HAL
open science

Computing Difference Abstractions of Metabolic Networks Under Kinetic Constraints

Emilie Allart, Cristian Versari, Joachim Niehren

To cite this version:

Emilie Allart, Cristian Versari, Joachim Niehren. Computing Difference Abstractions of Metabolic Networks Under Kinetic Constraints. CMSB 2019 - 17th International Conference on Computational Methods in Systems Biology, Luca Bortolussi; Guido Sanguinetti, Sep 2019, Trieste, Italy. pp.266-285, 10.1007/978-3-030-31304-3_14 . hal-02302463

HAL Id: hal-02302463

https://hal.science/hal-02302463v1

Submitted on 1 Oct 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

Computing Difference Abstractions of Metabolic Networks Under Kinetic Constraints

Emilie Allart[1,2], Joachim Niehren[1,3], and Cristian Versari[1,2]

[1] BioComputing Team, CRIStAL Lab, Lille
[2] Université de Lille
[3] Inria Lille

Abstract. Algorithms based on abstract interpretation were proposed recently for predicting changes of reaction networks with partial kinetic information. Their prediction precision, however, depends heavily on which heuristics are applied in order to add linear consequences of the steady state equations of the metabolic network. In this paper we ask the question whether such heuristics can be avoided while obtaining the highest possible precision. This leads us to the first algorithm for computing the difference abstractions of a linear equation system exactly without any approximation. This algorithm relies on the usage of elementary flux modes in a nontrivial manner, first-order definitions of the abstractions, and finite domain constraint solving.

Keywords. Gene knockout prediction, reaction networks, constraints, systems biology, synthetic biology, metabolism.

1 Introduction

Flux balance analysis [16,17] can be used to predict the effect of influx changes of metabolic networks at steady state. Such predictions can be based on reasoning with linear equations systems that describe the rates of the reactions in a steady state of the metabolic network, by using Gaussian elimination, elementary flux modes (EFMs) [13], or optimisation methods [14,6]. Most importantly, precise quantitative kinetic information is not required in contrast to classical mathematical analysis methods for reaction networks [9,2,11,7]. In fact, even when the kinetic functions associated to chemical reactions are known, the values of rate constants are most often missing, since it is difficult to measure them experimentally in the precise state of the regulation of the metabolic network at the time point of interest.

Recently, abstract interpretation [3,8,5] has been exploited to design novel algorithms [15,4,12] that can use partial kinetic information beneficially for predicting changes of metabolic networks. They can in particular exploit the knowledge about the enzymes and inhibitors. Similarly to flux balance analysis, the linear equations describing steady states are used, but in addition to them, kinetic constraints are inferred from the partial kinetic information of inhibitors and enzymes.

The steady state equations and the kinetic constraints enable gene knockout predictions based on abstract interpretation (in the finite relational structure Δ_6), based on the linear equations from the metabolic network and the constraints on its regulatory control. The unknown kinetic parameters are abstracted away, by interpretation over some finite relational structure, that contains a finite number of abstract differences rather than concrete differences in $\mathbb{R}^2_+$. Eventually, the prediction algorithm will apply a finite domain constraint solver for Δ_6 that we implemented in Minizinc [18] to enumerate all the changes that may or must lead to the target change.

The prediction quality of abstract interpretation approaches heavily depends on heuristics that find and add linear equations entailed by the steady state equations before constraint solving over Δ_6. This is necessary to enable global reasoning, since local reasoning alone is not able to

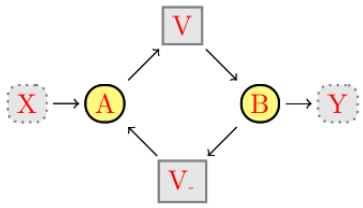


Fig. 1. A toy metabolic network with a simple cycle.

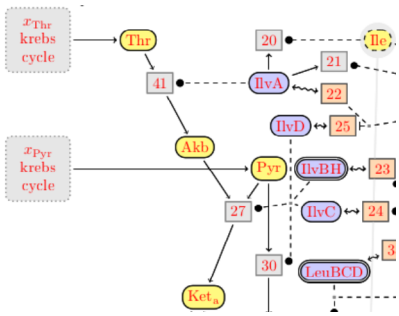


Fig. 2. A glimpse of the formal model from [4] of leucine production in *B. subtilis*.

deal with cycles in metabolic part of the network as we will illustrate in Section 2. On the other hand, it is impossible to add the infinity of all entailed linear equations before abstract interpretation. Therefore, these algorithms can at best approximate the abstraction of differences of solution set of linear equations. Whether this abstraction can be computed exactly is a long standing open problem, as well as how to measure the quality of approximation heuristics.

In this paper, we present the first exact algorithm that can compute the Δ_6 difference abstraction of the solution set of a linear equation system without any overapproximation. We apply it to the prediction of leucine overproduction, a benchmark task that is best studied with abstract interpretation. In this case, we need to deal with kinetic constraints in addition that are naturally interpreted over Δ_6. It turns out that a new heuristic that we also propose in the present paper does indeed compute well the difference abstraction at this benchmark task from systems biology, although being inexact in the general case. The main advantage of this heuristic is that it outperforms the exact algorithm dramatically in computation time: only 5 minutes are needed for the knockout prediction rather than 5 hours with the always exact algorithm.

2 Qualitative reasoning on metabolic reaction networks

The application of abstract interpretation to the analysis of metabolic reaction networks is based on an intuitive qualitative reasoning. Its aim is to predict how a living organism or its environment should be changed in order to maximize the production of some metabolite of interest, without exact knowledge of the quantitative parameters of the system. A change of the organism is represented for example by any modification of its genome, such as a gene knockout. A change of the environment is typically represented by the modification of the culture medium, which results in an increase or decrease of some inflows. Since changes can be arbitrarily combined to obtain or improve the wanted results, the problem that we tackle is highly combinatorial.

Examples of formal metabolic networks on which abstract interpretation can be applied are shown in Fig. 1 and Fig. 2. Fig. 1 shows a toy metabolic network with a simple cycle that will be used in the following to introduce the key ideas of the reasoning. In this network, metabolites are displayed in yellow rounded boxes, while reactions are in gray squared boxes, in the tradition of Petri nets. Reactions with dotted contour are inflows or outflows of the system.

Fig. 2 shows a glimpse of a bigger metabolic reaction network with regulation, where the regulatory part is represented by the enzymes in blue rounded boxes. The full network of Fig. 2 models a part of the metabolism of the gram positive bacterium *B. subtilis*. Our benchmark application – taken from [4] – is the overproduction of one of the metabolites of this network, the branched chain amino acid Leucine (Leu). This amino acid is a precursor of surfactin, a non ribosomal peptide with several applications in food and pharmaceutical industry.

Let us now reconsider the toy model with a simple cycle in Fig. 1. This network is composed of two chemical species A and B. The species A is continuously produced by an inflow at a fixed rate X, and is transformed into B by the reaction with rate V. The inverse reaction with rate $V_{\text{-}}$ transforms B back into A. The species B has an outflow with rate Y. All reactions but the inflow are controlled internally by the system. The outflow rate Y in particular is determined by the concentration of B which in turn depends on the rates V and $V_{\text{-}}$.

The only possible change in this toy network is the increase or decrease of the inflow X. In order to illustrate the reasoning method, we set the increase of Y as our final target. As usually done in flux balance analysis [16,17], we consider the system at the steady state, that gives us the following linear system of equations:

$$\exists V \exists V_{\text{-}}. \quad V = X + V_{\text{-}} \quad \wedge \quad V = Y + V_{\text{-}} \tag{1}$$

The existential quantifiers for V and $V_{\text{-}}$ allow us to hide the internal behaviour of the network, so to project to inflows and outflows. While the consequences of steady state equations may be difficult to interpret without using Gauss' algorithm, in this particular case it is easy to see that by subtracting the first equation in (1) from the second we obtain an important relation:

$$X = Y \tag{2}$$

Eq. (2) tells us that the only way for Y to increase is that X increases too. We formalize now this intuition by means of abstract interpretation, this time applied to *concrete differences* in $\mathbb{R}^2_+$. Concrete differences capture the essence of a change in the spirit of [15]: a change of the value of X for instance can be thought of as a pair of positive reals $(r^{\text{before}}, r^{\text{after}})$ representing the value of X at steady state respectively *before* and *after* the modification of the environment. We need to consider positive reals since the rates of irreversible reactions are positive. We call the above concrete differences an *increase* if $r^{\text{before}} < r^{\text{after}}$, a *decrease* if $r^{\text{before}} > r^{\text{after}}$ and a no-change if $r^{\text{before}} = r^{\text{after}}$. This intuition motivates the usage of abstract values in $\Delta_3 = \{\triangle, \triangledown, \approx\}$ where $\triangle = (0, 1)$ represents an increase, $\triangledown = (1, 0)$ a decrease, and $\approx = (0, 0)$ a no-change. The canonical mapping of concrete differences in $\mathbb{R}^2_+$ to the abstract differences in Δ_3 can be seen as a homomorphism between the relational structures $\mathbb{R}^2_+$ to Δ_3. This algebraic view of abstractions as homomorphisms enables various generalizations. An example is the abstraction from $\mathbb{R}^2_+$ to Δ_6 – as considered for gene knockout prediction [12,4] – which refines each of the three $\mathbb{R}^2_+$ equivalence classes produced by Δ_3 into two parts in Δ_6.

Any abstraction of concrete differences enables some form of *abstract qualitative reasoning* [10] based on operations of the relational structure of abstract differences, that can be used for change prediction in systems biology. Let us illustrate how to reason with Δ_3. As a first example, assume that we know for some reason that X and $V_{\text{-}}$ both increase, that is $X = \triangle$ and $V_{\text{-}} = \triangle$. Then we can use the first equation in (1) to deduce for sure V will also increase, since $\triangle +^{\Delta_3} \triangle = \triangle$. The full table defining the summation operator $+^{\Delta_3}$ on abstract difference is given in Fig. 3.

The above qualitative reasoning method, however, is quite weak when relying only on the steady state equations in (1). The main reason is that all reasoning steps are local, so that they overlook global properties of the network that are arising for example with metabolic cycles. To see this, suppose that we want to predict which environmental change may lead to an increase of Y. We can use the second equation in (1) to infer some constraints on the values of V and $V_{\text{-}}$: if $Y = \triangle$, we can for example infer that V cannot be $\triangledown$ if $V_{\text{-}}$ is $\triangle$. In fact, $\triangle + \triangle$ can never be equal to $\triangledown$. However, $V = \triangle, V_{\text{-}} = \triangledown$ is a partial solution that seems *compatible* with an increase of Y. This illustrates that we cannot infer any constraint on X with this kind of local reasoning.

Equation $X = Y$ in Eq. (2), in contrast, expresses a global property of the network that immediately implies that the only value for X compatible with an increase of Y is $\triangle$. In other

words, when reasoning with equations over abstract differences, Eq. (2) is *no longer an implicit consequence* of the system (1). Therefore, to be taken into account, such equations must be explicitly included in the system *before* applying the abstract reasoning. Unfortunately, the number of entailed linear equations is infinite in general. For instance, in our small network all the equations of the family $nV + mX = nV_- + (m+1)Y$ for any two naturals n, m are consequences of (1). So, instead of trying to infer the set of *all* the consequences of our system, we may try to compute a "good" subset of it, by including only the consequences that more heavily constrain the variables. The prediction quality of the existing approaches heaviliy depends on the heuristics chosen for adding entailed linear consequences.

One evident advantage of Eq. (2) is its small number of variables (there are only two: X and Y). If we consider an individual linear equation, the intuition is that removing a variable increases the constraining power on the remaining variables. So, we propose as a heuristics the inclusion of all linear consequences involving a minimal subset of variables. This idea is at the core of the first result that we present in this paper: a heuristic algorithm that enriches the steady state equations of the metabolic network with minimal support consequences before applying abstract interpretation. While this algorithm has been internally used for some time to increase the precision of the abstract interpretation, some key questions have been always open about it:

1. is the set of minimal support equations *complete*? that is, does it represent all the deducible constraints on the abstract system?
2. are these constraints sufficient to compute the *exact* set of abstract solutions?
3. if this is not the case, is there a method to compute them?

The definitive answers to these questions is the second main contribution of this paper. We show in particular that the above heuristic *does not* cover all the entailed constraints on the abstract system, i.e. it does not allow to compute the exact abstraction of a linear system in general. Intuitively, this happens because the approach takes for granted that the abstract reasoning is based on the linear system computed at the steady state, that is on the matrix equation $A\mathbf{X} = 0$ where A is the stoichiometry matrix associated to the metabolic network, and $\mathbf{X}$ is the set of metabolic flows representing the unknowns of the linear system. However, it is easy to notice that as soon as concrete differences are introduced in the reasoning, there is not only one, but actually *two* linear systems to consider: one *before* the environmental change, and one *after* it, that is one system for each value of the pairs representing concrete changes. Informally, we should therefore consider a bigger matrix equation including somehow both $A\mathbf{X}^{\text{before}} = 0$ and $A\mathbf{X}^{\text{after}} = 0$.

This idea is the starting point of development of the main contribution of the present paper: a method for the exact computation of the abstraction of a linear system, that we call the *exact* algorithm. This method not only provides us with the counterexamples to the exactness of the heuristic based on minimal support consequences, but gives us also an exact measure of its goodness as well as of the goodness of all the other heuristics used to improve our abstract analysis. Remarkably, both the heuristic and the exact algorithm have their root in the rewriting of a linear system in terms of its EFMs. The key difference between the two methods lies in the choice of the linear system (i.e. the matrix equation) initially used to compute the EFMs.

3 Preliminaries

*Set Notation.*We start with usual notation for sets. Let $\mathbb{N}$ be the set of natural numbers and $\mathbb{R}_+$ the set of positive real numbers, both including 0. For any set A and $n \in \mathbb{N}$, the set of n-tuples of elements in A is denoted by A^n. The i-th projection function on n-tuples of elements in A, where $1 \leq i \leq n$ is the function $\pi_i : A^n \to A$ such that $\pi_i(a_1, \ldots, a_n) = a_i$ for all $a_1, \ldots, a_n \in A$. If A is finite the number of elements of A is denote by $|A|$.

Σ-Algebras and Σ-Structures. We next recall the notions of Σ-algebras, Σ-structures, and homomorphism between Σ-structures. Let $\Sigma = \cup_{n\geq 0} F^{(n)} \uplus C$ be a ranked signature. The elements of $f \in F^{(n)}$ are called the n-ary function symbols of Σ and the elements in $c \in C$ its constants.

Definition 1. *A Σ-algebra $S = (dom(S), .^S)$ consists of a set $dom(S)$ and an interpretation $.^S$ such that $c^S \in dom(S)$ for all $c \in C$, and $f^S : dom(S)^n \to dom(S)$ for all $f \in F^{(n)}$ and $n \in \mathbb{N}$.*

We next reinterpret n-ary function symbols of Σ as $n+1$-ary relation symbols, so that we can reuse the same signature Σ for defining Σ-structures.

Definition 2. *A Σ-structure $\Delta = (dom(\Delta), .^\Delta)$ consists of a set $dom(\Delta)$ and an interpretation $.^\Delta$ such that $c^\Delta \in dom(\Delta)$ for all $c \in C$ and $f^\Delta \subseteq dom(\Delta)^{n+1}$ for all $f \in F^{(n)}$ and $n \in \mathbb{N}$.*

In this manner, any Σ-algebra is also a Σ-structure since any n-ary function is an $n+1$-ary relation. Note also that symbols in $F^{(0)}$ are interpreted as monadic relations in Σ-structures, i.e., as subsets of the domain, in contrast to constants in C that are interpreted as elements of the domain.

Definition 3. *A homomorphism between two Σ-structures S and Δ is a function $h : dom(S) \to dom(\Delta)$ such that for $c \in C$, $n \in \mathbb{N}$, $f \in F^{(n)}$, and $s_1, \ldots, s_{n+1} \in dom(S)$:*

1. *$h(c^S) = c^\Delta$, and*
2. *if $(s_1, \ldots, s_{n+1}) \in f^S$ then $(h(s_1), \ldots, h(s_{n+1})) \in f^\Delta$.*

If we consider $n+1$-ary relations as n-ary set valued functions, the second condition can be rewritten equivalently as $h(f^S(s_1, \ldots, s_n)) \subseteq f^\Delta(h(s_1), \ldots, h(s_n))$. For Σ-algebras, this condition is equivalent to $h(f^S(s_1, \ldots, s_n)) = f^\Delta(h(s_1), \ldots, h(s_n))$.

4 Σ-Abstractions

Throughout the paper we will use the signature $\Sigma = F^{(2)} \uplus C$ with two binary function symbols in $F^{(2)} = \{+, *\}$ and two constants $C = \{0, 1\}$. We will only consider Σ-algebras S in which $+^S$ and $*^S$ are associative and commutative, with neutral element 0^S and 1^S respectively.

Example 4. The set of positive real numbers $\mathbb{R}_+$ can be turned into a Σ-algebra with domain $\mathbb{R}_+$, by interpreting $+$ as the addition of positive real numbers $+^{\mathbb{R}_+}$, $*$ as the multiplication of positive real numbers $*^{\mathbb{R}_+}$, and interpreting the constants by themselves $0^{\mathbb{R}_+} = 0$ and $1^{\mathbb{R}_+} = 1$. We will deliberatly confuse the set $\mathbb{R}_+$ with the Σ-algebra $(\mathbb{R}_+, .^{\mathbb{R}_+})$.

Example 5. The set of Booleans $\mathbb{B} = \{0, 1\} \subseteq \mathbb{R}_+$ can be turned into a Σ-algebra with domain $\mathbb{B}$ by interpreting $+^{\mathbb{B}} = \vee^{\mathbb{B}}$ as disjunction, $*^{\mathbb{B}} = \wedge^{\mathbb{B}}$ as conjunction, and the constants by themselves $0^{\mathbb{B}} = 0$ and $1^B = 1$. We will deliberatly confuse the set $\mathbb{B}$ with the Σ-algebra $(\mathbb{B}, .^{\mathbb{B}})$.

We can abstract positive real numbers into booleans by defining a function $h_{\mathbb{B}} : \mathbb{R}_+ \to \mathbb{B}$ such that $h_{\mathbb{B}}(0) = 0$ and $h_{\mathbb{B}}(r) = 1$ for all $r \in \mathbb{R}_+ \setminus \{0\}$.

Lemma 6. *The function $h_{\mathbb{B}} : \mathbb{R}_+ \to \mathbb{B}$ is a homomorphism between Σ-algebras.*

The homomorphism $h_{\mathbb{B}}$ is the prime example of what we will call a Σ-abstraction.

Definition 7. *A Σ-abstraction is a homomorphism between Σ-structures S and Δ such that $dom(\Delta) \subseteq dom(S)$.*

5 Abstracting Concrete Differences

Concrete differences are pairs of positive real numbers such as $(r^{\text{before}}, r^{\text{after}}) \in \mathbb{R}_+^2$ in the example Section 2. We show how to abstract concrete differences into abstract differences in some finite Σ-structure.

The Tuple Σ-Algebra S^n. For any Σ-algebra S and natural number $n \in \mathbb{N}$ we define the Σ-algebra of n-tuples $S^n = (dom(S)^n, .^{S^n})$ such that for all $s_1, \ldots, s_n, s'_1, \ldots, s'_n \in dom(S)$ and $\odot \in F^{(2)}$:

$$(s_1, \ldots, s_n) \odot^{S^n} (s'_1, \ldots, s'_n) = (s_1 \odot^S s'_1, \ldots, s_n \odot^S s'_n)$$

The constants $c \in C$ are interpreted as $c^{S^n} = (c^S, \ldots, c^S)$. Note that if 0^S is the neutral element of $+^S$, then 0^{S^n} is the also the neutral element of $+^{S^n}$. In analogy, if 1^S is the neutral element of $*^S$ then 1^{S^n} is also the neutral element of $*^{S^n}$. Furthermore, the associativity and commutativity of $+^{S^n}$ and $*^{S^n}$ inherit from $+^S$ and $*^S$ respectively.

Note that we deliberatly confuse the set $\mathbb{R}_+^2$ with the Σ-algebra $(\mathbb{R}_+^2, .^{\mathbb{R}_+^2})$ with our notation. Given this, it follows from the above, that the algebra $\mathbb{R}_+^2$ has the neutral element $(0,0)$ for $+^{\mathbb{R}_+^2}$ and the neutral element $(1,1)$ for $*^{\mathbb{R}_+^2}$, and that these operations are associative and commutative.

For any function $h : A \to B$ and $n \in \mathbb{N}$ we define the function $h^n : A^n \to B^n$ such that $h^n(a_1, \ldots, a_n) = (h(a_1), \ldots, h(a_n))$ for all $a_1, \ldots, a_n \in A$.

Lemma 8. *If h is Σ-abstraction from S to Δ then h^n is a Σ-abstraction from S^n to Δ^n.*

Abstractions of Concrete Differences. A generic manner to abstract concrete differences in $\mathbb{R}_+^2$ is to start with a finite set $\Delta \subseteq \mathbb{R}_+^2$ of so called *abstract differences*, and some function $h : \mathbb{R}_+^2 \to \Delta$ that says how to abstract any concrete differences to some abstract difference. The function h defines a partition of $\mathbb{R}_+^2$ into the equivalences classes of concrete differences that are mapped to the same abstract difference.

Given such a function h, there is a unique manner to define an interpretation $.^\Delta$ such that $(\Delta, .^\Delta)$ becomes Σ-structure and h a Σ-abstraction. For any constant $c \in C$ we have to define $c^\Delta = h(c^{\mathbb{R}_+^2})$ and for any function symbol $\odot \in F^{(2)}$ we have to define a ternary relation $\odot^\Delta$, which seen as set-valued function $\odot^\Delta : \Delta \times \Delta \to 2^\Delta$ must satisfy for all abstract values $\delta_1, \delta_2 \in \Delta$:

$$\delta_1 \odot^\Delta \delta_2 = \{h(r_1 \odot^{\mathbb{R}_+} r_2, r'_1 \odot^{\mathbb{R}_+} r'_2) \mid h(r_1, r'_1) = \delta_1,\ h(r_2, r'_2) = \delta_2\}$$

Lemma 9. *$h : \mathbb{R}_+^2 \to \Delta$ is a Σ-abstraction.*

The Σ-Structure Δ_3. Our next objective is to recall the abstraction of concrete differences into the finite Σ-structure with domain $\Delta_3 = \{\triangle, \triangledown, \approx\}$ that is well-known from qualitative reasonning (see e.g. [10]). For this we start with the function $h_{\Delta_3}(r, r') \in \Delta_3$ such that for any all $r, r' \in \mathbb{R}_+$:

$$h_{\Delta_3}(r, r') = \begin{cases} \triangledown = (1,0) \text{ if } r > r' \\ \triangle = (0,1) \text{ if } r < r' \\ \approx = (0,0) \text{ if } r = r' \end{cases}$$

The relations $+^{\Delta_3}$ and $*^{\Delta_3}$ are is the symmetric closure of the relation in Fig. 3. Furthermore, $h_{\Delta_3} : \mathbb{R}_+^2 \to \Delta_3$ is a Σ-abstraction by Lemma 9.

The Σ-Structure Δ_6. We next recall the abstraction of concrete differences to the finite Σ-structure with domain $\Delta_6 = \{\uparrow, \downarrow, \sim, \Uparrow, \Downarrow, \approx\}$ that was introduced for gene knockout prediction in [15]. For defining this Σ-structure, we start with the function $h_{\Delta_6} : \mathbb{R}_+^2 \to \Delta_6$ such that for any two numbers $r, r' \in \mathbb{R}_+$:

$$h_{\Delta_6}(r, r') = \begin{cases} \uparrow = (1,2) & \text{if } 0 \neq r < r' \\ \downarrow = (2,1) & \text{if } r > r' \neq 0 \\ \sim = (1,1) & \text{if } r = r' \neq 0 \end{cases} \qquad h_{\Delta_6}(r, r') = \begin{cases} \Uparrow = (0,2) & \text{if } 0 = r < r' \\ \Downarrow = (2,0) & \text{if } r > r' = 0 \\ \approx = (0,0) & \text{if } r = r' = 0 \end{cases}$$

δ	δ'	$\delta +^{\Delta_3} \delta'$	$\delta *^{\Delta_3} \delta'$
$\triangle$	$\triangle$	$\{\triangle\}$	$\{\triangle\}$
$\triangle$	$\triangledown$	$\{\triangle, \approx, \triangledown\}$	$\{\triangle, \approx, \triangledown\}$
$\triangle$	$\approx$	$\{\triangle\}$	$\{\triangle, \approx\}$

δ	δ'	$\delta +^{\Delta_3} \delta'$	$\delta *^{\Delta_3} \delta'$
$\approx$	$\approx$	$\{\approx\}$	$\{\approx\}$
$\triangledown$	$\triangledown$	$\{\triangledown\}$	$\{\triangledown\}$
$\triangledown$	$\approx$	$\{\triangledown\}$	$\{\triangledown, \approx\}$

c	c^{Δ_3}
0	$\approx$
1	$\approx$

Fig. 3. Interpretation of Σ-structure Δ_3.

δ	δ'	$\delta +^{\Delta_6} \delta'$	$\delta *^{\Delta_6} \delta'$
$\uparrow$	$\uparrow$	$\{\uparrow\}$	$\{\uparrow\}$
$\uparrow$	$\downarrow$	$\{\uparrow, \sim, \downarrow\}$	$\{\uparrow, \sim, \downarrow\}$
$\uparrow$	$\sim$	$\{\uparrow\}$	$\{\uparrow\}$
$\uparrow$	$\Uparrow$	$\{\uparrow\}$	$\{\Uparrow\}$
$\uparrow$	$\Downarrow$	$\{\uparrow, \downarrow, \sim\}$	$\{\Downarrow\}$
$\uparrow$	$\approx$	$\{\uparrow\}$	$\{\approx\}$
$\Uparrow$	$\downarrow$	$\{\uparrow, \sim, \downarrow\}$	$\{\Uparrow\}$

δ	δ'	$\delta +^{\Delta_6} \delta'$	$\delta *^{\Delta_6} \delta'$
$\Uparrow$	$\sim$	$\{\uparrow\}$	$\{\Uparrow\}$
$\Uparrow$	$\Uparrow$	$\{\Uparrow\}$	$\{\Uparrow\}$
$\Uparrow$	$\Downarrow$	$\{\uparrow, \sim, \downarrow\}$	$\{\approx\}$
$\Uparrow$	$\approx$	$\{\Uparrow\}$	$\{\approx\}$
$\sim$	$\sim$	$\{\sim\}$	$\{\sim\}$
$\sim$	$\approx$	$\{\sim\}$	$\{\approx\}$
$\sim$	$\downarrow$	$\{\downarrow\}$	$\{\downarrow\}$

δ	δ'	$\delta +^{\Delta_6} \delta'$	$\delta *^{\Delta_6} \delta'$
$\sim$	$\Downarrow$	$\{\downarrow\}$	$\{\Downarrow\}$
$\approx$	$\approx$	$\{\approx\}$	$\{\approx\}$
$\approx$	$\downarrow$	$\{\downarrow\}$	$\{\Downarrow\}$
$\approx$	$\Downarrow$	$\{\Downarrow\}$	$\{\Downarrow\}$
$\downarrow$	$\downarrow$	$\{\downarrow\}$	$\{\downarrow\}$
$\downarrow$	$\Downarrow$	$\{\downarrow\}$	$\{\Downarrow\}$
$\Downarrow$	$\Downarrow$	$\{\Downarrow\}$	$\{\Downarrow\}$

c	c^{Δ_6}
0	$\approx$
1	$\sim$

Fig. 4. Interpretation of Σ-structure Δ_6.

The relations $+^{\Delta_6}$ and $*^{\Delta_6}$ are the symmetric closure of the relations in Fig. 4. By Lemma 9, $h_{\Delta_6} : \mathbb{R}_+^2 \to \Delta_6$ is a Σ-abstraction.

6 First-Order Logic

We first recall the standard first-order logic and then show how to enhance it with n-tuples without increasing the expressiveness.

We fix a set of variables $\mathcal{V}$ (for instance $\mathcal{V} = \mathbb{N}$). The variables in $\mathcal{V}$ will be ranged over by x and y. The set of first-order expressions $e \in \mathcal{E}_\Sigma$ and first-order formulas $\phi \in \mathcal{F}_\Sigma$ are constructed according to the abstract syntax in Fig. 5 from the symbols in the signature Σ, the variables in $\mathcal{V}$, the first-order connectives, and the equality symbol $\doteq$. As shortcuts, we define the formula $\mathit{true} =_{\text{def}} 1 \doteq 1$ and for any sequence of formulas $\phi_1, \ldots, \phi_n$ we define $\wedge_{i=1}^n \phi_i$ as $\phi_1 \wedge \ldots \wedge \phi_n$ which is equal to true if $n = 0$. We define formulas $e \not\doteq 0$ by $\neg e \doteq 0$.

The semantics of a formula $\phi \in \mathcal{F}_\Sigma$ is a truth value, which depends on the Σ-structures S of interpretation and on a variable assignment $\alpha : \mathcal{V} \to \mathit{dom}(S)$. Any Σ-expressions $e \in \mathcal{E}_\Sigma$ denotes a subset of values in $\mathit{dom}(S)$, which will be singleton in case that S was a Σ-algebra. The semantic of equations $e \doteq e'$ is, as expected when interpreted over Σ-algebras S: the unique values of e and e' in S must be equal. However, we will also need to interpret equations $e \doteq e'$ over Σ-structures. This is why, any expression e denotes a subset of the Σ-structure, not just a single element. We can then interpret equality as nondisjointness, i.e., $e \doteq e'$ holds in a Σ-structure S if e and e' are interpreted as nondisjoint subsets of $\mathit{dom}(S)$.

A variable assignment into a Σ-structure S is a partial function $\alpha : V \to \mathit{dom}(S)$ for some subset $V \subseteq \mathcal{V}$. Let S be a Σ-structure and α a variable assignment to S. Any Σ-expression e with $\mathcal{V}(e) \subseteq V$ can be interpreted as an element of $\mathit{dom}(S)$ and any Σ-formula $\phi \in \mathcal{F}_\Sigma$ with $\mathcal{V}(\phi) \subseteq V$ as a Boolean value. The set of solutions of a formula $\phi \in \mathcal{F}_\Sigma$ over a Σ-structure S with respect to some set of variables $V \supseteq \mathcal{V}(\phi)$ is defined by:

$$\mathit{sol}_V^S(\phi) = \{\alpha : V \to \mathit{dom}(S) \mid [\![\phi]\!]^{S,\alpha} = 1\}$$

If $V = \mathcal{V}(\phi)$ then we omit the index V, i.e., $\mathit{sol}^S(\phi) = \mathit{sol}_V^S(\phi)$.

We next extend the first-order logic to n-tuples where the parameter n is fixed. In applications, we will use the case $n = 2$, that is the first-order logic with pairs. Back and forth compilers from first-order logic with and without tuples will be convenient later on.

The syntax of first-order logic with n-tuples is given in Fig. 6. The expressions $o \in \mathcal{O}_\Sigma^n$ are like the expression $e \in \mathcal{E}_\Sigma$ except that variables x are now replaced by projection expressions

First-order expressions and formulas:

$$\begin{array}{ll} e \in \mathcal{E}_\Sigma \ ::= x \mid c \mid e \odot e' & \text{where } \odot \in F^{(2)},\ c \in C \\ \phi \in \mathcal{F}_\Sigma ::= e \doteq e \mid \exists x.\phi \mid \phi \wedge \phi \mid \neg\phi & \text{where } x \in \mathcal{V} \end{array}$$

Interpretation of expressions as sets of elements $[\![e]\!]^{S,\alpha} \subseteq dom(S)$, where S is a Σ-structures and $\alpha : V \to dom(S)$ where V contains all free variables.

$$[\![c]\!]^{S,\alpha} = c^S \qquad [\![x]\!]^{S,\alpha} = \{\alpha(x)\} \qquad [\![e \odot e']\!]^{S,\alpha} = \cup\{s \odot^S s' \mid s \in [\![e]\!]^{S,\alpha}, s' \in [\![e']\!]^{S,\alpha}\}$$

Interpretation of formulas as truth values $[\![\phi]\!]^{S,\alpha} \in \mathbb{B}$:

$$[\![e \doteq e']\!]^{S,\alpha} = \begin{cases} 1 \text{ if } [\![e]\!]^{S,\alpha} \cap [\![e']\!]^{S,\alpha} \neq \emptyset \\ 0 \text{ else} \end{cases} \qquad [\![\phi \wedge \phi']\!]^{S,\alpha} = [\![\phi]\!]^{S,\alpha} \wedge^{\mathbb{B}} [\![\phi']\!]^{S,\alpha}$$

$$[\![\neg\phi]\!]^{S,\alpha} = \neg^{\mathbb{B}}([\![\phi]\!]^{S,\alpha}) \qquad [\![\exists x.\phi]\!]^{S,\alpha} = \begin{cases} 1 \text{ if exists } s \in dom(S).\ [\![\phi]\!]^{S,\alpha[x/s]} = 1 \\ 0 \text{ else} \end{cases}$$

Fig. 5. Syntax and semantics of expressions and formulas of first-order logic.

$$\begin{array}{ll} o \in \mathcal{O}^n_\Sigma \ ::= \dot{\pi}_i(x) \mid c \mid o \odot o & \text{where } \odot \in F^{(2)},\ c \in C,\ 1 \leq i \leq n \\ \psi \in \mathcal{F}^{\mathrm{n}}_\Sigma ::= o \doteq o' \mid \exists x.\psi \mid \psi \wedge \psi \mid \neg\psi & \text{where } x \in \mathcal{V} \end{array}$$

Fig. 6. Σ-expressions and Σ-formulas of the first-order logic with n-tuples.

$\dot{\pi}_i(x)$ where $1 \leq i \leq n$. The reason is that any variable does now denote an n-tuple of values, rather than a single value (while the interpretation of constants and function symbols remain unchanged). The only change in the semantics is that variables assignment β do now map to n-tuples of values of the domain, and that $[\![\dot{\pi}_i(x)]\!]^{S,\beta} = \{\pi_i(\beta(x))\}$. The set of solutions of a formula $\psi \in \mathcal{F}^{\mathrm{n}}_\Sigma$ over a Σ-structure S is defined as follows:

$$n\text{-}sol^S(\psi) = \{\beta : \mathcal{V}(\psi) \to dom(S)^n \mid [\![\psi]\!]^{S,\beta} = 1\}$$

We next show how to express any first-order formulas in $\mathcal{F}_\Sigma$, interpreted over a tuple algebra S^n, by some formulas in $\mathcal{F}^{\mathrm{n}}_\Sigma$, interpreted over S. In a first step, we convert first-order expression in $e \in \mathcal{E}_\Sigma$ – that we will interpret over the Σ-algebra S^n – to n projected expressions $\Pi_i(e) \in \mathcal{O}^n_\Sigma$ where $1 \leq i \leq n$. For all operators $\odot \in F^{(2)}$ and constants $c \in C$ we define:

$$\Pi_i(e \odot e') =_{\text{def}} \Pi_i(e) \odot \Pi_i(e') \qquad \Pi_i(x) =_{\text{def}} \dot{\pi}_i(x) \qquad \Pi_i(c) =_{\text{def}} c$$

In the second step, we convert any formula $\phi \in \mathcal{F}_\Sigma$ without tuples – that will be interpreted in the tuple algebra S^n – to some formula $\langle\phi\rangle^n \in \mathcal{F}^{\mathrm{n}}_\Sigma$ with tuples.

$$\begin{array}{ll} \langle e \doteq e'\rangle^n =_{\text{def}} \wedge_{i=1}^n \Pi_i(e) \doteq \Pi_i(e') & \langle\phi \wedge \phi'\rangle^n =_{\text{def}} \langle\phi\rangle^n \wedge \langle\phi'\rangle^n \\ \langle\neg\phi\rangle^n =_{\text{def}} \neg\langle\phi\rangle^n & \langle\exists x.\phi\rangle^n =_{\text{def}} \exists x.\langle\phi\rangle^n \end{array}$$

Proposition 10. *For any $\phi \in \mathcal{F}_\Sigma$, Σ-structure S, and $n \geq 1$: $sol^{S^n}(\phi) = n\text{-}sol^S(\langle\phi\rangle^n)$.*

Example 11. Let $\mathbf{3} =_{\text{def}} 1 + 1 + 1$ and $\mathbf{4} =_{\text{def}} 1 + 1 + 1 + 1$. The formula $\phi \in \mathcal{F}_\Sigma$ equal to:

$$\mathbf{3} * x + \mathbf{4} * y \doteq 0$$

then has the same solutions over $\mathbb{R}^2_+$ than the formula $\langle\phi\rangle^2 \in \mathcal{F}^2_\Sigma$ over $\mathbb{R}_+$ below:

$$\mathbf{3} * \dot{\pi}_1(x) + \mathbf{4} * \dot{\pi}_1(y) \doteq 0 \wedge \mathbf{3} * \dot{\pi}_2(x) + \mathbf{4} * \dot{\pi}_2(y) \doteq 0$$

We next show how to rewrite any first-order formulas with tuples $\psi \in \mathcal{F}_\Sigma^n$ into some first-order formula $\tilde{\nu}(\psi) \in \mathcal{F}_\Sigma$ without tuples. The idea is to replace all projections $\pi_i(x)$ by new variables $\nu_i(x)$. For this, we first fix n generators of fresh variables $\nu_1, \ldots, \nu_n : \mathcal{V} \to \mathcal{V}$. Second, we map any expression $o \in \mathcal{O}_\Sigma^n$ with projections to some expressions $\tilde{\nu}(o) \in \mathcal{E}_\Sigma$ without new variables:

$$\tilde{\nu}(\dot{\pi}_i(x)) =_{\text{def}} \nu_i(x), \qquad \tilde{\nu}(c) =_{\text{def}} c, \qquad \tilde{\nu}(o_1 \odot o_2) =_{\text{def}} \tilde{\nu}(o_1) \odot \tilde{\nu}(o_2).$$

Third, we map any formula $\psi \in \mathcal{F}_\Sigma^n$ with projections to some formula $\tilde{\nu}(\psi) \in \mathcal{F}_\Sigma$ with fresh variables:

$$\begin{array}{ll} \tilde{\nu}(o = o') =_{\text{def}} \tilde{\nu}(o) = \tilde{\nu}(o') & \tilde{\nu}(\neg\psi) =_{\text{def}} \neg\tilde{\nu}(\psi) \\ \tilde{\nu}(\psi \wedge \psi') =_{\text{def}} \tilde{\nu}(\psi) \wedge \tilde{\nu}(\psi') & \tilde{\nu}(\exists x.\psi) =_{\text{def}} \exists \nu_1(x) \ldots \exists \nu_n(x).\ \tilde{\nu}(\psi) \end{array}$$

Given an variable assignment $\beta : V \to dom(S)^n$ with $V \subseteq \mathcal{V}$, we define $\nu(\beta) : \uplus_{i=1}^n \nu_i(V) \to dom(S)$ such that for all $x \in V$:

$$\nu(\beta)(\nu_i(x)) = \pi_i(\beta(x)))$$

Function ν is a bijection with range $\{\alpha \mid \alpha : \uplus_{i=1}^n \nu_i(V) \to dom(S)\}$. The inverse of this function satisfies $\nu^{-1}(\alpha)(x) = (\alpha(\nu_1(x)), \ldots, \alpha(\nu_n(x))$ for all α in the range and all $x \in V$.

Proposition 12. *For any $\psi \in \mathcal{F}_\Sigma^n$, Σ-structure S, and $n \geq 1$: $n\text{-}sol^S(\psi) = \nu^{-1}(sol^S(\tilde{\nu}(\psi)))$.*

We finish this section with a commutation property of the operator ν^{-1} with Σ-abstractions.

Proposition 13. *For any subset R of variable assignments of type $V \to dom(S)$ where $V \subseteq \mathcal{V}$, $n \geq 1$, and Σ-abstraction $h : S \to \Delta$: $\nu^{-1}(h \circ R) = h^n \circ \nu^{-1}(R)$.*

7 Difference Abstraction

We next show how to recast the notions of difference abstractions from [15,4,12] by applying our notion of Σ-abstractions to the Σ-algebra $\mathbb{R}_+^2$.

Let S be a Σ-algebra and $V \subseteq \mathcal{V}$ a subset of variables. For any two variable assignments $\alpha, \alpha' : V \to dom(S)$, we define an assignment of variables to pairs of elements in the domain of the structure $\mathit{diff}(\alpha, \alpha') : V \to dom(S)^2$ – that we call the differences of α and α' – such that for all variables $x \in V$, $\mathit{diff}(\alpha, \alpha')(x) = (\alpha(x), \alpha'(x))$. For any subset R of variable assignments of type $V \to dom(S)$ we define the *set of differences of assignments in R* by:

$$\mathit{diff}(R) = \{\mathit{diff}(\alpha, \alpha') \mid \alpha, \alpha' \in R\}$$

Furthermore, for any Σ-abstraction $h : S^2 \to \Delta$ and subset R' of difference abstractions of type $V \to dom(S)^2$ we define the application of the abstraction h to R' by $h \circ R' =_{\text{def}} \{h \circ \beta \mid \beta \in R'\}$

Definition 14. *For any Σ-abstraction $h : S^2 \to \Delta$ and formula $\phi \in \mathcal{F}_\Sigma$ we define the difference abstraction of the S-solution set of ϕ by: $sol^S(\phi)^\Delta = h \circ \mathit{diff}(sol^S(\phi)))$.*

The original definition of $sol(\phi)^{\Delta_6}$ in [15] did not make explicit the roles of *diff* and $h_{\Delta_6} : \mathbb{R}_+^2 \to \Delta_6$. Having done so, we can now see that the difference abstraction of the $\mathbb{R}_+$-solution sets of a formula is the $\mathbb{R}_+^2$-solution set of the same formula.

Lemma 15. *For any formula $\phi \in \mathcal{F}_\Sigma$ and Σ-structure S: $\mathit{diff}(sol^S(\phi)) = sol^{S^2}(\phi)$.*

As an immediate consequence, we have for any Σ-abstraction $h : S^2 \to \Delta$ that $sol(\phi)^\Delta = h_\Delta \circ sol^{S^2}(\phi)$. Our next objective is to show that we can overapproximate the set $sol(\phi)^\Delta$ by $sol^\Delta(\phi)$ (Corollary 19). In order to show this, let $h'{:}S' \to \Delta$ be a Σ-abstraction and α be a variable assignment into $dom(S')$:

Lemma 16. *For any expression $e \in \mathcal{E}_\Sigma$ with $V(e) \subseteq dom(\alpha)$: $h'([\![e]\!]^{S',\alpha}) \subseteq [\![e]\!]^{\Delta,h'\circ\alpha}$.*

Proposition 17. *For any positive formula $\phi \in \mathcal{F}_\Sigma$ with $V(\phi) \subseteq dom(\alpha)$: $[\![\phi]\!]^{S',\alpha} \leq [\![\phi]\!]^{\Delta,h'\circ\alpha}$.*

Theorem 18. *For any positive formula $\phi \in \mathcal{F}_\Sigma$: $h' \circ sol^{S'}(\phi) \subseteq sol^\Delta(\phi)$.*

Corollary 19. *For any Σ-abstraction $h : S^2 \to \Delta$ and positive first-order formula $\phi \in \mathcal{F}_\Sigma$:*

$$sol^S(\phi)^\Delta = h \circ \mathit{diff}(sol^S(\phi)) \subseteq sol^\Delta(\phi)$$

This is an obvious consequence from Theorem 18 and Proposition 15. If Δ is finite then the set $sol^\Delta(\phi)$ is finite. If furthermore ϕ is a conjunctive formula, we can therefore compute the set $sol^\Delta(\phi)$ by a finite domain constraint solver (such as e.g. Minizinc [18]). In contrast, it remains unclear how to compute the finite set $h \circ \mathit{diff}(sol^S(\phi))$ for infinite structures S. The problem is open, even if ϕ is a system of homogenous linear equations and $S = \mathbb{R}_+$, so that the infinite set $sol^S(\phi)$ has a finite solved form by a triangular matrix. This is the core of the objective that we tackle in the remainder of the present paper.

8 Objective

We formalize the full algorithmic problem that we will solve in this paper and illustrate its relevance to our benchmark application to systems biology.

Once having fixed the parameter $\Delta \in \{\Delta_3, \Delta_6\}$ the algorithmic problem has three inputs :

Linear system over $\mathbb{R}_+$: a first-order formula $\phi \in \mathcal{F}_\Sigma$ that represents a linear equation system. (This formula typically captures the steady state equations of the model.)

Constraint over Δ: a first-order formula $\phi' \in \mathcal{F}_{\Sigma\cup\Delta}$ where the signature Σ is extended with additional constants of Δ that will be interpreted by themselves. (This formula typically expresses the partial kinetic knowledge on the reactions in the model and the change target of the prediction task (e.g. overproduction of some metabolites).)

Set of observable variables: a finite subset of variables $V \subseteq \mathcal{V}(\phi) \cup \mathcal{V}(\phi')$. (This set typically contains the control variables such as inflows and gene knockouts as well as the target variables, but not the variables for the rate of the internal metabolic reactions. Since the number of solutions may be of cardinality $|\Delta|^{|V|}$, it is essential to choose V as small as possible.)

The algorithmic output that has to be produced is the Δ-abstraction of differences of $\mathbb{R}_+$-solutions of ϕ, but constrained to the solutions of ϕ' over the structure Δ, and projected to the variables of V. In other words, the algorithm will compute the following finite set where $V' = \mathcal{V}(\phi) \cup \mathcal{V}(\phi')$:

$$\{\beta_{|V} \mid \beta \in sol^{\mathbb{R}_+}_{V'}(\phi)^\Delta \cap sol^\Delta_{V'}(\phi')\}$$

The only restriction on the inputs that we will impose is that the first formula ϕ must represent a homogeneous system of linear equations in $\mathcal{F}_\Sigma$. For instance, the linear equation $x - 2y = 0$ is captured by the equation $x \doteq y + y$ in $\mathcal{F}_\Sigma$ where we cannot use the minus operator. See Section 9 for the general definition. The constraint $\phi' \in \mathcal{F}_{\Sigma\cup\Delta}$ in contrast may be arbitrary, including nonlinear equations and universal quantifiers but must be interpreted abstractly over Δ, while

Observable variables	Linear equations in $\mathcal{F}_\Sigma$	Δ_6 constraints in $\mathcal{F}_{\Sigma\cup\Delta_6}$
$V = \{\ x_{Thr},$ $x_{Akb},$ $y_{Leu},$ $\ldots\}$	$\phi =$ $x_{Thr} \doteq r_{27} + r_{30}$ (Thr) $\wedge\ r_{41} \doteq r_{27}$ (Akb) $\wedge\ r_{45} \doteq r_{35} + y_{Leu}$ (Leu) $\wedge \ldots$	$\phi' =$ $y_{Leu} \doteq \uparrow$ $(target)$ $\wedge\ r_{27} \doteq Pyr * Akb * IlvD$ $* IlvBH * IlvC$ (27) $\wedge \ldots$

Fig. 7. Inputs of our algorithm on the benchmark example from of leucine overproduction.

the linear equation system is valid over $\mathbb{R}_+$. Note however, that any universal quantifiers in ϕ' can be expressed by a simple conjunction, given that the interpretation domain Δ is finite.

In Fig. 7 we illustrate how the inputs will be instantiated for our benchmark application of leucine overproduction (a glimpse of the reaction network was given in Fig. 2). In this case, we choose the parameter $\Delta = \Delta_6$. The observable variables in V stand for the rates of the inflows Threonine (x_{Thr}), Akb(x_{Akb}), etc, the rate of the target outflow Leucine (y_{Leu}), and the possible gene knockouts. The system of linear equations ϕ contains the steady state equations for the metabolic reactions in the network. These require that all metabolites must be produced and consumed at the same rate. For instance, Pyruvate is produced by the inflow of Threonine at rate x_{Thr} and consumed by reactions 27 and 30 at rates r_{27} and r_{30} respectively. The yields the linear equation (Thr) of Fig. 7. Species Akb is produced by reaction 41 and consumed by 27, leading to the steady state equation (Akb). Leucine is produced by reaction 45 and consumed by its outflow, leading to equation (Leu). The constraint $\phi \in \mathcal{F}_{\Sigma\cup\Delta_6}$ contains the overproduction target $y_{Leu} = \uparrow$ in ($target$) and the kinetic constraints for all reactions, of which we show only constraint (27) for reaction 27. The kinetic constraints must be interpreted abstractly over Δ_6 according the formal semantics of the modeling language [15] rather than concretely over $\mathbb{R}_+$. Therefore, the meaning of the constraints is purely qualitative and not at all quantitative. For instance, the constraint (27) states (beside others) that rate of reaction 27 increases if either of the concentrations of the reactants Pyr and Akb or of the enzymes $IlvD$, $IlvBH$ or $IlvC$ increase. Nothing is said about quantities of these increases.

9 Exact Algorithms

We now present an exact solution of the problem presented in the previous section. Our approach is to characterize the abstraction of the solution set of a linear equation system as the solution set of some first-order formula over the abstract domain. We consider the abstractions $h_{\mathbb{B}}$, h_{Δ_3}, and h_{Δ_6} in this order.

Characterizing $\mathbb{B}$-Abstractions. We now present a result from [1] that shows that the boolean abstraction of the $\mathbb{R}_+$-solution set of a mixed linear system can be computed exactly. The development of this result was motivated by the needs of the present paper, but given that it is of independent interest and nontrivial to prove, we decided to present it independently.

Any natural numbers n can be described by the expression $\mathbf{n} =_{\text{def}} \sum_{i=1}^{n} 1$ in $\mathcal{E}_\Sigma$. This permits to define *linear equations* as equations in $\mathcal{F}_\Sigma$ that have the form:

$$\mathbf{n}_1 * x_1 + \ldots \mathbf{n}_m * x_m \doteq \mathbf{n}_{m+1} * x_{m+1} + \ldots + \mathbf{n}_p * x_p \tag{3}$$

where $m, p, n_1, \ldots, n_p \in \mathbb{N}$ and $x_1, \ldots, x_p \in \mathcal{V}$.

Mixed Linear Systems. A *product-zero-equation* in $\mathcal{F}_\Sigma$ is an equation of the form $x * y \doteq 0$ where $x, y \in \mathcal{V}$. A *mixed linear system* is a conjunctive formula in $\mathcal{F}_\Sigma$ of the form $\exists \mathbf{z}.\ L \wedge E$ where L is a conjunction of linear equations and E a conjunction of product-zero-equations.

Elementary Flux Modes. The support of a variable assignment $\alpha : V \to \mathbb{R}$ with $V \subseteq \mathcal{V}$ is $\text{supp}(\alpha) = \{x \in dom(\alpha) \mid \alpha(x) \neq 0\}$. Given a linear system ϕ, the EFMs of ϕ are the minimal

support solutions of ϕ over $\mathbb{R}_+$. The $\mathbb{R}$-EFMs of ϕ are the minimal support solutions of ϕ over $\mathbb{R}$. Note that the interpretation of $\mathbb{R}$ is natural for the steady-state equations of metabolic networks with reversible reactions, while the reactions of our networks are always irreversible.

Theorem 20 [1]. *Let ϕ be a mixed linear system. We can compute in at most exponential time some conjunctive formula ϕ' with existential quantifiers such that $h_{\mathbb{B}} \circ sol^{\mathbb{R}_+}(\phi) = sol^{\mathbb{B}}(\phi')$.*

Proof sketch. There are quite some insights behind this theorem that we can only sketch here. First, any linear equation L system can be rewritten in the form $A\mathbf{y} \doteq 0$ where A is an integer matrix and $\mathbf{y}$ a sequence of pairwise distinct variables such that $V(\mathbf{y}) = V(L)$. Let P be a positive integer matrix whose columns contain all the EFMs of A. These can be computed in at most exponential time [19]. Then $sol^{\mathbb{R}_+}(A\mathbf{y} \doteq 0)$ is equal to $sol^{\mathbb{R}_+}(\exists \mathbf{x}.\ P\mathbf{x} \doteq \mathbf{y})$ given that any solution of $A\mathbf{y} \doteq 0$ over $\mathbb{R}_+$ can be obtained from some linear combination of the EFMs of A. Second, a formula $\phi \in \mathcal{F}_\Sigma$ is called $h_{\mathbb{B}}$-exact if $sol^{\mathbb{B}}(\phi) = h_{\mathbb{B}} \circ sol^{\mathbb{R}_+}(\phi)$. Unfortunately, not every linear systems is $h_{\mathbb{B}}$-exact. However, the formula ϕ'' equal to $\exists \mathbf{x}.P\mathbf{x} \doteq \mathbf{y}$ can be shown to be $h_{\mathbb{B}}$-exact, roughly since matrix P contains only positive integers. Third, it was noticed that any conjunction of product-zero equations is $h_{\mathbb{B}}$-exact as well. Fourth, for any system of product-zero equations E and any sequence of variables $\mathbf{z}$, the formula ϕ' equal to $\exists \mathbf{z}.\phi'' \wedge E$ can be shown to be $h_{\mathbb{B}}$-exact. Finally, any mixed linear systems ϕ can be brought into the form of ϕ' by computing the EFMs of the matrix A of the linear subsystem of ϕ in exponential time. □

Exact Algorithm.. In order to compute the $h_{\mathbb{B}}$-abstraction of a mixed linear system ϕ, we first compute ϕ' along the lines of the sketch of the proof ideas of Theorem 20. Second, given that ϕ' is a conjunctive formula, we compute $sol^{\mathbb{B}}(\phi')$ by finite domain constraint programming.

Characterizing Δ_3-Abstractions. We present a characterization of Δ_3-abstractions of linear equation systems and show that it provides an exact algorithm solving the objective in the case of Δ_3. We first decompose h_{Δ_3} into $h_{\mathbb{B}}$ and the binary relation, that is defined by the following first-order formula in the logic with pairs $\mathcal{F}^2_\Sigma$:

$$proj_G(x, y) =_{\text{def}} \dot{\pi}_1(x) + \dot{\pi}_2(y) = \dot{\pi}_2(x) + \dot{\pi}_1(y) \wedge \dot{\pi}_1(y) * \dot{\pi}_2(y) = 0$$

We are mainly interested in interpreting this formula over $\mathbb{R}^2_+$.

Lemma 21. *The relation ${proj_G}^{\mathbb{R}^2_+}$ is a function satisfying $h_{\Delta_3} = h^2_{\mathbb{B}} \circ {proj_G}^{\mathbb{R}^2_+}$.*

We next define applications of $proj_G$ in FO-logic. For any sequence of variables $\mathbf{y}$ and FO-formula $\phi(\mathbf{y}) \in \mathcal{F}^2_\Sigma$ with $\mathcal{V}(\phi(\mathbf{y})) \subseteq \{\mathbf{y}\}$ we define a formula $proj_G(\phi(\mathbf{y})) \in \mathcal{F}^2_\Sigma$ describing the application of $proj_G$ to the solutions of $\phi(\mathbf{y})$ by $\exists \mathbf{z}.\ \phi(\mathbf{z}) \wedge proj_G(\mathbf{z}, \mathbf{y})$ where $\phi(\mathbf{z})$ is obtained from $\phi(\mathbf{y})$ by replacing the variables in $\mathbf{y}$ by arbitrarily but fixed fresh variables $\mathbf{z}$.

Lemma 22. ${proj_G}^{\mathbb{R}^2_+} \circ 2\text{-}sol^{\mathbb{R}_+}(\phi(\boldsymbol{y})) = 2\text{-}sol^{\mathbb{R}_+}(proj_G(\phi(\boldsymbol{y})))$

Theorem 23. *For any linear formula $L \in \mathcal{F}_\Sigma$ we can compute in at most exponential time a positive conjunctive formula with existential quantifiers $\phi \in \mathcal{F}_\Sigma$ such that:*

$$h_{\Delta_3} \circ \mathit{diff}(sol^{\mathbb{R}_+}(L)) = \nu^{-1}(sol^{\mathbb{B}}(\phi))$$

Proof Let $L(\mathbf{y})$ be a linear system with $\mathcal{V}(L(\mathbf{y})) = \mathcal{V}(\mathbf{y})$ where $\mathbf{y}$ is a sequence of distinct variables. The time for computing ϕ is dominated by the time for computing the elementary

modes, which can be done in at most exponential time.

$$
\begin{array}{rl}
 & h_{\Delta_3} \circ \mathit{diff}(\mathit{sol}^{\mathbb{R}_+}(L(\mathbf{y}))) \\
\textit{Proposition 15} & = h_{\Delta_3} \circ \mathit{sol}^{\mathbb{R}_+^2}(L(\mathbf{y})) \\
\textit{Pair FO Proposition 10} & = h_{\Delta_3} \circ 2\text{-}\mathit{sol}^{\mathbb{R}_+}(L_2(\mathbf{y})) \ \textit{with}\ L_2(\mathbf{y}) = \langle L(\mathbf{y})\rangle^2 \\
\textit{Decomposition Lemma 21} & = h^2_{\mathbb{B}} \circ \mathit{proj}_G{}^{\mathbb{R}_+^2} \circ 2\text{-}\mathit{sol}^{\mathbb{R}_+}(L_2(\mathbf{y}))) \\
\textit{FO-Definition Lemma 22} & = h^2_{\mathbb{B}} \circ 2\text{-}\mathit{sol}^{\mathbb{R}_+}(\mathit{proj}_G(L_2(\mathbf{y}))) \\
\textit{Proposition 12} & = h^2_{\mathbb{B}} \circ \nu^{-1}(\mathit{sol}^{\mathbb{R}_+}(\tilde{\nu}(\mathit{proj}_G(L_2(\mathbf{y})))))) \\
\textit{Proposition 13} & = \nu^{-1}(h_{\mathbb{B}} \circ \mathit{sol}^{\mathbb{R}_+}(\tilde{\nu}(\mathit{proj}_G(L_2(\mathbf{y})))))) \\
\textit{Definition of } \mathit{proj}_G(L_2(\boldsymbol{y})) & = \nu^{-1}(h_{\mathbb{B}} \circ \mathit{sol}^{\mathbb{R}_+}(\tilde{\nu}(\exists \mathbf{z}.\ L_2(\mathbf{z}) \wedge \mathit{proj}_G(\mathbf{z},\mathbf{y})))) \\
\textit{Mixed linear systems Theorem 20} & = \nu^{-1}(\mathit{sol}^{\mathbb{B}}(\phi)) \\
 & \quad \textit{where } \phi \text{ is an equivalent conjunctive formula for the} \\
 & \quad \textit{mixed linear sytem } \tilde{\nu}(\exists \mathbf{z}.\ L_2(\mathbf{z}) \wedge \mathit{proj}_G(\mathbf{z},\mathbf{y})) \qquad \square
\end{array}
$$

Note that $\mathit{sol}^{\mathbb{B}}(\phi)$ can be computed by finite domain constraint programming. This yields an exact algorithm for computing the Δ_3-abstraction of a system of linear equations, which is a special case of our general objective without kinetic constraints.

For adding a treatment of kinetic constraints, we consider the union $\mathbb{B} \cup \Delta_3$ as a relational structure providing the values and functions of both structures $\mathbb{B}$ and Δ_3. The signature of this mixed structure consists of the function symbols in $\{+^{\mathbb{B}}, *^{\mathbb{B}}, +^{\Delta_3}, *^{\Delta_3}\}$ and the constants in the set $\mathbb{B} \cup \Delta_3$, all of which are interpreted by themselves in the mixed structure $\mathbb{B} \cup \Delta_3$. The set of first-order formulas over the mixed signature is denoted by $\mathcal{F}_{\mathbb{B}\cup\Delta_3}$. For any $\alpha : V \to \mathit{dom}(S)$, we can define its restriction $\alpha_{|V'} : V' \to \mathit{dom}(S)$ such that for all $y \in V' \subseteq V$, $\alpha_{|V'}(y) = \alpha(y)$.

Proposition 24. *For any formulas $\phi \in \mathcal{F}_\Sigma$ and $\phi' \in \mathcal{F}_{\Sigma\cup\Delta_3}$ and sets of variables $V \subseteq V' = \mathcal{V}(\phi) \cup \mathcal{V}(\phi')$ we can compute in linear time a formula $\phi_M \in \mathcal{F}_{\mathbb{B}\cup\Delta_3}$ over the mixed signature such that: $\mathit{sol}^{\mathbb{B}\cup\Delta_3}(\phi_M) = \{\beta_{|V} \mid \beta \in \nu^{-1}(\mathit{sol}^{\mathbb{B}}_{V'}(\phi)) \cap \mathit{sol}^{\Delta_3}_{V'}(\phi')\}$.*

The set $\mathit{sol}^{\mathbb{B}\cup\Delta_3}(\phi_M)$ can be computed by a finite domain constraint programming, since $\mathbb{B}\cup\Delta_3$ is a finite structure. By combining Theorem 23 and Proposition 29 we obtain an algorithm for solving the general problem of Section 8 in the cases of Δ_3.

Characterizing Δ_6-Abstractions. The case of Δ_6 is considerably more envolved that the case of Δ_3, even though follwoing the same general approach. For this, we consider the abstraction h_{Δ_6} as an element of the algebra of total functions on $\mathbb{R}^2_+$, that we denote as $\mathbb{R}^2_+ \to \mathbb{R}^2_+$. The following lemma shows that h_{Δ_6} is the sum of h_{Δ_3} and $h^2_{\mathbb{B}}$ in this Σ-algebra.

Lemma 25. *$h_{\Delta_6} = h^2_{\mathbb{B}} + h_{\Delta_3}$ where $+ = +^{\mathbb{R}^2_+ \to \mathbb{R}^2_+}$*

Let $\mathit{idproj}_G{}^{\mathbb{R}^2_+} : \mathbb{R}^2_+ \to (\mathbb{R}^2_+)^2$ such that for any $p \in \mathbb{R}^2_+$ $\mathit{idproj}_G{}^{\mathbb{R}^2_+}(p) = (p, \mathit{proj}_G{}^{\mathbb{R}^2_+}(p))$. Furthermore, we define for any two functions $g : A \to B \times C$ and $f : B \times C \to D$ the pseudo composition $f \bullet g : A \to D$ such that for all $a \in A$: $(f \bullet g)(a) = f(\pi_1(g(a)), \pi_2(g(a)))$. The Σ-abstraction $h^2_{\mathbb{B}} : \mathbb{R}^2_+ \to \mathbb{B}^2$ allows us to define $(h^2_{\mathbb{B}})^2 : (\mathbb{R}^2_+)^2 \to (\mathbb{B}^2)^2$

Lemma 26 Decomposition. *$h_{\Delta_6} = +^{\mathbb{R}^2_+} \bullet (h^2_{\mathbb{B}})^2 \circ \mathit{idproj}_G{}^{\mathbb{R}^2_+}$.*

We can now define the ternary relation $\mathit{idproj}_G{}^{\mathbb{R}^2_+}$ in the first-order logic with pairs by $\mathit{idproj}_G : \mathcal{V} \times \mathcal{V}^2 \to \mathcal{F}^2_\Sigma$ such that for all $x, y_1, y_2 \in \mathcal{V}$:

$$\mathit{idproj}_G(x, y_1, y_2) =_{\text{def}} \langle x = y_1\rangle^2 \wedge \mathit{proj}_G(x, y_2)$$

We next define applications of $idproj_G$ in FO-logic. For any sequence of variables $\mathbf{y}$ and FO-formula $\phi(\mathbf{y}) \in \mathcal{F}^2_\Sigma$ with $\mathcal{V}(\phi(\mathbf{y})) \subseteq \{\mathbf{y}\}$ we define a formula $idproj_G(\phi(\overline{\mathbf{y}})) \in \mathcal{F}^2_\Sigma$ for describing the application of $idproj_G$ to the solution set of $\phi(\mathbf{y})$. We let $idproj_G(\phi(\overline{\mathbf{y}})$ be $\exists \mathbf{z}.\ \phi(\mathbf{z}) \wedge idproj_G(\mathbf{z}, \mathbf{y}^1, \mathbf{y}^2)$ where $\phi(\mathbf{z})$ is obtained from $\phi(\mathbf{y})$ by replacing the variables in $\mathbf{y}$ by arbitrarily but fixed fresh variables $\mathbf{z}$ and by fixing two sequences of fresh variables $\mathbf{y}^1, \mathbf{y}^2 \in \mathcal{V}^m$ such that $\overline{\mathbf{y}} = (\mathbf{y}^1, \mathbf{y}^2)$.

Lemma 27. $idproj_G{}^{\mathbb{R}^2_+} \circ 2\text{-}sol^{\mathbb{R}_+}(\phi(\boldsymbol{y})) = \{[\boldsymbol{y}/(\alpha(\boldsymbol{y}^1), \alpha(\boldsymbol{y}^2)] \mid \alpha \in 2\text{-}sol^{\mathbb{R}_+}(idproj_G(\phi(\overline{\boldsymbol{y}})))\}$.

Theorem 28. *For any linear formula $L(\boldsymbol{y})$ with free distinct variable $\boldsymbol{y}$ we can compute in at most exponential time a positive conjunctive formula with existential quantifiers $\phi' \in \mathcal{F}_\Sigma$ and sequences of variables $\boldsymbol{y}^1, \boldsymbol{y}^2$ such that:*

$$h_{\Delta_6} \circ diff(sol^{\mathbb{R}_+}(L(\boldsymbol{y}))) = \{[\boldsymbol{y}/(\beta(\boldsymbol{y}^1) +^{\mathbb{R}^2_+} \beta(\boldsymbol{y}^2)] \mid \beta \in \nu^{-1}(sol^{\mathbb{B}}(\phi'))\}$$

Proof Let $L(\mathbf{y})$ be a linear formula L with free distinct variable $\mathbf{y} \in \mathcal{V}^m$.

$$\begin{array}{rl}
 & h_{\Delta_6} \circ diff(sol^{\mathbb{R}_+}(L(\mathbf{y}))) \\
\textit{Proposition 15} & = h_{\Delta_6} \circ sol^{\mathbb{R}^2_+}(L(\mathbf{y})) \\
\textit{Proposition 10} & = h_{\Delta_6} \circ 2\text{-}sol^{\mathbb{R}_+}(L_2(\mathbf{y})) \ \textit{with}\ L_2(\mathbf{y}) = \langle L(\mathbf{y})\rangle^2 \\
\textit{Decomposition Lemma 26} & = +^{\mathbb{R}^2_+} \bullet (h^2_{\mathbb{B}})^2 \circ idproj_G{}^{\mathbb{R}^2_+} \circ 2\text{-}sol^{\mathbb{R}_+}(L_2(\mathbf{y})) \\
\textit{FO-Definition Lemma 27} & = +^{\mathbb{R}^2_+} \bullet (h^2_{\mathbb{B}})^2 \circ \{[\mathbf{y}/(\alpha(\mathbf{y}^1), \alpha(\mathbf{y}^2)] \mid \alpha \in 2\text{-}sol^{\mathbb{R}_+}(idproj_G(L_2(\overline{\mathbf{y}})))\} \\
 & = +^{h_{\mathbb{R}^2_+}} \bullet \{[\mathbf{y}/(\beta(\mathbf{y}^1), \beta(\mathbf{y}^2)] \mid \beta \in h^2_{\mathbb{B}} \circ 2\text{-}sol^{\mathbb{R}_+}(idproj_G(L_2(\overline{\mathbf{y}})))\} \\
 & = \{[\mathbf{y}/(\beta(\mathbf{y}^1) +^{\mathbb{R}^2_+} \beta(\mathbf{y}^2)] \mid \beta \in h^2_{\mathbb{B}} \circ 2\text{-}sol^{\mathbb{R}_+}(idproj_G(L_2(\overline{\mathbf{y}})))\}
\end{array}$$

We can now finish the proof by computing the $h^2_{\mathbb{B}}$ abstraction of the above solution set similarly to the case of Δ_3.

$$\begin{array}{rl}
 & h^2_{\mathbb{B}} \circ 2\text{-}sol^{\mathbb{R}_+}(idproj_G(L_2)(\overline{\mathbf{y}})) \\
\textit{Proposition 12} & = h^2_{\mathbb{B}} \circ \nu^{-1}(sol^{\mathbb{R}_+}(\tilde{\nu}(idproj_G(L_2(\overline{\mathbf{y}})))) \\
\textit{Proposition 13} & = \nu^{-1}(h_{\mathbb{B}} \circ sol^{\mathbb{R}_+}(\tilde{\nu}(idproj_G(L_2(\overline{\mathbf{y}}))))) \\
\textit{Definition of } idproj_G(L_2(\overline{\boldsymbol{y}})) & = \nu^{-1}(h_{\mathbb{B}} \circ sol^{\mathbb{R}_+}(\tilde{\nu}(\exists \mathbf{z}.\ L_2(\mathbf{z}) \wedge idproj_G(\mathbf{z}, \overline{\mathbf{y}})))) \\
\textit{Mixed linear systems Theorem 20} & = \nu^{-1}(sol^{\mathbb{B}}(\phi')) \\
 & \quad \textit{where } \phi' \textit{ is conjunctive formula equivalent to the} \\
 & \quad \textit{mixed linear system } \tilde{\nu}(\exists \mathbf{z}.\ L_2(\mathbf{z}) \wedge idproj_G(\mathbf{z}, \overline{\mathbf{y}})) \qquad \square
\end{array}$$

For adding a treatment of kinetic constraints, we consider the union $\mathbb{B} \cup \Delta_6$ as a relational structure providing the values and functions of both structures $\mathbb{B}$ and Δ_6 in analogy to the case of Δ_3. We denote the of first-order formulas over the signature of this mixed structure by $\mathcal{F}_{\mathbb{B}\cup\Delta_6}$.

Proposition 29. *For any formula $\phi(\boldsymbol{y}) \in \mathcal{F}_\Sigma$ with distinct free variables $\boldsymbol{y}$, formula $\phi' \in \mathcal{F}_{\Sigma\cup\Delta_6}$, and variable sets $V \subseteq V' = \mathcal{V}(\boldsymbol{y}) \cup \mathcal{V}(\phi')$ we can compute in linear time a formula $\phi_M \in \mathcal{F}_{\mathbb{B}\cup\Delta_6}$ such that $sol^{\mathbb{B}\cup\Delta_6}(\phi_M) = \{\beta_{|V} \mid \beta \in \{[\boldsymbol{y}/(\beta'(\boldsymbol{y}^1) +^{\mathbb{R}^2_+} \beta'(\boldsymbol{y}^2)) \mid \beta' \in \nu^{-1}(sol^{\mathbb{B}}_{V'}(\phi))\} \cap sol^{\Delta_6}_{V'}(\phi')$.*

The set $sol^{\mathbb{B}\cup\Delta_6}(\phi_M)$ can be computed by a finite domain constraint programming, since $\mathbb{B}\cup\Delta_6$ is a finite structure. By combining Theorem 28 and Proposition 29 we obtain an algorithm for solving the general problem of Section 8 in the case of Δ_6.

Network	Count type	pure abstr. interpr.	min. support consequences	exact
Simple metabolic cycle (Fig. 1)	abstract solutions	19	6	6
Leucine overproduction [4]	knockouts	16	14	14
	abstract solutions	292	228	228
Counterexample	abstract solutions	$\geq$ 10000	4454	4374

Fig. 8. Predictions for the networks analysed in this paper, obtained respectively by pure abstract interpretation, the heuristic based on minimal support consequences and the exact algorithm.

10 Heuristic Algorithm Based on Minimal Support Consequences

The intuition behind the heuristic with minimal support consequences relies on two facts: first, adding consequences to a given linear system L before applying abstract interpretation can improve the precision of the abstraction, as already discussed (2); second, the smaller the number of variables in an equation, the more constraining generally is its abstract interpretation. The heuristics is therefore very simple: before abstracting from $\mathbb{R}_+$ to Δ, the linear system L containing the steady state equations of the system is replaced by a linear system $L_{\min}$ containing all the minimal support $\mathbb{R}_+$-consequences of the equations in L. The linear system $L_{\min}$ can be computed by applying any existing algorithm for the calculation of $\mathbb{R}$-EFMs to the orthogonal complement of $L^{\perp}$ as follows:

1. From L compute a linear system $L^{\perp}$ whose solution space – seen as a subspace of the vector space $\mathbb{R}^{\mathcal{V}(L)}$ – is the orthogonal complement of $sol^{\mathbb{R}_+}(L)$. This can be done for example by using a variant of Gauß' triangularization method.
2. From $L^{\perp}$ compute the $\mathbb{R}$-EFMs $l_1^{\min}, \ldots, l_k^{\min}$ with any known $\mathbb{R}$-EFMs algorithm.
3. Build $L_{\min}$ by using $l_1^{\min}, \ldots, l_k^{\min}$ as the coefficients of the equations of $L_{\min}$.

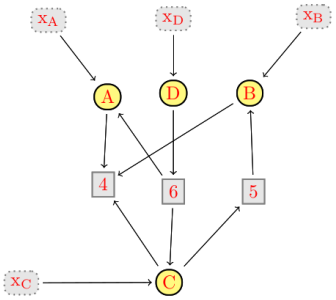
11 Experimental Results

We experimentally compare three algorithms for overapproximating the objective $\{\beta_{|V} \mid \beta \in sol_{V'}^{\mathbb{R}_+}(\phi)^{\Delta_6} \cap sol_{V'}^{\Delta}(\phi')\}$ given a linear system of equations ϕ, kinetic constraints ϕ', and observable variables V. The first algorithm directly applies pure abstract interpretation to ϕ to compute $sol^{\Delta_6}(\exists \overline{V}.(\phi \wedge \phi'))$ by finite domain programming where $\overline{V} = \mathcal{V}(\phi) \setminus V$, overapproximating the objective by Theorem 18. The second algorithm enriches the linear system ϕ with its minimal support consequences as discussed in Section 10 before applying abstract interpretation. The third algorithm is the exact algorithm that can be derived from Theorem 28.

The experimental results are summarized in Fig. 8. The first instance verifies our expectations on the toy metabolic network with a simple cycle in Fig. 1, without kinetic constraints and $V = \{X, Y\}$ as observable variables. The exact algorithm shows that there are 6 abstract solutions, one for each value of Δ_6. The heuristic with minimal support consequences finds exactly these same 6 abstract solutions, while by applying pure abstract interpretation we find 19 abstract solutions (out of 36 possible assignments), thus a large overapproximation.

The second real scale instance treats leucine overproduction on the network from Fig. 2, see Fig. 7 for a discussion of the precise inputs. The heuristic and the exact algorithm produce the same result with 226 abstract solutions, while by pure abstract interpretation 292 abstract solutions are found. Thereby, the two new algorithms both remove the same 2 wrong gene knockout predictions with respect to baseline algorithm by pure abstract interpretation.

However, the heuristics with minimal support consequences is not always exact: we found a counter example given on the right for which it slightly over-approximates the exact solution set.
On the other hand, the heurstic algorithm with *EFM-consequences* is remarkably faster than the exact algorithm – in the benchmark on leucine overproduction, we have 5 minutes versus 5 hours – while still being equally precise in most cases.



References

1. E. Allart, J. Niehren, and C. Versari. Computing sign abstractions of linear systems. In preparation.
2. L. Calzone, F. Fages, and S. Soliman. BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, July 2006.
3. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *POPL*, pages 269–282, 1979.
4. F. Coutte, J. Niehren, D. Dhali, M. John, C. Versari, and P. Jacques. Modeling Leucine's Metabolic Pathway and Knockout Prediction Improving the Production of Surfactin, a Biosurfactant from Bacillus Subtilis. *Biotechnology Journal*, 10(8):1216–34, Aug. 2015.
5. V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Abstracting the differential semantics of rule-based models: Exact and automated model reduction. In *LICS*, pages 362–381. IEEE Computer Society, 2010.
6. G. Facchetti and C. Altafini. Partial inhibition and bilevel optimization in flux balance analysis. *BMC Bioinformatics*, pages 344–344, 2013.
7. F. Fages, S. Gay, and S. Soliman. Inferring reaction systems from ordinary differential equations. *Theor. Comput. Sci.*, 599:64–78, 2015.
8. F. Fages and S. Soliman. Abstract interpretation and types for systems biology. *Theor. Comput. Sci.*, 403(1):52–70, 2008.
9. M. Feinberg. Chemical reaction network structure and the stability of complex isothermal reactors. *Chemical Engineering Science*, 42(10):2229 – 2268, 1987.
10. K. D. Forbus. Qualitative reasoning. In A. B. Tucker, editor, *The Computer Science and Engineering Handbook*, pages 715–733. CRC Press, 1997.
11. S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. Copasi—a complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, 2006.
12. M. John, M. Nebut, and J. Niehren. Knockout Prediction for Reaction Networks with Partial Kinetic Information. In *14th International Conference on Verification, Model Checking, and Abstract Interpretation*, Rom, Italy, Jan. 2013.
13. K. Lotz, A. Hartmann, E. Grafahrend-Belau, and B. Schreiber, F.and Junker. Elementary flux modes, flux balance analysis, and their application to plant metabolism. *Plant Metabolism. Methods in Molecular Biology (Methods and Protocols)*, 2014.
14. C. D. Maranas and A. R. Zomorrodi. *Flux Balance Analysis and LP Problems*, chapter 3, pages 53–80. Wiley-Blackwell, 2016.
15. J. Niehren, C. Versari, M. John, F. Coutte, and P. Jacques. Predicting Changes of Reaction Networks with Partial Kinetic Information. *BioSystems*, 149:113–124, July 2016.
16. J. D. Orth, I. Thiele, and B. O. Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–248, 2010.
17. J. A. Papin, J. Stelling, N. D. Price, S. Klamt, S. Schuster, and B. O. Palsson. Comparison of network-based pathway analysis methods. *Trends in biotechnology*, 22(8):400–405, 2004.
18. A. Rendl, T. Guns, P. J. Stuckey, and G. Tack. Minisearch: A solver-independent meta-search language for minizinc. In G. Pesant, editor, *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015*, volume 9255 of *LNCS*, pages 376–392, 2015.
19. D. Zanghellini, D. E. Ruckerbauer, M. Hanscho, and C. Jungreuthmayer. Elementary flux modes in a nutshell: Properties, calculation and applications. *Biotechnology Journal*, pages 1009–1016, 2013.