x_{j_2}/\alpha x_{j_1}] \text{ for } r_i[x_{j_2}/\alpha x_{j_1}] \Rightarrow p_i[x_{j_2}/\alpha x_{j_1}]\ \}_{i \in \{1 \ldots m\}}$$

For non-integer values of α, we may have to extend our definition of reaction model to allow non-integer stoichiometry in reaction models for some values of α. Still, this (extended) reaction model has reaction graph $m_{x_{j_1},x_{j_2}}(G)$. □

A typical phenomenon that may make concentrations almost proportional is fast equilibria. When the reactions between some species are much faster than reactions involving those species and other species, then those fast reactions may make their species almost proportional in concentration, depending on the type of kinetic laws involved.

The modeler may choose to reduce a system displaying this behaviour by approximating the concentrations as being proportional.

7.2.4 Proportional rates

The same type of phenomenon can happen for reaction rates. In a chain of reactions, the slowest one, or rate-limiting step, can force the reactions that follow it to go no faster than itself.

Proposition 7.9. *If two reactions i_1 and i_2 have proportional rates, then there exists a reaction model R' with reaction graph $m_{i_1,i_2}(G)$ that has equivalent ODEs.*

Proof. Suppose $e_{i_2} = \alpha e_{i_1}$. The set of equations associated to the model is

$$\left\{ \dot{x_j} = \sum_{i=1}^{n} (p_i(x_j) - r_i(x_j)) * e_i \right\}_{j \in S} \tag{7.1}$$

and it can be rewritten

$$\left\{ \dot{x_j} = \sum_{i=1, i \neq i_1, i \neq i_2}^{n} (p_i(x_j) - r_i(x_j)) * e_i \right. \tag{7.2}$$

$$\left. + \left(p_{i_1}(x_j) + \alpha p_{i_2}(x_j) - r_{i_1}(x_j) - \alpha r_{i_2}(x_j)\right) * e_{i_1} \right\}_{j \in S} \tag{7.3}$$

which is exactly the ODE system of the reaction model

$$R' = \{ \ e_i \text{ for } r_i \Rightarrow p_i \ \}_{i \neq i_1, i_2} \cup \{ e_{i_1} \text{ for } r_{i_1} + \alpha r_{i_2} \Rightarrow p_{i_1} + \alpha p_{i_2} \} \tag{7.4}$$

This reaction model has reaction graph $m_{i_1,i_2}(G)$. □

7.3 Conclusion

We have seen that SEPI captures common model reductions, in the sense that if $\mathcal{M}$ reduces to $\mathcal{M}'$ using these reductions, then there is a SEPI from $G(\mathcal{M})$ to $G(\mathcal{M}')$. This allows us to filter out impossible reductions: if there is no SEPI from $G(\mathcal{M})$ to $G(\mathcal{M}')$, then there is no way to reduce $\mathcal{M}$ to $\mathcal{M}'$ using captured model reductions.

Since we have the algorithms to do so, the next step is to try the framework on real models. We need a second ingredient for our trial: data is taken care of in the next chapter.

Chapter 8

Retrieving Reaction Graphs from ODE Models

In Systems Biology, many models are presented as a system of Ordinary Differential Equations (ODEs). This simple mathematical formalism completely defines the dynamical behavior of a system of biochemical reactions once the kinetic parameter values are fixed. It provides powerful tools for both transient and steady-state analyses via numerical integration (for instance using MATLAB®, Copasi, ...), parameter sensitivity analyses, or bifurcation analyses, (with tools like XPPAUT [21] for instance), but only when kinetic information is available.

In absence of knowledge on the kinetics of each reaction, various qualitative analyses can be performed on the *structure* of the reaction network. This approach has rapidly developed in Systems Biology for reasoning on large interaction networks, with for instance, the analysis of qualitative attractors in a logical dynamics of gene networks [65], reachability and temporal logic properties in reaction networks [20, 5, 12, 24, 10], structural invariants in the Petri net representation of the reactions [54, 71, 2, 14, 58, 63], or model reductions using graph theory concepts [31, 29]. These qualitative analysis tools do not rely on kinetic information, but on the structure of the reaction network which has thus to be correctly written as a set of formal reaction rules, with well-identified reactants, products and modifiers (and in certain cases their stoichiometry) for each reaction.

For instance, in [38], it is elaborated that structural information hidden in kinetic laws may affect the results obtained from structural analyses, such as elementary mode analysis [59], flux balance analysis [67], chemical organization theory [18], deficiency analysis or chemical reaction network theory [25, 60]. Furthermore, the correct structure is mandatory when a reaction network must be interpreted as a stochastic process *à la* Gillespie [32].

It is worth noticing that these structural analyses may also directly support dynamic analyses. For instance, [39] applies network decomposition for a modular parameter estimation approach, [2] introduces a structural persistence criterion, Petri net place invariants

reveal conservation laws in [62], while transition invariants can be used to identify fragile vertices and the core of a network [35], or to determine steady state solutions [53].

We intend to try SEPI on real-life models repository BioModels ([43]), however the expressivity allowed by the SBML format and the common practice of hard-coding ODEs into models make it so that extracting reaction graphs straightforwardly would yield incomplete structure from models. In this chapter, we explain the import algorithms used in Biocham ([23, 11]) to deal with the gap between an intended reaction model and the given encoding.

This preprocessing algorithm has been published in [22], the contribution of the author is mainly the hidden molecule inferrence algorithm, which was actually already used in another form in [31].

8.1 Issues Related to BioModels Encodings

Any ODE model can be transcribed in a reaction system using artificial synthesis and degradation reaction rules for each molecular species, with the positive, respectively negative, terms of the differential equation for the variable as kinetic expression. This is obviously correct as far as the ODE semantics is concerned, but prevents the use of structural analysis methods or stochastic simulations since the structure of the reactions are then meaningless.

Since the primary concern of SBML is to communicate models and play simulations, some ODE models have been transcribed in SBML using the artificial reactions scheme. This is the case for instance of model `BIOMD0000000008.xml` in BioModels, for the ODE model of [27]. This model adds a control mechanism to the cell-cycle model of Goldbeter et al. in [33] but with this transcription in SBML, the reaction graph is not even connected.

Here are some of the reactions of this model which illustrate the problem:

```
r4: (1+ -1*[M])*V1*(r4K1+(-1*[M]+1))^-1 for _ => M.
r5: [M]*r5V2*(r5K2+[M])^-1       for M => _.
r6: V3*(1+ -1*[X])*(r6K3+(-1*[X]+1))^-1 for _ => X.
r7: r7V4*[X]*(r7K4+[X])^-1       for X => _.
```

One of the issues is hidden in the definition of `V1` and `V3`:

```
macro(V1,[C]*V1p*([C]+K6^-1)).
macro(V3,[M]*V3p).
```

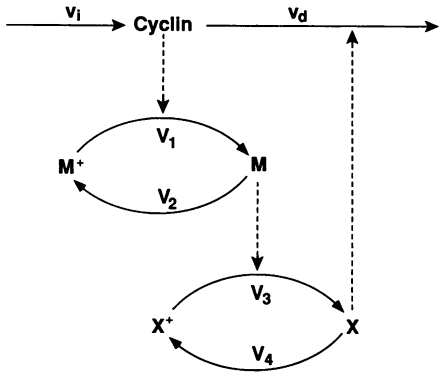
These definitions show that, as pointed out by a "K not R" warning, `C` is indeed involved in the kinetics of `r4` but is not marked as modifier.

One can also note that, though encoded in complicated MathML expressions, `1 - [M]` (resp. `1 - [X]`) appears in the synthesis of `M` (resp. `X`) as a hidden form of the inactive form of `M` (resp. `X`). Indeed, [33] states that *"(1 - M) thus represents the fraction of inactive (i.e., phosphorylated) cdc2 kinase, while (1 - X) represents the fraction of inactive (i.e., dephosphorylated) cyclin protease"*.

When applied to the ODE system associated to this model, our reaction system inference algorithm infers the following well-formed and strict reactions:

```
r4: V1*[Mi]*(r4K1+[Mi])^-1 for Mi+C => M+C.
r5: r5V2*[M]*(r5K2+[M])^-1   for M => Mi.
r6: V3*[Xi]*(r6K3+[Xi])^-1 for Xi+M => X+M.
r7: r7V4*[X]*(r7K4+[X])^-1   for X => Xi.
```

The fact that the two inactive forms are now explicitly represented by new molecules and that the action of `C` on `M` and of `M` on `X` are properly transcribed, provides a well-formed reaction system which is consistent with the usual graphical representation of the paper [33] and suitable for further structural analysis:



The following sections present our reaction system inference algorithm in two steps: first the algorithm for inferring hidden molecules eliminated by invariants, second the algorithm for inferring well-formed reaction rules whenever possible.

8.2 Inference Algorithm for Hidden Molecules

ODE models often contain algebraic invariants, among which linear invariants, e.g. mass conservation invariants or Petri-net place invariants, are an important particular case. A linear invariant can be used to simplify a model by eliminating one variable and replacing it with a linear expression. This may have several advantages, but when writing the model with reactions, such simplifications performed on the ODE system need be reversed in order to restore the eliminated molecular species, as shown in the previous section.

A preprocessor is thus applied before the reaction inference algorithm in order to reverse the elimination of linear invariants and infer hidden molecules. Obviously, the expressions f for which new molecules are introduced need be chosen with care, otherwise useless variables may be introduced, for instance if $f = x_i$. Restricting the search to expressions

of the form $k - x$ or $k - x - y$ where k is a constant or parameter, and x and y are molecule concentrations, leads to

Algorithm 8.1 (Hidden molecule inference)**.**

Input: parametric ODE system O over variables for molecular concentrations
Output: parametric ODE system O over more variables for molecular concentrations, including hidden molecules

1. *iteratively replace in O any expression of the form $-1 * x + y$ by $y - x$,*
2. *for each expression of the form $x - y - z$ in O where x is a constant or a parameter, and y and z are variables,*
   (a) *introduce a new variable $v = x - y - z$ with corresponding initial value and time derivative,*
   (b) *substitute any occurrence of $x - y - z$ in O by v,*
   (c) *substitute any occurrence of $x + w - y - z$ in O by $w + v$ for any w,*
   (d) *substitute any occurrence of $x - y + w - z$ in O by $w + v$ for any w,*
3. *for each expression $x - y$ appearing in O where x is a constant or a parameter and y is a variable,*
   (a) *introduce a new variable $v = x - y$ with corresponding initial value and time derivative,*
   (b) *substitute any occurrence of $x - y$ in O by v,*
   (c) *substitute any occurrence of $x + w - y$ in O by $w + v$ for any w,*

Proposition 8.2 (Soundness)**.** *The system obtained from the execution of algorithm 8.1 on an ODE system O is equivalent to O once projected on the original variables.*

Proof. We will prove that each step of the algorithm verifies the property and thus that the whole execution does. It is easy to check that step (1) is purely syntactical and does not change the ODE system O. Now note that all the other changes are of two forms. Either the introduction of a new variable v equal to a linear combination f of variables $f = \sum \lambda_i x_i$, steps (2c) and (3c). The new equation for v is $\dot{v} = \sum \lambda_i \dot{x}_i$, the $\dot{x}_i$ being given by the rest of the system. This does indeed not change the system once the new variable is projected out. Since one also imposes as initial condition for v the value of f at the initial state, the other changes, namely the substitution in the original system of some occurrences of f by v, steps (2d-f) and (3d-e), are guaranteed to keep an equivalent system. □

8.3 Inference Algorithm for Reactions

The inference algorithm for reactions is based on a syntactical normal form for ODE systems which facilitates the recognition of common subterms in the equations.

We represent mathematical expressions in ODEs or in kinetic laws as terms with mathematical operators ($+$, $-$, $/$, $*$, etc.) as function symbols, constants of $\mathbb{R}$ and variables representing species concentrations and parameters. This corresponds precisely to formalizing the MathML description of mathematical formulae.

Let us call *non-decomposable* a term that:

- is not an addition nor a subtraction, i.e., its functor (top function symbol) is neither $+$ nor $-$;
- cannot be reduced at top-level by the laws of distributivity of the product and division on addition and subtraction, i.e., if its functor is $*$ then the arguments do not have $+$ or $-$ as functor.

Definition 8.3. *A reaction* f `for` r `/` m `=>` p *over molecular species* $\{x_1, \ldots, x_s\}$ *is* non-decomposable *if* f *is syntactically a non-decomposable term.*

The non-decomposability condition excludes the composition of several reactions in a single one with a sum as kinetic expression.

Definition 8.4. *A mathematical expression is in* additive normal form *if it is of the form* $\sum_{i=1}^{k} c_i * t_i$ *where* c_i *are integer coefficients and* t_i *are distinct non-decomposable terms without integer coefficients or leading unary* $-$.

An ODE system is in additive normal normal form if each equation is in additive normal normal form, i.e. if it is of the form

$$\dot{x_i} = \sum_{j=1}^{l} c_{i,j} * t_j, \ 1 \leq i \leq s$$

where l *is the number of non-decomposable terms in the system.*

Additive normal forms are not unique, but any ODE system can be written in additive normal form through standard algebraic transformations. Now, given an ODE system in additive normal form, we can infer an equivalent reaction system by sorting the terms of the equations and creating a reaction rule for each term:

Algorithm 8.5 (Reaction inference)**.**

Input: parametric ODE system O *over variables for molecular concentrations*
Output: Reaction System R.

1. *put* O *in additive normal form*

2. *build the set* $\mathcal{T}$ *of all terms appearing in* O

3. *let* $R \leftarrow \varnothing$

4. *for each non-decomposable term* t *in* $\mathcal{T}$*,*

   (a) *let* $r \leftarrow \_$, $p \leftarrow \_$, $m \leftarrow \_$

   (b) *for each variable* x *where* t *occurs with integer coefficient* c *in* $\dot{x}$ *in* O*,*

      i. *if* $c < 0$ *then* $r(x) \leftarrow -c$,

      ii. *if* $c > 0$ *then* $p(x) \leftarrow c$,

   (c) *for each variable* x *such that* $r(x) = 0$ *and* $\partial t/\partial x > 0$ *for some values,*

      i. $r(x) \leftarrow 1$,

      ii. $p(x) \leftarrow p(x) + 1$,

   (d) *for each variable* x *such that* $\partial t/\partial x < 0$ *for some values,*

      i. $m(x) \leftarrow 1$,

   (e) $R \leftarrow R \cup \{t$ `for` r `/` m `=>` $p\}$,

Steps 4(c) and 4(d) require checking the sign of a partial derivative. As described in Section 8.4.1, this check can be arbitrarily difficult for arbitrary mathematical expressions, but can be over-approximated.

Example 8.6. *The model*

```
k1*[pMPF]*[Cdc25] for pMPF + Cdc25 => MPF + Cdc25
k2*[MPF]*[Wee1]   for MPF + Wee1 => pMPF + Wee1
k3/(k4+[Clock])   for _ / Clock => Wee1
```

has one invariant: $[pMPF]+[MPF]$ *is a constant* c *(the sum of initial values of* $pMPF$ *and* MPF*) since* $[\dot{pMPF}]+[\dot{MPF}] = 0$*. One variable, e.g.* $[pMPF]$*, can thus be eliminated and replaced by* $c - [MPF]$*. This yields the ODE system*

$$\begin{aligned} [\dot{MPF}] &= k1 * (c - [MPF]) * [Cdc25] - k2 * [MPF] * [Wee1] \\ [\dot{Wee1}] &= k3/(k4 + [Clock]) \\ [\dot{Cdc25}] &= 0 \\ [\dot{Clock}] &= 0 \end{aligned}$$

If directly applied to this system, Algorithm 8.5 infers the following reactions:

```
c*k1*[Cdc25]      for Cdc25 => Cdc25 + MPF
k1*[Cdc25]*[MPF]  for MPF + Cdc25 => Cdc25
k2*[MPF]*[Wee1]   for MPF + Wee1 => Wee1
k3/(k4+[Clock])   for _ / Clock => Wee1
```

By applying first the invariant inference algorithm, a hidden molecular species `MPFi` *is introduced for the expression* `c-[MPF]` *corresponding to the linear invariant* `[MPFi]+[MPF]=c`. *We have*

$$[\dot{MPF}_i] \quad = \quad -k1 * [MPF_i] * [Cdc25] + k2 * [MPF] * [Wee1]$$

and when applied to this ODE system after the preprocessing step, Algorithm 8.5 computes the correct reactions:

```
k1*[MPFi]*[Cdc25] for MPFi + Cdc25 => MPF + Cdc25
k2*[MPF]*[Wee1]   for MPF + Wee1 => MPFi + Wee1
k3/(k4+[Clock])   for _ / Clock => Wee1
```

8.4 Evaluation Results on BioModels

8.4.1 Computability Issues

Since we allow arbitrary mathematical expression for kinetic expressions, checking the well-formed conditions may raise arbitrary difficult symbolic computation problems. These conditions can be checked however by doing some approximations.

In our implementation in Biocham, the kinetic expressions are first normalized as if they were polynomials, stopping when a non-polynomial operator (anything else than $*$, $-$ and $*$) is found. For the polynomials, the exact computation of the sign of any partial derivative is easy, for the other terms, either they are recognized as a standard kinetics (like Hill functions) and once again the exact sign is extracted, or they are considered unknown and for any variable appearing we will assume that it is possible that $\partial f/\partial x$ becomes positive for some values, and negative for some values. This is a conservative over-approximation.

With these provisions, different syntactical conditions may indicate that a reaction is not well-formed. The conditions for a reaction to be ill-formed can be classified into three categories:

- "K not R" indicates that the concentration of a compound appears in the kinetic law of a reaction, but this compound is neither a reactant nor an inhibitor of the reaction;

- "R not K" indicates that some compound is marked as reactant or inhibitor in a reaction, but does not appear in the kinetic expression;

- "Negative" indicates that a kinetic expression may be negative with positive concentration values.

	K not R	R not K	Negative	Any warning
Original	173	123	157	234 (64.81 %)
Inferred	0	67	70	103 (28.53 %)

Table 8.1: Number of models having a K not R, R not K, or negative kinetics warning among the original 361 models of the curated part of BioModels, and among the reaction systems automatically inferred from their ODE semantics.

Indeed, in a well-formed reaction with kinetic expression f, if a species x is neither a reactant nor an inhibitor, then $\partial f/\partial x = 0$, hence x should not appear in the kinetic expression f. Similarly, if a species is a reactant or an inhibitor, then $\partial f/\partial x \neq 0$, so x should appear in f. Moreover, f should be well-defined and positive.

These ill-formedness conditions are checked in Biocham using the previous approximations. They correspond to the warning messages that Biocham can raise when loading a reaction system.

8.4.2 Global analysis

The 424 models from the curated branch of the latest version (release 24) of the BioModels repository [43] were used as benchmark to test our reaction system inference algorithm, and compare the results with the original writing of the models in SBML. Out of those 424 models only 361 define *reactions* with proper *kineticLaws*. The other ones only describe systems through events and rules, or with no kinetic information, and thus have no ODE semantics.

Table 8.4.2 summarizes the result of the procedure, as detected by Biocham warnings. Over the 361 reaction systems of the original curated part of BioModels, our algorithm detects 58 models with *hidden molecules*, 173 models with K not R warning, 123 models with R not K warning and 157 models with negative kinetics warning. Our algorithm is able to automatically curate the writing of these models with reaction rules by reducing the number of non well-formed models with a warning by more than the half, from 65% to 29%.

The algorithm 8.5 completely removes the "K not R" warnings. For the two other warnings, since the algorithm focuses on *non-decomposable* kinetics, it results in curated models quite close to the original ones, but does not tackle thoroughly the case of reactions with rates independent of some reactant. Therefore, 103 over 361 models remain with a non well-formedness warning.

8.4.3 Model inconsistencies studied in [38]

In [38], the authors also scan the whole BioModels repository and report finding 5 inconsistencies: models 44, 93, 94, 143 and 151. Their diagnostics is as follows, some reaction fluxes become negative during the simulations of those models because of missing reversibility indications in models 93, 94 and 143. In the two first cases they report that adding the reverse reactions makes the models consistent, whereas for 143 it is also necessary to change some kinetic law. For model 151 they report a missing step, but since the opposite reaction is part of the model, once again this amounts to adding a reverse reaction to an existing one. Finally, for model 44 they describe that the issue is that some kinetic expression does not depend on one of the reactants of the reaction, making it possible for that reactant's concentration to become negative.

For models 93, 94 and 151, that indeed are flagged by the "Negative" warning, our algorithm correctly adds the missing reverse reactions, directly from the kinetic expressions. The models automatically curated this way do not raise any warning at the end.

For model 44, the automatic curation allows us to get rid of a "K not R" warning by transforming the reaction `v3`
`A::cyt + Y::ves => A::cyt + Z::cyt` with the kinetic law
$cytosol * Vm3 * [A]^4 * [Y]^2 * [Z]^4/((Ka^4 + [A]^4) * ((Ky^2 + [Y]^2) * (Kz^4 + [Z]^4)))$ into
`Z::cyt + A::cyt + Y::ves => 2*Z::cyt + A::cyt`

However, as expected, the "R not K" warning identified by Kaleta *et al.* remains, the obtained model is still not well-formed. The same happens with model 143 where indeed a "R not K" warning remains after automatic curation, in accordance with the earlier results.

Chapter 9

Performance Evaluation

9.1 Data

The reaction graphs used for this benchmark come from the BioModels repository [44], in particular, the same models adopted in [31].

These are real-life, curated models retranscribed from publications. The method used to extract a reaction graph from a BioModels file will be described in the next part, more specifically in chapter 8.

A thematic clustering has been accomplished, using information available from the notes of SBML models. The four most populated classes are: *i)* mitogen-activated protein kinase (abbreviated as *mapk*, 11 models), *ii)* circadian clock (*circ*, 11 models), *iii)* calcium oscillations (*caoscill*, 11 models), and *iv)* cell cycle (*ccycle*, 9 models).

Some data about the models is regrouped in Table 9.1.

Class	#models	Min size	Max size	Median size
mapk	11	20	213	51
circ	11	24	68	43
caoscill	11	6	44	12
ccycle	9	7	334	43

Table 9.1: Some data about the reactions graphs used for the following benchmarks: in each line, the name of the clustered class, the number of vertices of the smallest graph of the class, the number of vertices of the largest graph of the class, and the median number of vertices of the graphs.

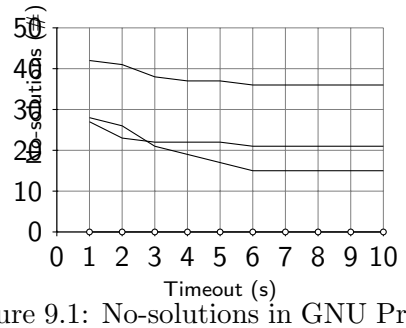


Figure 9.1: No-solutions in GNU Prolog.

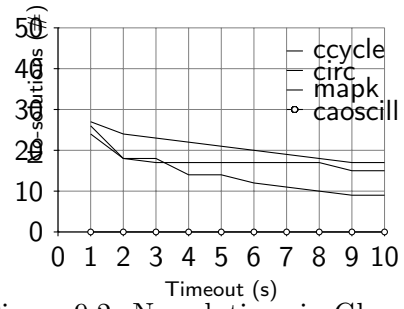


Figure 9.2: No-solutions in Glucose.

9.2 SEPI

We implemented the CLP model of chapter 5 using GNU Prolog [17] 1.4.4, and the SAT model from chapter 6 is solved with Glucose [3] 2.2.

In the experiments reported in Tab. 9.2, the computation time was limited with a timeout of 20 minutes. Performance has been evaluated on an Intel Core 2 Duo 2.4Ghz processor. The four macro-columns respectively show the number of intra-class comparisons, the number of relations found between models (i.e., of reductions), and the number of no-relations found, and, finally, the number of no-results (where timeout occurs). Each sub-column respectively reports performance for Glucose, GNU Prolog, and the methods combined together, using the same timeout for both (20min + 20min).

Clearly, in order to evaluate the two methods, the interesting value is the number of timeouts: here the greater efficacy of Glucose can be appreciated, in particular on class *circ* (from 33 no results to 0), but also on class *ccycle* (from 20 no results to 9), and, lastly, some improvement on class *mapk* (from 12 to 9). Class *caoscill* has the smallest graphs of the four clusters, which is probably what allows both approaches to decide all relations.

These results have been further investigated in detail, discovering that *mapk* is the only class among the four where the GNU Prolog set of relations is not a subset of the one found with Glucose. This difference set (equivalent to $49 \rightarrow 9$, $49 \rightarrow 11$, $49 \rightarrow 28$, $49 \rightarrow 30$) also corresponds exactly to the difference set of no-results between glucose and GNU Prolog. From this the reader can deduce that from a merging of GNU Prolog and SAT implementations, only 4 no-results less on *mapk* can be gained (which would correspond to 4 additional relations). Nevertheless, it is also possible to deduce that on some models our GNU Prolog version can run more efficiently: in this case, model 49 is better matched with our CP approach that with our SAT approach.

The lists of comparisons for which no result can be obtained with either SAT or GNU are respectively, $\{49 \rightarrow 146, 146 \rightarrow 9, 146 \rightarrow 11, 146 \rightarrow 28, 146 \rightarrow 30\}$ on *mapk*, and $\{56 \rightarrow 7, 56 \rightarrow 111, 56 \rightarrow 144, 109 \rightarrow 7, 109 \rightarrow 111, 109 \rightarrow 144, 144 \rightarrow 111, 144 \rightarrow 169, 144 \rightarrow 196\}$ on *ccycle*. Few of the models seem to represent a bottleneck, due to the high frequency of the same models in these two lists.

Table 9.2: Solvers performance collected in 20min. Multicolumns: Relations is the number of SEPI proved by the solver, Nonrelations is the number of SEPI arrows proved impossible, Timeouts is the number of comparisons for which no answer was given after the time limit. Columns: GNU is the CP model with GNU-Prolog, Glucose is the Boolean model with glucose, Union is the set union of (non)arrows proved by either solver in the time limit.

Class(Comparisons)	Relations			Nonrelations			Timeouts		
	GNU	Glucose	Union	GNU	Glucose	Union	GNU	Glucose	Union
mapk (110)	38	38	42	60	63	63	12	9	5
circ (110)	17	37	37	60	73	73	33	0	0
caoscill (110)	38	38	38	72	72	72	0	0	0
ccycle (72)	9	12	12	43	51	51	20	9	9

Table 9.3: Solvers performance collected in 10s.

Class(Comparisons)	Relations			Nonrelations			Timeouts		
	GNU	Glucose	Union	GNU	Glucose	Union	GNU	Glucose	Union
mapk (110)	36	35	41	59	60	60	15	15	9
circ (110)	15	33	33	59	68	68	36	9	9
caoscill (110)	38	38	38	72	72	72	0	0	0
ccycle (72)	9	6	10	42	49	49	21	17	13

Moreover, Tab. 9.3 shows that both implementations also perform well within a short timeout of 10 seconds. This is particularly true with our GNU Prolog implementation (only 7 no-results less over the four classes, from 10sec to 20min), while more debatable with Glucose (23 no-results less in total). Fig. 9.1 and 9.2 show how the number of no-solutions decreases by increasing the timeout from 1 up to 10 seconds (GNU Prolog and Glucose respectively).

9.3 SEPI glb

Evaluation of the SAT coding has been done on the previous benchmark for SEPI. The input is a pair of graphs (G_1, G_2) and $k = \min(|G_1|, |G_2|) * (100 - b)\%$. The results are summarized in Fig. 9.3 and in Fig. 9.4.

With $b = 0$, there is a SEPI glb iff there is a SEPI from one graph to the other. In Fig. 9.4, for the ca_oscil class, the results are complete. It is worth noting that there are 74 SEPI glbs, when there were 38 SEPI relations in this class.

For every relation $G_1 \rightarrow^*_{md} G_2$, there is a positive SEPI glb answer for instances $(G_1, G_2, 0)$ and $(G_2, G_1, 0)$. So there are nearly twice as many positive SEPI glb answers as there are SEPI relations, but why not exactly twice as many? Because of isomorphisms: in the ca_oscil class, models 115 and 117 are isomorphic, so they account for 2 relations in SEPI, and 2 positive answers for SEPI glb.

This is why there seems to be a discrepancy: actually, there are 2 positive SEPI glb answers for every SEPI between nonisomorphic pair, and only 1 for every isomorphic pair.

Another interesting observation is that the SAT instances should be the same for (G_1, G_2, b) and for (G_2, G_1, b). However, for some cases, there are an odd number of timeouts. This does not come from a computation finish exactly before the timeout and its "copy" stopping afterwards: instead, it comes from the order in which the clauses are put in the SAT instance. Indeed, the order of clauses can affect SAT learning and computation time.

9.4 SEPI lub

The same remarks as for SEPI glb are true for SEPI lub.

The results are summarized in Fig. 9.3 and in Fig. 9.4.

9.5 Conclusion

A synthetic way to summarize the results from this chapter is: deciding SEPI is clearly feasible, even for the larger graphs of BioModels. However, computing SEPI glbs and lubs is out of reach for larger graphs, and still hard for medium sized graphs.

To deal with hard instances of our problem, we still may improve our solutions.

Figure 9.3: Computation results for SEPI glb, with 10s timeout. First column: set of models used for the benchmark. Second column: bound used to limit the size of the glb, the formula for k is $k = \min(|G_1|, |G_2|) * (100 - \text{Bound})/100$. Last columns are number of relations $\exists G, |G| \geq k, G_1 \longrightarrow G \longleftarrow G_2$ found, number of nonrelations = relations proven impossible, and number of timeouts.

Class	Bound	Relation	No relation	Timeout
ca_oscil	0	74	36	0
	5	74	36	0
	10	86	24	0
	15	86	24	0
	20	100	10	0
	25	110	0	0
	30	110	0	0
circ	0	40	12	58
	5	48	2	60
	10	58	0	52
	15	67	0	43
	20	71	0	39
	25	78	0	32
	30	90	0	20
mapk	0	67	10	33
	5	68	10	32
	10	70	6	34
	15	74	4	32
	20	78	2	30
	25	78	2	30
	30	82	0	28
cell_cycle	0	18	20	34
	5	26	4	42
	10	28	2	42
	15	30	2	40
	20	33	2	37
	25	35	0	37
	30	37	0	35

Figure 9.4: Computation results for SEPI glb, with 30s timeout. First column: set of models used for the benchmark. Second column: bound used to limit the size of the glb, the formula for k is $k = \min(|G_1|, |G_2|) * (100 - \text{Bound})/100$. Last columns are number of relations $\exists G, |G| \geq k, G_1 \longrightarrow G \longleftarrow G_2$ found, number of nonrelations = relations proven impossible, and number of timeouts.

Class	Bound	Relation	No relation	Timeout
ca_oscil	0	74	36	0
	5	74	36	0
	10	86	24	0
	15	86	24	0
	20	100	10	0
	25	110	0	0
	30	110	0	0
circ	0	42	12	56
	5	49	2	59
	10	65	2	43
	15	72	0	38
	20	73	0	37
	25	89	0	21
	30	94	0	16
mapk	0	72	10	28
	5	73	10	27
	10	74	7	29
	15	78	4	28
	20	80	2	28
	25	80	2	28
	30	84	0	26
cell_cycle	0	24	20	28
	5	32	7	33
	10	37	2	33
	15	39	2	31
	20	43	2	27
	25	46	0	26
	30	46	0	26

Figure 9.5: Computation results for SEPI lub, with 10s timeout. First column: set of models used for the benchmark. Second column: bound used to limit the size of the lub, the formula for k is $k = \max(|G_1|, |G_2|) * (100 + \text{Bound})/100$. Last columns are number of relations $\exists G, |G| \leq k, G_1 \longrightarrow G \longleftarrow G_2$ found, number of nonrelations = relations proven impossible, and number of timeouts.

Class	Bound	Relation	No relation	Timeout
ca_oscil	0	74	36	0
	5	94	1	15
	10	98	0	12
	15	102	0	8
	20	108	0	2
	25	107	0	3
	30	103	0	7
circ	0	10	9	91
	5	2	0	108
	10	0	0	110
	15	2	0	108
	20	5	0	105
	25	7	0	103
	30	11	0	99
mapk	0	33	9	68
	5	26	2	82
	10	22	2	86
	15	22	0	88
	20	25	0	85
	25	22	0	88
	30	24	0	86
cell_cycle	0	0	11	61
	5	0	0	72
	10	1	0	71
	15	2	0	70
	20	1	0	71
	25	1	0	71
	30	4	0	68

Figure 9.6: Computation results for SEPI lub, with 30s timeout. First column: set of models used for the benchmark. Second column: bound used to limit the size of the lub, the formula for k is $k = \max(|G_1|, |G_2|) * (100 + \text{Bound})/100$. Last columns are number of relations $\exists G, |G| \leq k, G_1 \longrightarrow G \longleftarrow G_2$ found, number of nonrelations = relations proven impossible, and number of timeouts.

Class	Bound	Relation	No relation	Timeout
ca_oscil	0	74	36	0
	5	94	6	10
	10	98	0	12
	15	106	0	4
	20	109	0	1
	25	108	0	2
	30	110	0	0
circ	0	46	12	52
	5	23	0	87
	10	27	0	83
	15	29	0	81
	20	30	0	80
	25	27	0	83
	30	26	0	84
mapk	0	49	10	51
	5	45	2	63
	10	47	2	61
	15	48	0	62
	20	51	0	59
	25	50	0	60
	30	51	0	59
cell_cycle	0	4	18	50
	5	2	0	70
	10	4	0	68
	15	3	0	69
	20	4	0	68
	25	6	0	66
	30	9	0	63

First, the CP approach could be improved: GNU Prolog was used because Biocham was coded in GNU-Prolog, and some constraints are not available in this framework. The search heuristic we showed is better than a pure first-fail, but it could still be improved.

For the application point of view, we only consider pure graph structure, but using information on vertices as a side contraint, such as "these vertices are central to the model", could be interpreted as "these vertices should not be deleted, and not merged among themselves", or even simply distinguishing vertices by using labels, could greatly help reduce the search space.

Chapter 10

Evaluation of SEPI on BioModels

10.1 Motivation

BioModels [43] is a large repository of systems biology models, coded from original articles to SBML files, hand curated to match the published results. The website hosts hundreds of models and has many features, however these models can only be sorted by theme or keywords.

We apply our work on SEPI and reaction graphs to map the partial order of SEPI to this collection of models. Does the framework reveal an intended structure in the collection of models?

10.2 Results

In the experiments reported below, the computation time was limited with a timeout of 20 minutes but most of the problems were solved in less than 5 seconds on standard PC quadcore at 2.8 GHz.

These results where first published in [31].

10.2.1 Mapk models

The matchings found between the models of the MAPK cascade are depicted in Fig. 10.1. This class contains the family of models of [51] numbered 26 to 31. The reductions found automatically among these models are interesting for checking whether the formalism is faithful to biological reasoning, since the authors describe refinements between them. The models are of different sizes but always consider only one level of the traditional three levels of the MAPK cascade.

In this family, models 27, 29 and 31 are the simpler ones: they have few molecules because the catalyses are represented with only one reaction. The epimorphism exhibited

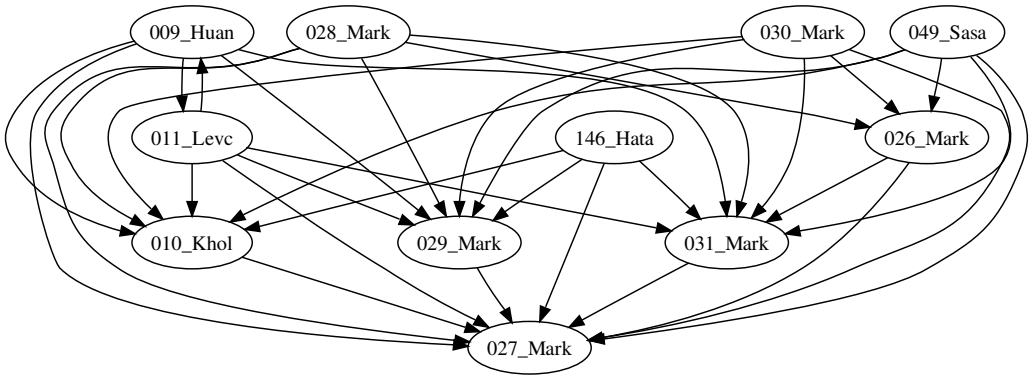


Figure 10.1: Matchings found between all models of the MAPK cascade (Schoeberl's model 14 and Levchenko's model with scaffold 19 are not represented here, they do not map each other but can be mapped to small models).

from model 31 to 27 corresponds to the splitting of two variants of MAPKK in 31. Model 29 distinguishes between the sites of phosphorylation of Mp, yielding a model with two molecules MpY and MpT. The subgraph epimorphism found from 29 to 27 corresponds to the deletion of one variant of Mp. Conversely, this distinction prevents the existence of an epimorphism from 31 or 27 to 29.

Models 26, 28 and 30 have more detailed catalyze mechanisms and differ as previously by the phosphorylation sites of Mp.

However, some epimorphisms from big models to small ones may have no biological meaning. This comes from the absence of constraint on the vertices that can be merged, and the relatively high number of arcs in Markevich's small models where most molecules are catalysts. Still, model 26 (with non-differentiated Mp) does not reduce to model 29 since that model indeed distinguishes MpY and MpT variants.

Now, concerning 3-step MAPK cascade models, the models 9 and 11 of [37] and [48] respectively are detected as isomorphic. Indeed, they only differ by molecule names and parameter values. They do not reduce to 28 and 30, which are models that do not differentiate sites of phosphorylation. They do not reduce to 26 either, which uses a more detailed mechanism for dephosphorilations.

Model 10 is another 3-step MAPK with no catalysts for dephosphorilations. It has the particularity to be cyclic, that is, the last level's most phosphorylated molecule catalyzes the phosphorylations of the first level. This is shown here as a reduction of the previous models obtained by merging the output of the third level with the catalyst of the first level.

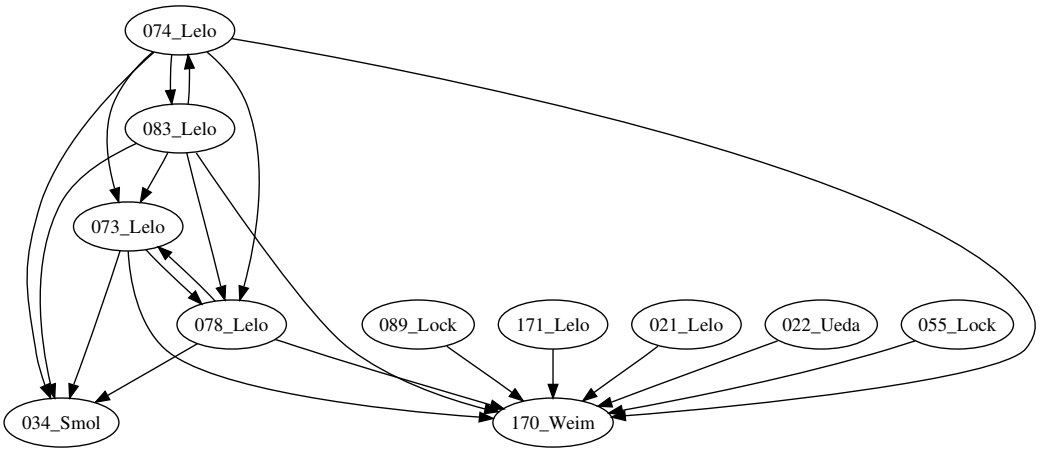


Figure 10.2: Matchings between the models of the circadian clock.

Finally, models 49 and 146 are bigger than the others and can easily be matched by them, and there were some comparisons for which no result was found before the timeout.

10.2.2 Circadian clock models

The matchings found in the class of circadian clock models are depicted in Fig. 10.2. Models 16, 24, 25 and 36 being very small oscillators were matched by most of other models, and for that reason were left out from the picture.

Let us first have a look at the isomorphisms found.

Models 73 and 78 are isomorphic. This is in accordance with the fact that these quite detailed models come from [47] and differ indeed by parameter values.

Models 74 and 83 are isomorphic too. They also correspond to two versions of a second model from the same article, but this time with the addition of the Rev-Erbα loop, greyed out in Fig. 1 of [47]. The authors explain *"Taking into account explicitly the role of REV-ERBα in the indirect negative feedback exerted by BMAL1 on the expression of the Bmal1 gene requires an extension of the model, which is now governed by 19 instead of 16 kinetic equations"*. The mapping to the previous models is automatically detected in accordance with these explanations, by merging the three new species (Rev-Erbα mRNA, protein in the cytoplasm and protein in the nucleus named Mr, Rc and Rn in model 74) to Bmal1 in the nucleus (named Bn in model 73).

Model 34 [61] is a quite small model of the Drosophila's circadian clock. The fact that its structure is included in that of the mammalian clock of the above models is in

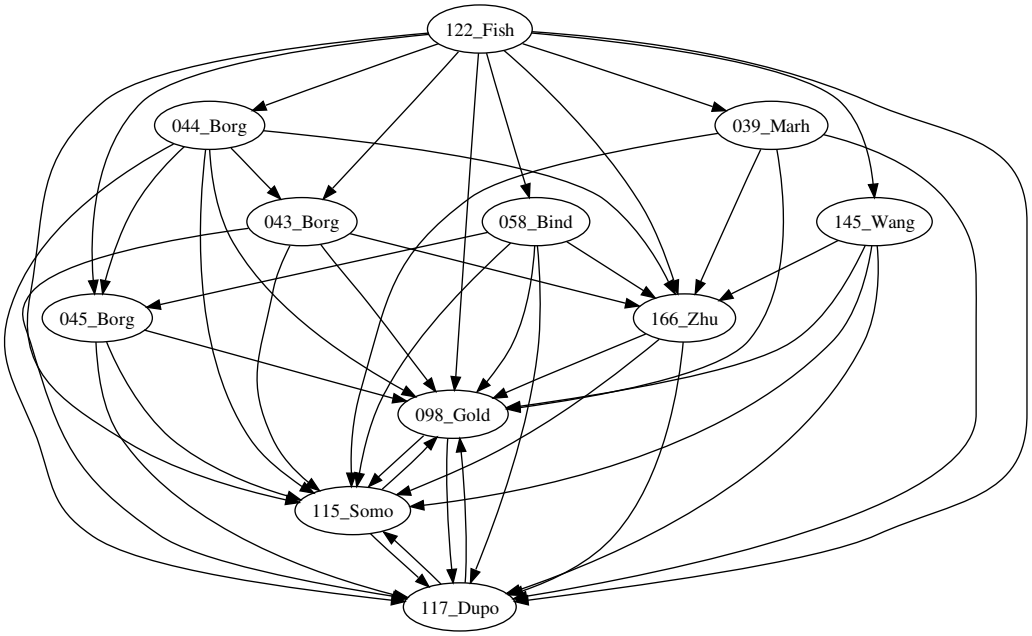


Figure 10.3: Subgraph epimorphisms for models of Calcium Oscillations from BioModels.

accordance with the fact those models were built on top of knowledge from the Drosophila [34] with a similar clock mechanism.

Models 171 [45] presents a model for the Drosophila, including Per/Tim (with two levels of phosphorylation) and the complex. Model 21 [46] actually studies the same model, unfortunately a different encoding in SBML (variable parameters instead of species for instance) makes it impossible to find a matching.

Many models map to model 170 [4] which focusses on the positive feedback loop of the circadian cycle oscillator. It is quite small but has two compartments, which explains why only 34 cannot be reduced to it. Model 22 [66] is a quite detailed model that focusses on the interlocked feedback loops, which can be mapped to 170 but not 34. Models 55 [50] and 89 [49] are both from Locke and others and about the circadian clock of Arabidopsis but include, in one case light induction, and in the other a new feedback loop. This explains why they do not give any matching either, except to the small oscillator model 170.

10.2.3 Calcium oscillation models

Fig. 10.3 shows that many models of calcium oscillation are connected.

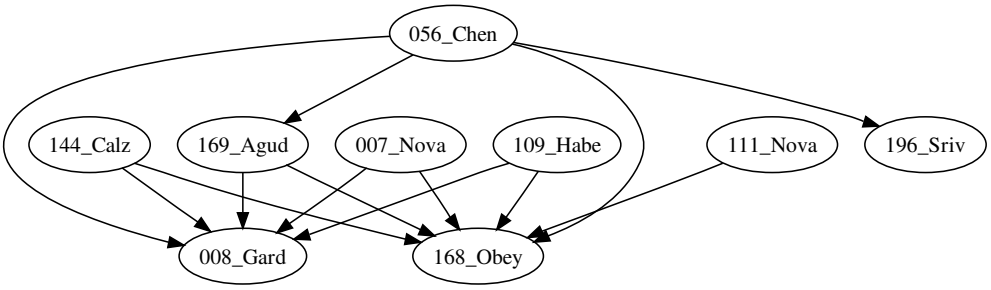


Figure 10.4: SEPI for some models of the cell cycle.

Models 98, 115 and 117 are in fact isomorphic due to their very small size (only two species) and differ only by their kinetics. There is a morphism from model 166 to them in accordance to the addition of a third species in this model where Ca2+ oscillations are seen as a mediator of genetic expression.

Models 43, 44 and 45 all relate to three different models from the same article [8]. Model 43 is the "basic one pool" model and there is a match from 44, the "1-pool model with IP3 degradation" since the latter is indeed a refinement of the former. The morphisms from 43 and 44 to 166 correctly exhibit the inclusion of the basic three-element oscillator in those models. A false positive morphism is found however from 44 to 45, the "2-pool model". This morphism is purely formal and has no biological meaning. It could be eliminated by using annotations as further constraints, for instance by taking into account the references to UniProt/KEGG or ChEBI databases that are already present in some SBML models.

Model 122 [26] is actually a big model of NFAT and NFκB with a side Calcium oscillator. It includes however many reversible reactions and thus structurally maps to all of the other models of this class.

Model 58 is a coupled oscillator version which interestingly maps to the "2-pool" oscillator of[8] by merging some components of the two oscillators into one.

Finally, models 39, related to mitochondria, and 145, related to ATP-induced oscillations, only map the small oscillators already described.

10.2.4 Cell cycle models

The reaction graphs of the cell cycle models are plagued by a common problem: these models originate from ODE models and the reaction graphs extracted from their encoding in SBML format does not correctly represent the structure of these models. It is thus hard to make sense of mappings between such graphs. For instance, the graphs of models 7, 8

and 56 are disconnected. Models 111, 144 and 196 have ghost molecules, that is, molecules which appear in the kinetics but not in the stoichiometry.

Nevertheless, models 144, 56 and 109 are relatively big with more than 50 reactions, and map easily on smaller models. Actually, there are 16 comparisons missing from this graph, and 13 are comparisons from these bigger graphs to the smaller ones.

Models 8, 168 and 196 are small (less than 15 reactions), which make them easy to match to, excepted for 196, which has a big diameter. There is no matching from 111 to 8 however. This is explained by the erroneous structure of 8 which is disconnected.

10.2.5 Negative control

For the sake of completeness of the evaluation of our method, the reduction relations between all pairs of models of the BioModels repository have been computed (with a time out of 20 minutes per problem).

Some matchings between unrelated model classes were found. These false positive matchings typically arise with small models that formally appear as reductions of large models without any biological meaning, for the same reasons as in the cases discussed above within a same class. These false positives arise in less than 9% of the total inter-class pairs, and in 1.2% of the tests after the removal of the small models.

Chapter 11

Conclusion

This works aimed to see model reduction not as a process taking a single model to a smaller one, but as a relation between models.

Instead of the local view entailed by the "process" view of model reduction, the relational view immediately yields a partial order on the set of all models, which raises partial order oriented questions about the structure of model reduction: what about comparability, distance, meets, joins?

11.1 Findings

In order to answer these questions, we developed a theory based on the structure of systems biology models, and capture model reduction by graph operations. We added a merge operation to the obvious delete operation, which allowed us to capture model reductions.

We wanted to be able to use this framework to handle real systems biology models, and since the merge/delete framework has been little studied previously, creating algorithms to answer the partial order questions first needed theoretical foundations.

We showed that graph morphisms allow the removal of symmetries in merge/delete chains; that least upper bounds (lub) and greatest lower bounds (glb) always exist in cases that interest us, but are usually not unique and not even of the same size; that the natural distance on the partial order can be computed by finding a glb of maximal size, and that the partial order is not a well-quasi order.

On the computational side, we showed that the comparability of two graphs is an NP-complete problem, even if graphs are bipartite as in our application, this also puts glb and lub computations in the NP-hard territory; we developed solutions to decide comparability and compute glb and lub, which, while too hard on larger graphs, proves to be feasible on smaller ones: the comparability problem proves to be relatively easy in practice.

We linked the graph theory with the model reduction application in theory by showing that merge/delete do capture model reductions, and in practice by showing that the graph

comparability problem approximates model reduction quite accurately. In order to do so, we also contributed to the routines used for extraction of reaction graphs in Biocham (to be specific, the hidden molecule inference).

The framework showed that on the BioModels repository of systems biology models, the SEPI partial order is a good approximation of the model reduction order between models.

11.2 Perspectives

This work is an end-to-end exposition from a theory on graphs to its application in systems biology. The main contribution of this work is to give a framework to allow a partial order view of model reduction. The particular choices made at each step can be reworked:

Using reaction graphs to approximate reaction models. Reactions graphs seem to have the minimal amount of structure that should feature in a reaction model relaxation. We could have used a more accurate relaxation, such as Petri Nets (reaction graphs with stoichiometry).

Choice of Graph Operations. We tried some restrictions on the graph operations we showed, to make the framework more accurate and the computations faster, but found out that we lost reductions that we wanted to capture.

If we keep the idea that reductions should be composable, an approximation by graph operations looks like a good choice, but is there a better set of operations that would be more accurate, easier to explore computationally?

NP-Hardness. We proved that even for bipartite graphs, the decision problem for SEPI comparability is NP-complete. However, reaction graphs coming from real models usually have a particular structure, e.g. low degree, connectedness, low treewidth [52]. Are the problems we presented tractable for some interesting families of graphs?

Constraint Model We used GNU-Prolog for our Constraint Programming model, because Biocham is coded with it. SWI-Prolog was also tried and proved competitive, despite its particular integer domain representation. This is because we could use an AC version of the alldifferent constraint in SWI-Prolog on antecedent variables, which is absent in GNU-Prolog. Indeed, solver features guide modelling: we could have used or developed more elaborate constraints if the solver allowed it, if [70] describes a global constraint for SISO, why not develop one for SEPI?

The computational framework has been developed with composability in mind, i.e. the possibility to add unrelated constraints to a problem. This raises a last question:

Application-specific constraints. We proposed models to decide comparability and compute glbs/lubs, but what if some additional constraint has to be met? One could forbid some vertices to be merged or deleted, for instance. The label morphism framework allows some expressivity in these matters, but it may be expanded.

Bibliography

[1] R. Ambauen, Stefan Fischer, and Horst Bunke. Graph edit distance with node splitting and merging. In *IAPR Workshop on Graph-based Representation in Pattern Recognition*, volume 2726 of *Lecture Notes in Computer Science*, pages 95–106. Springer-Verlag, 2003.

[2] D. Angeli, P. De Leenheer, and E. D. Sontag. A petri net approach to persistence analysis in chemical reaction networks. In *Biology and Control Theory: Current Challenges*, volume 357 of *LNCIS*, pages 181–216. Springer-Verlag, 2007.

[3] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *IJCAI*, volume 9, pages 399–404, 2009.

[4] Sabine Becker-Weimann, Jana Wolf, Hanspeter Herzel, and Achim Kramer. Modeling feedback loops of the mammalian circadian oscillator. *Biophysical journal*, 87(5):3023–3034, November 2004.

[5] Gilles Bernot, Jean-Paul Comet, Adrien Richard, and J. Guespin. A fruitful application of formal methods to biological regulatory networks: Extending thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347, 2004.

[6] Christian Bessiere, George Katsirelos, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. Decompositions of all different, global cardinality and related constraints. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI'09*, pages 419–424, 2009.

[7] A. et al Biere. *Handbook of Satisfiability*. Frontiers in artificial intelligence and applications. IOS Press, 2009.

[8] J. M. Borghans, G. Dupont, and A. Goldbeter. Complex intracellular calcium oscillations. a theoretical exploration of possible mechanisms. *Biophysical chemistry*, 66(1):25–41, May 1997.

[9] Horst Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[10] Laurence Calzone, Nathalie Chabrier-Rivier, François Fages, and Sylvain Soliman. Machine learning biochemical networks from temporal logic properties. In Gordon Plotkin, editor, *Transactions on Computational Systems Biology VI*, volume 4220 of *Lecture Notes in BioInformatics*, pages 68–94. Springer-Verlag, November 2006. CMSB'05 Special Issue.

[11] Laurence Calzone, François Fages, and Sylvain Soliman. BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.

[12] Nathalie Chabrier-Rivier, Marc Chiaverini, Vincent Danos, François Fages, and Vincent Schächter. Modeling and querying biochemical interaction networks. *Theoretical Computer Science*, 325(1):25–44, September 2004.

[13] Pierre-Antoine Champin and Christine Solnon. Measuring the similarity of labeled graphs. In *ICCBR 2003*, LNAI 2689, pages 80–95. Springer, 2003.

[14] Claudine Chaouiya, Elisabeth Remy, and Denis Thieffry. Petri net modelling of biological regulatory networks. *Journal of Discrete Algorithms*, 6(2):165–177, June 2008.

[15] Michael Codish and Moshe Zazon-Ivry. Pairwise cardinality networks. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 154–172. Springer, 2010.

[16] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971.

[17] Daniel Diaz, Salvador Abreu, and Philippe Codognet. On the implementation of GNU Prolog. *Theory and Practice of Logic Programming*, 12(1-2):253–282, 2012.

[18] Peter Dittrich and Pietro di Fenizio. Chemical organisation theory. *Bulletin of Mathematical Biology*, 69(4):1199–1231, April 2007.

[19] Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26, 2006.

[20] Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, José Meseguer, and M. Kemal Sönmez. Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the seventh Pacific Symposium on Biocomputing*, pages 400–412, January 2002.

[21] Bard Ermentrout. *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students*. SIAM, Philadelphia, 2002.

[22] François Fages, Steven Gay, and Sylvain Soliman. Inferring reaction models from ODEs. In *CMSB'12: Proceedings of the tenth international conference on Computational Methods in Systems Biology*, volume 7605 of *Lecture Notes in BioInformatics*, pages 370–373. Springer-Verlag, September 2012.

[23] François Fages and Sylvain Soliman. Formal cell biology in BIOCHAM. In M. Bernardo, P. Degano, and G. Zavattaro, editors, *8th Int. School on Formal Methods for the Design of Computer, Communication and Software Systems: Computational Systems Biology SFM'08*, volume 5016 of *Lecture Notes in Computer Science*, pages 54–80, Bertinoro, Italy, February 2008. Springer-Verlag.

[24] François Fages, Sylvain Soliman, and Nathalie Chabrier-Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4(2):64–73, October 2004.

[25] Martin Feinberg. Mathematical aspects of mass action kinetics. In L. Lapidus and N. R. Amundson, editors, *Chemical Reactor Theory: A Review*, chapter 1, pages 1–78. Prentice-Hall, 1977.

[26] Wayne G. Fisher, Pei-Chi C. Yang, Ram K. Medikonduri, and M. Saleet Jafri. Nfat and nfkappab activation in t lymphocytes: a model of differential activation of gene expression. *Annals of biomedical engineering*, 34(11):1712–1728, November 2006.

[27] T. S. Gardner, M. Dolnik, and J. J. Collins. A theory for controlling cell cycle dynamics using a reversibly binding inhibitor. *Proceedings of the National Academy of Sciences of the United States of America*, 95(24):14190–14195, November 1998.

[28] M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, New York, 1979.

[29] Steven Gay, François Fages, Thierry Martinez, Sylvain Soliman, and Christine Solnon. On the subgraph epimorphism problem. *Discrete Applied Mathematics*, 162:214–228, January 2014.

[30] Steven Gay, Thierry Martinez, Sylvain Soliman, and François Fages. A constraint program for subgraph epimorphisms with application to identifying model reductions in systems biology. In *Proceedings of the seventh Workshop on Constraint Based Methods for Bioinformatics WCB'11, colocated with CP 2011*, pages 59–66, September 2011.

[31] Steven Gay, Sylvain Soliman, and François Fages. A graphical method for reducing and relating models in systems biology. *Bioinformatics*, 26(18):i575–i581, 2010. special issue ECCB'10.

[32] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[33] Albert Goldbeter. A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *PNAS*, 88(20):9107–9111, 1991.

[34] Albert Goldbeter. A model for circadian oscillations in the drosophila period protein (per). *Proceedings. Biological sciences / The Royal Society*, 261(1362):319–324, September 1995.

[35] Eva Grafahrend-Belau, Falk Schreiber, Monika Heiner, Andrea Sackmann, Björn H. Junker, Stefanie Grunwald, Astrid Speer, Katja Winder, and Ina Koch. Modularization of biochemical networks based on a classification of petri net by T-invariants. *BMC Bioinformatics*, 9(90), February 2008.

[36] Pavol Hell and Jaroslav Nesetril. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[37] Chi-Ying Huang and James E. Ferrell, Jr. Ultrasensitivity in the mitogen-activated protein kinase cascade. *PNAS*, 93(19):10078–10083, September 1996.

[38] Christoph Kaleta, Stephan Richter, and Peter Dittrich. Using chemical organization theory for model checking. *Bioinformatics*, 25(15):1915–1922, 2009.

[39] G. Koh, H.F.C. Teong, M.-V. Clement, D. Hsu, and P.S. Thiagarajan. A decompositional approach to parameter estimation in pathway modeling: a case study of the akt and mapk pathways and their crosstalk. *Bioinformatics*, 22(14):e271–e280, 2006.

[40] Miyuki Koshimura, Tong Zhang, Hiroshi Fujita, and Ryuzo Hasegawa. QMaxSAT: A partial max-SAT solver system description. *Journal on Satisfiability, Boolean Modeling and Computation*, 8:95–100, 2012.

[41] S.M. Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer, 1998.

[42] Vianney le Clément, Yves Deville, and Christine Solnon. Constraint-based graph matching. In *15th International Conference on Principles and Practice of Constraint Programming (CP 2009)*, volume 5732 of *Lecture Notes in Computer Science*, pages 274–288, Lisbon, Portugal, 2009. Springer-Verlag.

[43] Nicolas le Novère, Benjamin Bornstein, Alexander Broicher, Mélanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, Jacky L. Snoep, and Michael Hucka. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acid Research*, 1(34):D689–D691, January 2006.

[44] Nicolas le Novère et al. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acid Research*, 1(34):D689–D691, January 2006.

[45] Jean-Christophe Leloup and Albert Goldbeter. A model for circadian rhythms in drosophila incorporating the formation of a complex between the per and tim proteins. *Journal of biological rhythms*, 13(1):70–87, February 1998.

[46] Jean-Christophe Leloup and Albert Goldbeter. Chaos and birhythmicity in a model for circadian oscillations of the per and tim proteins in drosophila. *Journal of Theoretical Biology*, 198:445–449, 1999.

[47] Jean-Christophe Leloup and Albert Goldbeter. Toward a detailed computational model for the mammalian circadian clock. *Proceedings of the National Academy of Sciences*, 100:7051–7056, 2003.

[48] Andre Levchenko, Jehoshua Bruck, and Paul W. Sternberg. Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. *PNAS*, 97(11):5818–5823, May 2000.

[49] James C. Locke, László Kozma-Bognár, Peter D. Gould, Balázs Fehér, Eva Kevei, Ferenc Nagy, Matthew S. Turner, Anthony Hall, and Andrew J. Millar. Experimental validation of a predicted feedback loop in the multi-oscillator clock of arabidopsis thaliana. *Molecular systems biology*, 2, 2006.

[50] James C. Locke, Megan M. Southern, László Kozma-Bognár, Victoria Hibberd, Paul E. Brown, Matthew S. Turner, and Andrew J. Millar. Extension of a genetic network model by iterative experimentation and mathematical analysis. *Molecular systems biology*, 1(1):msb4100018–E1–msb4100018–E9, June 2005.

[51] Nick I. Markevich, Jan B. Hoek, and Boris N. Kholodenko. Signaling switches and bistability arising from multisite phosphorylation in protein kinase cascades. *Journal of Cell Biology*, 164(3):353–359, February 2005.

[52] Faten Nabli, François Fages, Thierry Martinez, and Sylvain Soliman. A boolean model for enumerating minimal siphons and traps in petri-nets. In *Proceedings of CP'2012, 18th International Conference on Principles and Practice of Constraint Programming*, volume 7514 of *Lecture Notes in Computer Science*, pages 798–814. Springer-Verlag, October 2012.

[53] Faten Nabli and Sylvain Soliman. Steady-state solution of biochemical systems, beyond S-Systems via T-invariants. In Paola Quaglia, editor, *CMSB'10: Proceedings of the 8th International Conference on Computational Methods in Systems Biology*, pages 14–22. CoSBi, ACM, October 2010.

[54] Venkatramana N. Reddy, Michael L. Mavrovouniotis, and Michael N. Liebman. Petri net representations in metabolic pathways. In Lawrence Hunter, David B. Searls, and Jude W. Shavlik, editors, *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 328–336. AAAI Press, 1993.

[55] Neil Robertson and Paul D. Seymour. Graph minors. i. excluding a forest. *J. Comb. Theory, Ser. B*, 35(1):39–61, 1983.

[56] Neil Robertson and Paul D. Seymour. Graph minors .xiii. the disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

[57] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory*, Series B 92(2):325–357, 2004.

[58] Christian Rohr, Wolfgang Marwan, and Monika Heiner. Snoopy - a unifying petri net framework to investigate biomolecular networks. *Bioinformatics*, 26(7):974–975, 2010.

[59] Stefan Schuster, David A. Fell, and Thomas Dandekar. A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nature Biotechnology*, 18:326–332, 2002.

[60] Guy Shinar and Martin Feinberg. Structural sources of robustness in biochemical reaction networks. *Science*, 327(5971):1389–1391, 2010.

[61] Paul Smolen, Paul E. Hardin, Brian S. Lo, Douglas A. Baxter, and John H. Byrne. Simulation of drosophila circadian oscillations, mutations, and light responses by a model with vri, pdp-1, and clk. *Biophysical journal*, 86(5):2786–2802, May 2004.

[62] Sylvain Soliman. Finding minimal P/T-invariants as a CSP. In *Proceedings of the fourth Workshop on Constraint Based Methods for Bioinformatics WCB'08, associated to CPAIOR'08*, May 2008.

[63] Sylvain Soliman. Invariants and other structural properties of biochemical models as a constraint satisfaction problem. *Algorithms for Molecular Biology*, 7(15), May 2012.

[64] Christine Solnon. AllDifferent-based filtering for subgraph isomorphism. *Artificial Intelligence*, 174(12-13):850–864, August 2010.

[65] René Thomas. Regulatory networks seen as asynchronous automata : a logical description. *Journal of Theoretical Biology*, 153:1–23, 1991.

[66] H. R. Ueda, M. Hagiwara, and H. Kitano. Robust oscillations within the interlocked feedback model of drosophila circadian rhythm. *Journal of theoretical biology*, 210(4):401–406, June 2001.

[67] A. Varma and B.O. Palsson. Metabolic flux balancing: basic concepts, scientific and practical use. *Nature Biotechnology*, 12(10):994–998, 1994.

[68] Narayan Vikas. *Computational Complexity of Graph Compaction*. PhD thesis, Simon Fraser University, August 1997.

[69] Toby Walsh. Symmetry breaking using value precedence. In *ECAI*, pages 168–, 2006.

[70] Stéphane Zampelli, Yves Deville, and Christine Solnon. Solving subgraph isomorphism problems with constraint programming. *Constraints*, 15(3):327–353, 2010.

[71] Ionela Zevedei-Oancea and Stefan Schuster. Topological analysis of metabolic networks based on petri net theory. *In Silico Biology*, 3(29), 2003.