Computing Difference Abstractions of Linear Equation Systems

Emilie Allart, Joachim Niehren, Cristian Versari

▶ **To cite this version:**

Emilie Allart, Joachim Niehren, Cristian Versari. Computing Difference Abstractions of Linear Equation Systems. 2021. hal-03156136v1

HAL Id: hal-03156136
https://hal.science/hal-03156136v1

Preprint submitted on 2 Mar 2021 (v1), last revised 17 Jun 2021 (v3)

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.

Computing Difference Abstractions of Linear Equation Systems

Emilie Allart[a,c], Joachim Niehren[a,b], Cristian Versari[a,c]

[a] *BioComputing CRIStAL - Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL) - UMR 9189*
[b] *Linking Dynamic Data CRIStAL - UMR 9189, Inria Lille - Nord Europe*
[c] *Université de Lille*

Abstract

Abstract interpretation was proposed for predicting changes of reaction networks with partial kinetic information in systems biology. This requires to compute the set of difference abstractions of a system of linear equations under nonlinear constraints. We present the first practical algorithm that can compute the difference abstractions of linear equation systems exactly. We also present a new heuristics based on minimal support consequences for overapproximating the set of difference abstractions. Our algorithms rely on elementary modes, first-order definitions, and finite domain constraint programming. We implemented our algorithms and applied them to change prediction in systems biology. It turns out experimentally that the new heuristics is often exact in practice, while outperforming the exact algorithm.

Keywords: abstract interpretation, constraint programming, first-order definitions, gene knockout prediction, reaction networks, systems biology, synthetic biology, metabolic networks, boolean abstraction.

1. Introduction

Motivated by analysis questions for steady states [24, 25] of chemical reaction networks [12, 5, 16, 10] we study the problem how to compute difference abstractions for the solution sets of linear equations systems. These may be subject to abstract difference constraints for expressing partial information on the kinetics of reactions [23].
Problem. We consider systems of homogeneous linear equations with variables for positive real numbers in $\mathbb{R}_+$ including 0, possibly existentially quantified, such as for instance:

$$E(X,Y): \quad \exists U.\exists V. \quad \begin{pmatrix} -1 & 0 & 1 & -1 \\ 0 & -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ U \\ V \end{pmatrix} \doteq 0 \tag{1}$$

Given that the matrix is triangular and the variables U, V are existentially quantified, we can read off the set of solutions of $E(X,Y)$ easily. A variable assignment is a solution over $\mathbb{R}_+$ if and only if it maps X and Y to the same number:

$$sol^{\mathbb{R}_+}(E(X,Y)) = \{\alpha : \{X,Y\} \to \mathbb{R}_+ \mid \alpha(X) = \alpha(Y)\}$$

Email address: emilie.allart@univ-lille.fr (Emilie Allart)

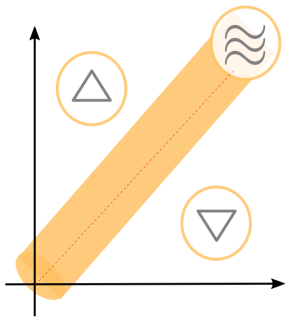


Figure 1: The difference abstraction to Δ_3.

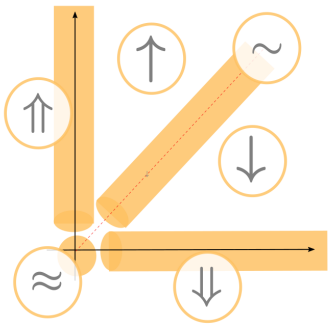


Figure 2: The difference abstraction to Δ_6.

An instance of the question studied in this article is, what happens to the value of Y if we increase the value of X, while jumping from one solution to another? Clearly, the value of Y must be increased as well, since X and Y must have the same value in all solutions. More generally, we want to compute all possible relationships of any two solutions of $E(X, Y)$ over the positive reals.

For making this problem more precise, we consider a partition of the space $\mathbb{R}_+^2$ of concrete differences into the set of abstract differences $\Delta_3 = \{\triangle, \triangledown, \approxeq\}$, which is illustrated in Figure 1. The abstract difference $\triangle$ stands for an increase, $\approxeq$ for a no-change, and $\triangledown$ for a decrease. Given a finite set of variables V, we define the difference abstraction of two variable assignments to the positive reals $\alpha, \alpha' : V \to \mathbb{R}_+$ as the assignment $\beta : V \to \Delta_3$ such that for all $x \in V$:

$$\beta(x) = \begin{cases} \triangle & \text{if } \alpha(x) < \alpha'(x) \\ \approxeq & \text{if } \alpha(x) = \alpha'(x) \\ \triangledown & \text{if } \alpha(x) > \alpha'(x) \end{cases}$$

Our objective then is to compute for any given linear equation system the set of difference abstractions of any two positive real solutions. In our example system $E(X, Y)$, the expected answer is $\{\beta : \{X, Y\} \to \Delta_3 \mid \beta(X) = \beta(Y)\}$. In general, the computation can be done by a naive generate and test the algorithm that we will describe below. This naive algorithm, however, is too slow in practice, to be applicable to change prediction tasks in systems biology. This is why the previous approach by John et.al [23, 7, 17] computed an overapproximation only, by solving the equation system over the canonical relational structure with finite domain Δ_3 and signature $\{+, *, 0, 1\}$ by finite domain constraint programming [21, 22]. The question we study in the present article is whether there exists a better algorithm useful for change prediction in systems biology, that can compute the difference abstraction exactly while avoiding any overapproximation.

Furthermore, two variants of the above problem must be supported for practical application to systems biology. First, we need to be able to treat a refined difference abstraction with 6 values $\Delta_6 = \{\uparrow, \downarrow, \sim, \Uparrow, \Downarrow, \approx\}$ illustrated in Figure 2. Note that $\triangle = \uparrow \uplus \Uparrow$, $\triangledown = \downarrow \uplus \Downarrow$ and $\approxeq = \sim \uplus \approx$, depending on whether the change started or ended with 0 or not. Second, for capturing partial kinetic information up to similarity, we must be able to impose additional constraints on the abstract solutions we are interested in. These will be given by some first-order formulas that are to be interpreted over the abstract domain.

Contributions. First, we generalize the difference abstractions $h_{\Delta_3} : \mathbb{R}_+^2 \to \Delta_3$ and $h_{\Delta_6} : \mathbb{R}_+^2 \to \Delta_6$ algebraically to Σ-abstractions, which are homomorphisms between the Σ-structures. The set of concrete differences $\mathbb{R}_+^2$ becomes a Σ-algebra with signature $\Sigma = \{+, *, 0; 1\}$ equipped

with pointwise addition and multiplication. The sets of abstract differences Δ_3 and Δ_6 become Σ-structures that are naturally induced by the finite partitioning of $\mathbb{R}_+^2$ in Figures 1 and 2.

Second, we show that John's overapproximation theorem [23, 17] can be lifted to general Σ-abstractions $h : S \to \Delta$ and arbitrary positive Σ-formulas ϕ of first-order logic:

$$h \circ sol^S(\phi) \subseteq sol^\Delta(\phi)$$

This inclusion states that all h-abstractions of S-solutions of the formula are also Δ-solutions. And if Δ is finite in addition, then we can compute $sol^\Delta(\phi)$ by finite domain constraint programming and this was shown to be applicable in the application to systems biology [7].

The objective of this article, however, is to compute $h \circ sol^S(\phi)$ exactly, in the case where ϕ is a system of linear equations possibly with existential quantifiers (which clearly can be seen as Σ-formulas) and $h = h_{\Delta_3}$ or $h = h_{\Delta_6}$. The problematic with John's overapproximation can already be illustrated at the example of $E(X, Y)$. When interpreted over the Σ-structure Δ_3, the system $E(X, Y)$ admits the abstract solution $\beta = [X/\triangle, Y/\triangledown]$, since $\triangle +^{\Delta_3} \triangledown$ may be related to any value in Δ_3 nondeterministically. However, β is not the difference abstraction of any concrete solution, given that $E(X, Y)$ implies $X \doteq Y$. In other words, $X \doteq Y$ is a logical consequence of $E(X, Y)$ over $\mathbb{R}_+$, but not over Δ_3, so precisely the information that we seek in our example is lost by John's overapproximation.

In order to avoid any overapproximation, one may want to enrich the linear equation system before solving it over the finite abstract structure, by adding all its logical consequences over $\mathbb{R}_+$, which correspond to all the linear combinations of the rows of the matrix (its row space). However this leads to an infinite number of consequences, so one would be forced to consider them modulo equivalence up to abstract interpretation. Since the abstract domains considered are finite, the number of equivalence classes would be finite too. However, finding a representative for each equivalence class corresponds to solving a linear programming problem, and the number of consequences to be added is exponential in the dimension of the matrix. Therefore, even if this approach leads to a finite representation of the set of logical consequences of the linear equation system, it is still unfeasible in practice for complexity reasons. A less inefficient approach but still infeasible in practice is detailed in the next section: it consist of checking the satisfiability of each abstract solution candidate individually by enumeration.

As a third contribution we propose a new heuristic based on minimal support consequence to improve John's overapproximation algorithm. Given a system of linear equations, the idea is to add all linear consequences that have a minimal number of variables and normalized coefficients, before computing the abstract solutions. We show how to compute this finite set of linear consequences based on elementary modes and orthogonal complements.

As a fourth and most important contribution, we present algorithms for computing the difference abstractions for linear equation system exactly. They can deal with Δ_3 and Δ_6 and with the addition of constraints on the abstract differences. The exact abstraction problems are reduced to finite domain constraint problems, that can be solved in practice with existing finite domain constraint solvers. The reductions are based on properties of first-order definitions that permit to reason with concrete differences in a first-order logic with pairs. Furthermore, we rely on a recent algorithm [3] for the exact rewriting of linear equation systems with respect to the boolean abstraction $h_{\mathbb{B}} : \mathbb{R}_+ \to \mathbb{B}$, which maps 0 to 0 and all other positive real numbers to 1. The linear equation system ϕ obtained by exact rewriting for the boolean abstraction satisfies $h_{\mathbb{B}} \circ sol^{\mathbb{R}_+}(\phi) = sol^{\mathbb{B}}(\phi)$ and is based on on the computation of elementary modes.

Fifth, we implemented the minimal support heuristica and our exact abstraction algorithm for Δ_6, and applied them to the prediction of leucine overproduction, a benchmark task for change prediction in systems biology [23, 7]. It turns out that the minimal support heuristics indeed computes the difference abstraction to Δ_6 exactly for this benchmark, while it does do

so in general. 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.

This article extends on a conference paper at CMSB'2019 [2], which was an extended abstract without any proofs, where most of the technical difficulties could not be exposed. In particular, we could not explain the technical difficulties in decomposing the difference abstractions for Δ_3 and Δ_6 into the boolean abstraction and functions defined in first-order logic. Also we could not describe in any sufficient detail how the addition of kinetic constraint can be dealt with. The major part of the additional content consists of detailed sections about computing exactly the difference abstractions to Δ_3 and Δ_6 also in the presence of kinetic constraints. The case of Δ_6 is considerably more difficult to treat, since it requires more advanced kinds of first-order definitions. The paper now contains full proofs, of which some simpler inductions are delegated to the appendix.

2. Related Work

We first discuss the naive solution to our problem based on existing results for linear equations systems. Given an abstract difference $d \in \Delta_3$ and two variables y, z we define a formula saying that the abstraction of the concrete difference denoted by (y, z) is equal to d:

$$abs_d(y,z) = \begin{cases} y < z & \text{if } d = \triangle \\ x \doteq y & \text{if } d = \approx \\ z < y & \text{if } d = \triangledown \end{cases}$$

Let $\mathbf{x} = x_1 \dots x_m$, $\mathbf{y} = y_1 \dots y_m$ and $\mathbf{z} = z_1 \dots z_m$ be sequences of distinct variables. Given an assignement of the variable in $\mathbf{x}$ to abstract differences $\beta : \{\mathbf{x}\} \to \Delta_3$ we define a formula $abs_\beta(\mathbf{y}, \mathbf{z})$ as follows:

$$abs_\beta(\mathbf{y}, \mathbf{z}) = \bigwedge_{i=1}^{m} abs_{\beta(x_i)}(y_i, z_i)$$

The assignment β then is the difference abstraction of some pair of solutions in $sol^{\mathbb{R}_+}(E(\mathbf{x}))$ if and only if the following formula is satisfiable over $\mathbb{R}_+$.

$$E(\mathbf{y}) \wedge E(\mathbf{z}) \wedge abs_\beta(\mathbf{y}, \mathbf{z})$$

Lemma 1. *The satisfiability over $\mathbb{R}_+$ of systems of homogeneous linear equation and strict linear inequations without constants is in polynomial time.*

Proof A strict linear inequation without constants $x < y$ is equivalent over $\mathbb{R}_+$ to $\exists z.\ x + z \doteq y \wedge z \neq 0$. Therefore, it is sufficient to consider the satisfiability over $\mathbb{R}_+$ of systems of homogeneous linear equation $E(\mathbf{x})$ and nonzero equation $\mathbf{x}' \neq 0$. The solution set $sol^{\mathbb{R}_+}(E(\mathbf{x}))$ is a cone (while for more general linear programs it could be more general polytopes). The elements of a cone can be multiplied by positive reals without leaving the cone. Therefore, the nonzero equation $\mathbf{x}' \neq 0$ can be rewritten to $\mathbf{x}' \geq 1$ without affecting the satisfiability of the formula. This rewriting eliminates the strict inequations without constants at the cost of introducing nonstrict inequations with constants. The result is a linear program. Instead of interpretation over $\mathbb{R}_+$ we can add inequations $\mathbf{x} \geq 0 \wedge \mathbf{x}' \geq 0$ and change to an interpretation over $\mathbb{R}$. It well known that the satisfiability of linear programs over $\mathbb{R}$ can be tested in polynomial time [18]. □

So by a naive enumerate and test algorithm, we can compute in the case of Δ_3 the set of all difference abstractions for $sol^{\Delta_3}(E(\mathbf{x}))$ in time $O(3^m poly(|E(\mathbf{x})|))$. For Δ_6, the analogous argument yields $O(6^m poly(|E(\mathbf{x})|))$.

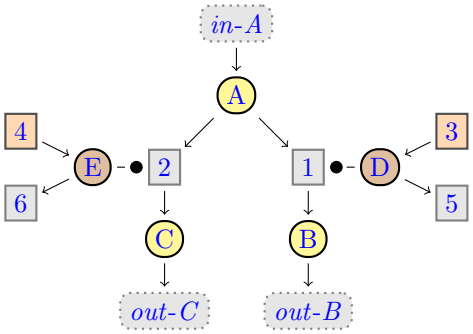


Figure 3: An example of reaction network with partial kinetic information.

Linear equation system over $\mathbb{R}_+$:	Nonlinear constraints over Δ_6:
$v_{in\text{-}A} \doteq v_1 + v_2$	$v_1 \doteq \mathrm{A} * \mathrm{D}$
$v_1 \doteq v_{out\text{-}B}$	$v_2 \doteq \mathrm{A} * \mathrm{E}$
$v_2 \doteq v_{out\text{-}C}$	$v_3 \in \{\Downarrow, \sim\}$
$v_4 \doteq v_6$	$v_4 \in \{\Downarrow, \sim\}$
$v_3 \doteq v_5$	$v_5 \doteq \mathrm{E}$
	$v_6 \doteq \mathrm{D}$
	$v_{out\text{-}B} \doteq \mathrm{B}$
	$v_{out\text{-}C} \doteq \mathrm{C}$

Figure 4: Steady state semantics.

Flux balance analysis [24, 25] 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 equation systems that describe the rates of the reactions in a steady state of the metabolic network, by using Gaussian elimination, elementary flux modes (EFMs) [19], or optimisation methods [20, 9]. Most importantly, precise quantitative kinetic information is not required in contrast to classical mathematical analysis methods for reaction networks. 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 [6, 11, 8] has been exploited to design novel algorithms [23, 17] 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.

3. Application to Change Prediction in Systems Biology

Reaction networks [12, 5, 16, 10] are widely used in systems biology to model the dynamics of biological systems, so that their behaviour can be simulated or analysed. We are interested in change predictions for reaction networks with partial kinetic information [23, 7, 17]. The steady state semantics of such networks yields a system of linear equations, and a set of nonlinear constraints about the differences abstraction in Δ_6 of solutions of the linear system.

A simple example of reaction network with partial formal kinetic information is given in Figure 3; the linear equation system and the nonlinear constraints on its difference abstraction are given in Figure 4. The networks has five species A, B, C, D, E and nine reactions 1, ..., 6, *in-A*, *out-B*, *out-C*. As with the graphical notation for Petri nets, the species are nodes drawn as circles and the reactions nodes drawn as boxes. The colors of the species indicate their biological role, but do not contribute to the semantics. Metabolites are drawn in yellow and enzymes in brown circles. Reaction *in-A* is an inflow of the metabolite A, while reactions *out-B* and *out-C* are outflows of the metabolites B and C respectively. The inflows are controlled externally, while the outflows are controlled internally in the system. Reactions 3 and 4 correspond to the

gene expression of the enzymes C and D. These reactions may be knocked out, modeling a gene knockout. The knockouts are changes that are controlled externally similarly to the change of inflow *in-A*. None of the other reactions can be changed externally. Reaction 6 degrades the enzyme C and reaction 7 degrades the enzyme D. Reaction 1 transforms its metabolic substrate A, into its metabolic product B, while being activated and accelerated by the enzyme C. Symmetrically, the reaction 2 transforms its metabolic substrate A, into its metabolic product C, while being activated and accelerated by the enzyme D.

In a steady state of the reaction network, the concentrations of all its species become stable. For each species we have a linear equation, that states that the rate of its production is equal to the rate of its consumption. For species A, for instance, this is the following equation over $\mathbb{R}_+$, since species A is produced by inflow *in-A* at rate $v_{in\text{-}A}$, while it is consumed by reaction 1 at rate v_1 and reaction 2 at rate v_2:

$$v_{in\text{-}A} \doteq v_1 + v_2$$

The network has partial kinetic information: we know the enzymes (and inhibitors) of the reactions, but not necessarily their precise kinetics. For instance, the precise kinetics of reaction 1 is unknown. But since D is an enzyme of reaction 1 it follows that the rate v_1 is zero if the concentration of species E is zero, and that v_1 increases if the concentration of species E increases. Furthermore, since A is a substrate of reaction 1 it follows that v_1 is zero if the concentration of species A is zero, and that v_1 increases if the concentration of species A increases. This means that the following nonlinear constraint holds after difference abstraction to Δ_6:

$$v_1 \doteq \mathrm{A} * \mathrm{D}$$

For reactions 3 and 4 which may be knocked out, we have the following constraints about their difference abstraction to Δ_6:

$$v_3 \in \{\Downarrow, \sim\}$$
$$v_4 \in \{\Downarrow, \sim\}$$

So the rates of these reactions may either decrease to zero or remain unchanged but different from zero.

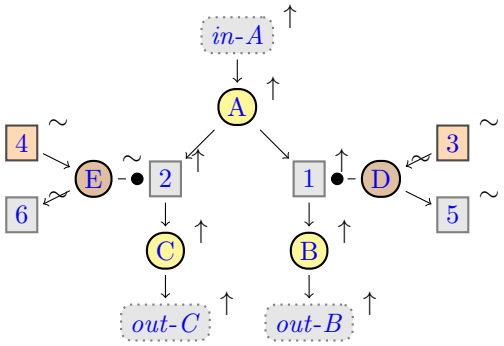
A typical question for change prediction is which changes can be applied to the example network in order to increase the outflow of B, that is $v_{out\text{-}B} \doteq \uparrow$. The set of potential changes is to increase or decrease the inflow of A, i.e., $v_{in\text{-}A} \in \{\uparrow, \downarrow\}$, or to shut down reactions 3 or 4, that is $v_3 = \Downarrow$ or $v_4 = \Downarrow$. The two single-change predictions that answer this question are:

1. increase the inflow of A, that is $v_{in\text{-}A} \doteq \uparrow$, or
2. knock down reaction 4, i.e., $v_4 \doteq \Downarrow$ and thus of the gene expression of enzyme E.

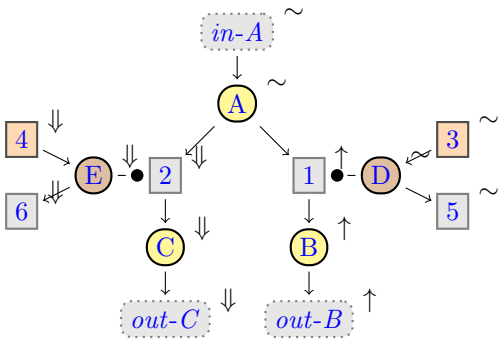
These two predictions correspond to the two abstract solutions over Δ_6 in Figure 5. The first solution in Figure 5a motivates the prediction of an increase of *in-A* and the second solution in Figure 5b the prediction of a knock down of reaction 4, the gene expression producing enzyme E.

Both abstract solutions are difference abstractions over Δ_6 of real positive solutions of the linear equation system in Figure 4, so that these difference abstractions do also satisfy the nonlinear constraints over Δ_6 given there. We can find both predictions by applying John's overapproximation algorithm [23, 17], that is by computing the Δ_6 solutions of linear equation system and the nonlinear constraints. This can be done in practice by using finite domain constraint programming.

Qualitative reasoning can also be performed manually for this simple example. For increasing *out-B* be must increase the concentration of B and thus the rate of reaction 1. For this, we must either increase the concentration of enzyme D which is impossible by the available changes, or



(a) Increase inflow of A.



(b) Knock down gene expression 4 of E.

Figure 5: Two changes leading to an increase of the outflow of B.

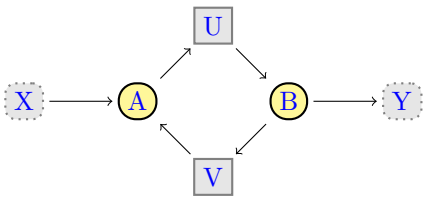


Figure 6: A reaction network with a simple loop.

$$U \doteq X + V$$
$$U \doteq Y + V$$

Figure 7: Linear equation system of the reaction network in Figure 6 with the simple loop.

increase the concentration of A. The latter requires to either increase the inflow of A, leading to the first abstract solution in Figure 5a, or else decrease the rate of reaction 2. This is possible by decreasing the concentration of E by knocking out the reaction 4, the gene expression producing this enzyme. This yields the second abstract solution in Figure 5b.

John's algorithm does not lead to any overapproximation for this example. The main reason is that the graph of the reaction network in Figure 3 is acyclic, even in the absence of partial kinetic information. The situation changes for reaction networks with cycles. The simplest counterexample is the simple loop in Figure 6. The linear equation system of this network is exactly the system from the introduction, where John's algorithm predicts unjustified changes.

This network has two species X and Y and four reactions: an inflow of X, an outflow of Y, a reaction U transforming X to Y and a inverse reaction V. So each molecule X that flows into the system may loop for a while, changing to Y and back, before sometimes outflowing as Y. In a steady state, the rate of the inflow of X is equal to the rate of the outflow of Y. The argument can be understood more easily, when considering the elementary flux modes [29, 26] of this reaction network, which are shown graphically in Figure 8 in red and respectively in green. An elementary flux mode is a linear combination of reactions that can become steady. The simple loop network has two elementary flux modes, corresponding to the linear combination of reactions $1X + 1U + 1Y$ using red edges and the linear combination of reactions $1U + 1V$ with green edges respectively.

While some molecules may loop with the green edges, transforming A's to B's and back, all inflowing A's must follow the red edges and thus eventually outflow as B. From an algebraic per-

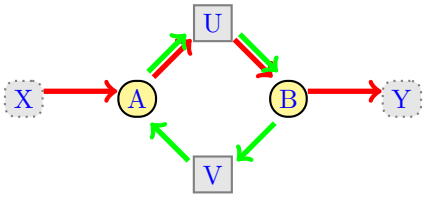


Figure 8: Elementary flux modes of the simple loop network.

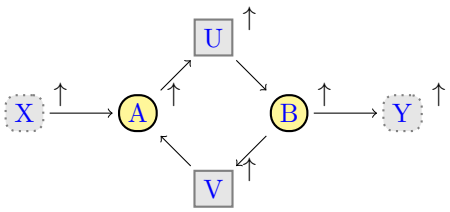


Figure 9: A justified abstract solution.

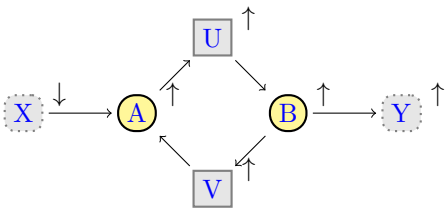


Figure 10: An unjustified abstract solution.

spective, the elementary flux modes of a reaction network correspond exactly to the elementary modes of its stoichiometry matrix [15]. An elementary mode of a matrix A is a positive solution of $A\mathbf{x} \doteq 0$ with minimal support (a minimal number of reactions) that is normalized. Note that the stoichiometry matrix of the simple loop network was given within the linear equation system (1) at the beginning of the introduction. Its elementary modes are $(1, 1, 1, 0)$ and $(0, 0, 1, 1)$.

The difference abstraction of any two concrete solutions of the linear equations of the simple loop network must satisfy $X \doteq Y$. The corresponding abstract solution over Δ_6 satisfying $X \doteq \uparrow$ and $Y \doteq \uparrow$ is illustrated in Figure 9. However, the alternative variable assignment to Δ_6 in Figure 10 is also a solution of all linear equations when interpreted over Δ_6, while not being justified by any pair of concrete solutions over $\mathbb{R}_+$.

The intuitive reason for this failure is that John's overapproximation algorithm performs only local reasoning, considering one linear equation at a time, i.e., one species of the reaction network. In this manner, it cannot see, that $X \doteq Y$ is a logical consequence of the linear equation system of the network over $\mathbb{R}_+$, while it is not a consequence over Δ_6. So what we are searching is way to reason globally with all species of a reaction network at a time. For this we have to take into account all linear combinations of the equations of the system.

4. Preliminaries

We present standard notion of sets, partial and total functions, relations, Σ-algebras and Σ-structures.

4.1. Set and Functions

We start with the 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 denoted by $|A|$.

We continue with notion for total and partial functions. A partial function is a relation $f \subseteq A \times B$ such that for all $a \in A$ there exists at most one $b \in B$ such that $(a, b) \in f$. In this case, we write $f(a) = b$. The domain of the partial function is $dom(f) = \{a \mid \exists b \in B. f(a) = b\}$ and its range $ran(f) = \{b \mid \exists a \in A.\ f(a) = b\}$. A total function $f : A \to B$ is a partial function $f \subseteq A \times B$ such that $dom(f) = A$. Given a total function $f : A \to B$ and a partial function $g : B \times C$ such that $ran(f) \subseteq dom(g)$ we define the function composition as the total function $g \circ f : A \to C$ such that $(g \circ f)(x) = g(f(x))$ for all $x \in A$. Furthermore if $R \subseteq \{f : A \to B\}$ then we define:

$$g \circ R = \{g \circ f : A \to C \mid f \in R,\ ran(f) \subseteq dom(g)\}$$

Note that $g \circ f$ is defined only if $ran(f) \subseteq dom(g)$, so functions $f \in R$ violating this condition will be ignored all over in the composition $g \circ R$. This is since we want all the functions in $g \circ R$ to be total even if g is partial.

4.2. Σ-Algebras and Σ-Structures

We next recall the notions of Σ-algebras, Σ-structures, and homomorphism between Σ-structures. These classical notions of universal algebra will be fundamental to our algebraic generalization of difference abstractions to the notion of Σ-abstractions in Definition 8.

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 2. *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 3. *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.

It is sometimes useful to add the elements of the domain of a Σ-structure A to the constants. Therefore, we define the extended signature:

$$\Sigma[dom(A)] = \Sigma \uplus dom(A)$$

The Σ-structure A can be lifted to a $\Sigma[dom(A)]$-structure by interpreting the new constants by themselves, i.e., $a^A = a$ for all $a \in dom(A)$, and all symbols in Σ as before.

Definition 4. *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* $(h(s_1), \ldots, h(s_{n+1})) \in f^\Delta$*.*

For Σ-algebras, the second condition is equivalent to $h(f^S(s_1,\ldots,s_n)) = f^\Delta(h(s_1),\ldots,h(s_n))$.

For any Σ-structure S we can reinterpret $n+1$ ary relations f^S as n-ary set valued functions. In order to do so, we define for any sequence $s_1,\ldots,s_n \in dom(S)$ a subset of values:

$$f^\Delta(s_1,\ldots,s_n) = \{s \in dom(S) \mid (s_1,\ldots,s_n,s) \in f^\Delta\}$$

With this set-valued reinterpretation, the second condition of homomophisms can be rewritten equivalently to:

$$h(f^S(s_1,\ldots,s_n)) \subseteq f^\Delta(h(s_1),\ldots,h(s_n))$$

5. Σ-Abstractions

We introduce the concept of Σ-abstractions for general signatures. Before doing so, we start with an example for a Σ-abstraction, which is the boolean abstraction of positive real numbers. It has the signature of arithmetics $\Sigma = F^{(2)}_{pos\text{-}arith} \uplus C_{pos\text{-}arith}$ with two binary function symbols and two constants such that:

$$F^{(2)}_{pos\text{-}arith} = \{+,*\}$$
$$C_{pos\text{-}arith} = \{0,1\}$$

For all Σ-algebras considered, the operators $+^S$ and $*^S$ are associative and commutative, with neutral element 0^S and 1^S respectively.

Example 5. *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 $(\mathbb{R}_+, .^{\mathbb{R}_+})$ whose domain $dom(\mathbb{R}_+)$ is equal to the set of positive reals $\mathbb{R}_+$.*

Example 6. *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 $(\mathbb{B}, .^{\mathbb{B}})$ whose domain $dom(\mathbb{B})$ is the set of booleans $\mathbb{B}$.*

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 7. *The function $h_{\mathbb{B}} : \mathbb{R}_+ \to \mathbb{B}$ is a homomorphism between Σ-algebras where $\Sigma = F^{(2)}_{pos\text{-}arith} \uplus C_{pos\text{-}arith}$.*

Proof For all $r, r' \in \mathbb{R}_+$ we have:

$$h_{\mathbb{B}}(r +^{\mathbb{R}_+} r') = 1 \quad\Leftrightarrow\quad r +^{\mathbb{R}_+} r' \neq 0 \quad\Leftrightarrow\quad r \neq 0 \vee r' \neq 0 \quad\Leftrightarrow\quad h_{\mathbb{B}}(r) = 1 \vee h_{\mathbb{B}}(r') = 1$$
$$h_{\mathbb{B}}(r *^{\mathbb{R}_+} r') = 1 \quad\Leftrightarrow\quad r *^{\mathbb{R}_+} r' \neq 0 \quad\Leftrightarrow\quad r \neq 0 \wedge r' \neq 0 \quad\Leftrightarrow\quad h_{\mathbb{B}}(r) = 1 \wedge h_{\mathbb{B}}(r') = 1$$

Hence $h_{\mathbb{B}}(r +^{\mathbb{R}_+} r') = h_{\mathbb{B}}(r) +^{\mathbb{B}} h_{\mathbb{B}}(r')$ and $h_{\mathbb{B}}(r *^{\mathbb{R}_+} r') = h_{\mathbb{B}}(r) *^{\mathbb{B}} h_{\mathbb{B}}(r')$. Finally, for both constants $c \in C$ we have that $h_{\mathbb{B}}(c^{\mathbb{R}_+}) = h_{\mathbb{B}}(c) = c = c^{\mathbb{B}}$. □

The boolean abstraction $h_{\mathbb{B}}$ is the prime example of what we will call a Σ-abstraction. The following definition applies for general signatures.

Definition 8. *A Σ-abstraction is a homomorphism between Σ-structures S and Δ such that $dom(\Delta) \subseteq dom(S)$.*

We assume that $dom(\Delta) \subseteq dom(S)$ since this will permit us to reason about Σ-abstractions by talking at the same time about concrete value in $dom(S)$ and abstract values in $dom(\Delta)$ by first-order formulas interpreted over the Σ-structure S.

6. Abstracting Concrete Differences

Concrete differences are pairs of positive in $\mathbb{R}_+^2$. We show how to abstract concrete differences into abstract differences. For this, we consider $\mathbb{R}_+^2$ as a Σ-algebra that we then abstract into finite Σ-structures Δ_3 and Δ_6.

6.1. The Tuple Σ-Algebra S^n

For any Σ-algebra S where $\Sigma = F^{(2)} \cup C$ 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 deliberately 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 9. *If h is a Σ-abstraction from S to Δ then h^n is a Σ-abstraction from S^n to Δ^n.*

Proof Let $\odot \in F^{(2)}$ and $t = (s_1, \ldots, s_n), t' = (s'_1, \ldots, s'_n) \in dom(S)^n$. Then we have:

$$\begin{array}{lll}
h^n(t \odot^{S^n} t') & = (h(s_1 \odot^S s'_1), \ldots, h(s_n \odot^S s'_n)) & \text{definitions of } h^n \text{ and } S^n \\
& = (h(s_1) \odot^\Delta h(s'_1), \ldots, h(s_n) \odot^\Delta h(s'_n)) & \text{since } h \text{ is homomorphism} \\
& = (h(s_1), \ldots, h(s_n)) \odot^{\Delta^n} (h(s'_1), \ldots, h(s'_n)) & \text{definition of } \Delta^n \\
& = h^n(t) \odot^{\Delta^n} h^n(t') & \text{definition of } h^n
\end{array}$$

Finally, for both constants $c \in C$ we have:

$$\begin{array}{lcll}
h^n(c^{S^n}) & = & h^n(c^S, \ldots, c^S) & \text{definition } S^n \\
& = & (h(c^S), \ldots, h(c^S)) & \text{definition } h^n \\
& = & (c^\Delta, \ldots, c^\Delta) & \text{since } h \text{ is homomorphism} \\
& = & c^{\Delta^n} & \text{definition of } \Delta^n
\end{array}$$

□

6.2. Abstractions of Concrete Differences

Given that $\mathbb{R}_+$ is a Σ-algebra with signature $\Sigma = F^{(2)}_{pos\text{-}arith} \cup C_{pos\text{-}arith}$, we have that $\mathbb{R}^2_+$ is also a Σ-algebra with the same signature.

We now show how to abstract the concrete differences in $\mathbb{R}^2_+$ to abstract difference. A generic manner to do so is to start with some some partition $h : \mathbb{R}^2_+ \to \Delta$ into a finite set Δ. The elements of this set will be called the *abstract differences*. The function h says how to abstract concrete to abstract differences. Since it is a partition, it splits $\mathbb{R}^2_+$ into finitely many equivalence classes.

For any partition $h : \mathbb{R}^2_+ \to \Delta$, there is a unique manner to define an interpretation $.^\Delta$ such that $(\Delta, .^\Delta)$ becomes Σ-structure with domain Δ 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 $d_1, d_2 \in \Delta$:

$$d_1 \odot^\Delta d_2 = \{h(r_1 \odot^{\mathbb{R}_+} r_2, r'_1 \odot^{\mathbb{R}_+} r'_2) \mid h(r_1, r'_1) = d_1,\ h(r_2, r'_2) = d_2\}$$

Lemma 10. *$h : \mathbb{R}^2_+ \to \Delta$ is a Σ-abstraction where $\Sigma = F^{(2)}_{pos\text{-}arith} \cup C_{pos\text{-}arith}$.*

Proof For any $p_1 = (r_1, r'_1)$, $p_2 = (r_2, r'_2) \in \mathbb{R}^2_+$ the second condition for homomorphisms follows for all $\odot \in F^{(2)}_{pos\text{-}arith}$:

$$h(p_1 \odot^{\mathbb{R}^2_+} p_2) \quad = h(r_1 \odot^{\mathbb{R}_+} r_2, r'_1 \odot^{\mathbb{R}_+} r'_2) \quad \in h(p_1) \odot^\Delta h(p_2)$$

Finally, for all constants $c \in C_{pos\text{-}arith}$ we have by definition that $h(c^{\mathbb{R}^2_+}) = c^\Delta$. □

6.3. The Σ-Structure Δ_3

We continue with the signature of arithmethics $\Sigma = F^{(2)}_{pos\text{-}arith} \uplus C_{pos\text{-}arith}$ Our next objective is to recall the abstraction of concrete differences from the Σ-algebra $\mathbb{R}_+$ into the finite Σ-structure with domain $\Delta_3 = \{\triangle, \triangledown, \approxeq\}$ that is well-known from qualitative reasoning (see e.g. [13]). For this we start with the function $h_{\Delta_3}(r, r') \in \Delta_3$ such that for any $r, r' \in \mathbb{R}_+$:

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

We define the ternary relation $+^{\Delta_3}$ as the relation that is symmetric in the first two arguments and has the set-valued reinterpretation $d +^{\Delta_3} d' \subseteq \Delta_3$ in Figure 11 for all $d, d' \in \Delta_3$. The definition of $*^{\Delta_3}$ is given in analogy in the same figure. The interpretation of the constants are $1^{\Delta_3} = 0^{\Delta_3} = \approxeq$. By Lemma 10, $h_{\Delta_3} : \mathbb{R}^2_+ \to \Delta_3$ is a Σ-abstraction.

6.4. 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 [23]. 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 $r, r' \in \mathbb{R}_+$:

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

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

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

c	c^{Δ_3}
0	$\approx$
1	$\approx$

Figure 11: Interpretation of Σ-structure Δ_3.

d	d'	$d +^{\Delta_6} d'$
$\uparrow$	$\uparrow$	$\{\uparrow\}$
$\uparrow$	$\downarrow$	$\{\uparrow, \sim, \downarrow\}$
$\uparrow$	$\sim$	$\{\uparrow\}$
$\uparrow$	$\Uparrow$	$\{\uparrow\}$
$\uparrow$	$\Downarrow$	$\{\uparrow, \downarrow, \sim\}$
$\uparrow$	$\approx$	$\{\uparrow\}$

d	d'	$d +^{\Delta_6} d'$
$\Uparrow$	$\downarrow$	$\{\uparrow, \sim, \downarrow\}$
$\Uparrow$	$\sim$	$\{\uparrow\}$
$\Uparrow$	$\Uparrow$	$\{\Uparrow\}$
$\Uparrow$	$\Downarrow$	$\{\uparrow, \sim, \downarrow\}$
$\Uparrow$	$\approx$	$\{\Uparrow\}$
$\Downarrow$	$\Downarrow$	$\{\Downarrow\}$

d	d'	$d +^{\Delta_6} d'$
$\sim$	$\sim$	$\{\sim\}$
$\sim$	$\approx$	$\{\sim\}$
$\sim$	$\downarrow$	$\{\downarrow\}$
$\sim$	$\Downarrow$	$\{\downarrow\}$
$\downarrow$	$\downarrow$	$\{\downarrow\}$
$\downarrow$	$\Downarrow$	$\{\downarrow\}$

d	d'	$d +^{\Delta_6} d'$
$\approx$	$\approx$	$\{\approx\}$
$\approx$	$\downarrow$	$\{\downarrow\}$
$\approx$	$\Downarrow$	$\{\Downarrow\}$

c	c^{Δ_6}
0	$\approx$
1	$\sim$

d	d'	$d *^{\Delta_6} d'$
$\uparrow$	$\uparrow$	$\{\uparrow\}$
$\uparrow$	$\downarrow$	$\{\uparrow, \sim, \downarrow\}$
$\uparrow$	$\sim$	$\{\uparrow\}$
$\uparrow$	$\Uparrow$	$\{\Uparrow\}$
$\uparrow$	$\Downarrow$	$\{\Downarrow\}$
$\uparrow$	$\approx$	$\{\approx\}$

d	d'	$d *^{\Delta_6} d'$
$\Uparrow$	$\uparrow$	$\{\Uparrow\}$
$\Uparrow$	$\sim$	$\{\Uparrow\}$
$\Uparrow$	$\Uparrow$	$\{\Uparrow\}$
$\Uparrow$	$\Downarrow$	$\{\approx\}$
$\Uparrow$	$\approx$	$\{\approx\}$
$\Downarrow$	$\Downarrow$	$\{\Downarrow\}$

d	d'	$d *^{\Delta_6} d'$
$\sim$	$\sim$	$\{\sim\}$
$\sim$	$\approx$	$\{\approx\}$
$\sim$	$\downarrow$	$\{\downarrow\}$
$\sim$	$\Downarrow$	$\{\Downarrow\}$
$\downarrow$	$\downarrow$	$\{\downarrow\}$
$\downarrow$	$\Downarrow$	$\{\Downarrow\}$

d	d'	$d *^{\Delta_6} d'$
$\approx$	$\approx$	$\{\approx\}$
$\approx$	$\downarrow$	$\{\Downarrow\}$
$\approx$	$\Downarrow$	$\{\Downarrow\}$

Figure 12: Interpretation of Σ-structure Δ_6.

We define the ternay relation $+^{\Delta_6}$ as the relation that is symmetric in the first two arguments and has the set-valued reinterpretation $d +^{\Delta_6} d' \subseteq \Delta_6$ in Figure 12 for all $d, d' \in \Delta_6$. The relation $*^{\Delta_6}$ is defined in the same style in Figure 12. The constants are interpreted as $0^{\Delta_6} = \approx$ and $1^{\Delta_6} = \sim$. By Lemma 10, $h_{\Delta_6} : \mathbb{R}_+^2 \to \Delta_6$ is a Σ-abstraction.

7. 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.

7.1. Standard First-Order Logic

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 signature $\Sigma = F^{(2)} \uplus C$ is arbitrary here.

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 Figure 13 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 $e \dot{\neq} 0$ by $\neg e \doteq 0$.

First-order expressions and formulas:

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

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

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

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

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

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

Figure 13: Syntax and semantics of expressions and formulas of first-order logic.

The semantics of expressions in Figure 13 is defined such that the following formula becomes true in the structure Δ_3 taken with the signature extended with extra constants $\Sigma[dom(\Delta_3)]$:

$$\begin{array}{ll} & \triangle + \triangledown \doteq \triangle \\ \wedge & \triangle + \triangledown \doteq \triangledown \\ \wedge & \triangle + \triangledown \doteq \approx \end{array}$$

The first reason is that relations are reinterpreted as set-valued functions by the semantics of first order logic. In paricular, we have $\triangle +^{\Delta_3} \triangledown = \Delta_3$. The second reason is that the meaning of the equality operator $\doteq$ of the logic is nondetermistic equality, that is the nondisjointness. Also note that the following formula is unsatisfiable:

$$\exists x.\ (x \doteq \triangle \ \wedge \ x \doteq \triangledown)$$

This is since for any variable assignment the expression x must evaluated to a singleton, which cannot contain both $\triangle$ and $\triangledown$. Another way to see this is that $\exists x.\ (x \doteq \triangle \wedge x \doteq \triangledown)$ is equivalent to $\triangledown \doteq \triangle$ which evaluates to false.

More generally, 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 $\alpha : \mathcal{V} \to dom(S)$. Any Σ-expressions $e \in \mathcal{E}_\Sigma$ denotes a subset of values in $dom(S)$, 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 $dom(S)$.

A variable assignment into a Σ-structure S is a partial function $\alpha : V \to dom(S)$ 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 $fv(e) \subseteq V$ can be interpreted as an element of $dom(S)$ and any Σ-formula $\phi \in \mathcal{F}_\Sigma$ with $fv(\phi) \subseteq V$ 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 $V \supseteq fv(\phi)$ is defined by:

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

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

Figure 14: Expressions and formulas of the first-order logic with n-tuples.

If $V = fv(\phi)$ then we omit the index V, i.e., $sol^S(\phi) = sol^S_V(\phi)$.

7.2. First-Order Tuple Logic

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 Figure 14. The expressions $o \in \mathcal{O}^n_\Sigma$ are like the expression $e \in \mathcal{E}_\Sigma$ except that variables x are now replaced by projection expressions $\dot{\pi}_i(x)$ 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:

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

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.

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

Lemma 11. *For any $e \in \mathcal{E}_\Sigma$, Σ-algebra S, $n \geq 1$, and $\beta : V \to dom(S)^n$ with $\mathcal{V}(\phi) \subseteq V \subseteq \mathcal{V}$:*

$$
[\![e]\!]^{\beta,S^n} = [\![(\Pi_1(e), \ldots, \Pi_n(e))]\!]^{\beta,S}
$$

Proof sketch. By induction on the structure of expressions in $\mathcal{E}_\Sigma$. □

Proposition 12. *For any $\phi \in \mathcal{F}_\Sigma$, Σ-algebra S, and $n \geq 1$: $sol^{S^n}(\phi) = n\text{-}sol^S(\langle\phi\rangle^n)$.*

Proof sketch. By induction on the structure of formulas in $\mathcal{F}_\Sigma$. We only show the base case of Σ-equations. So let ϕ is a Σ-equation of the form $e \doteq e'$ where $e, e' \in \mathcal{E}_\Sigma$ and β a variable assignment $\beta : V \to dom(S)^n$ such that $\mathcal{V}(\phi) \subseteq V \subseteq \mathcal{V}$. Then:

$$
\begin{array}{ll}
sol^{S^n}(e \doteq e') & = \{\beta \mid [\![e \doteq e']\!]^{\beta,S^n} = 1\} \\
\textit{Lemma 11} & = \{\beta \mid [\![\bigwedge_{i=1}^n \Pi_i(e) \doteq \Pi_i(e')]\!]^{\beta,S} = 1\} \\
& = \{\beta \mid [\![\langle e \doteq e' \rangle^n]\!]^{\beta,S} = 1\} \\
& = n\text{-}sol^S(\langle e \doteq e' \rangle^n)
\end{array}
$$

The inductive cases for the other formulas are straightforward. □

7.3. Polynomial Equations

In the case of the arithmetic signature $\Sigma = F^{(2)}_{pos\text{-}arith} \uplus C_{pos\text{-}arith}$, the arithmetic equations $e \doteq e' \in \mathcal{F}_\Sigma$ provided by the formulas of standard FO-logic subsume the usual polynomial equations with natural coefficients. We will use the following notation for writing polynomials. For any natural n and expression $e, e_1, \ldots, e_n \in \mathcal{E}_\Sigma$, we define the expression $\prod_{i=1}^n e_i = e_1 * \ldots * e_n$, which is equal to 1 if $n = 0$ and $\sum_{i=1}^n e_i = e_1 + \ldots + e_n$ which is equal to 0 if $n = 0$. Furthermore, let $e^n = \prod_{i=1}^n e$ and $ne =_{\text{def}} \sum_{i=1}^n e$. The analogous definitions no and o^n can be made for object expression in $o \in \mathcal{O}^n_\Sigma$ of the FO-tuple logic instead of expression in $e \in \mathcal{E}_\Sigma$ of the standard first-order logic.

Example 13. *Let $\phi \in \mathcal{F}_\Sigma$ be the polynomial equation $3x + 4y^5 \doteq 0$ of the standard FO-logic. Proposition 12 shows that ϕ has the same solutions over $\mathbb{R}_+^2$ than the formula $\langle\phi\rangle^2 \in \mathcal{F}^2_\Sigma$ of the tuple FO-logic over $\mathbb{R}_+$. The latter is the system of polynomial equations $3\dot\pi_1(x) + 4\dot\pi_1(y)^5 \doteq 0 \wedge 3\dot\pi_2(x) + 4\dot\pi_2(y)^5 \doteq 0$.*

Concrete differences can be described by systems of polynomial equations of the standard FO-logic but interpreted over ${\mathbb{R}_+}^2$. As shown by Proposition 12, such systems can thus be mapped to systems of polynomial equations in the FO-pair logic, but now interpreted over $\mathbb{R}_+$. This is done by copying each equation over ${\mathbb{R}_+}^2$ into two equations of $\mathbb{R}_+$, as illustrated by the above example.

Our next objective is to introduce fresh variables for projections in order to rewrite equation systems from the FO-pair logic into equation systems from the standard FO-logic. For instance, when given two fresh variable generators ν_1 and ν_2, the system of polynomial equations $3\dot\pi_1(x) + 4\dot\pi_1(y)^5 \doteq 0 \wedge 3\dot\pi_2(x) + 4\dot\pi_2(y)^5 \doteq 0$ in the FO-pair logic can be mapped to the systems of polynomial equations $3\nu_1(x) + 4\nu_1(y)^5 \doteq 0 \wedge 3\nu_2(x) + 4\nu_2(y)^5 \doteq 0$ in the standard FO-logic. The 4 fresh variables such as $\nu_i(x)$ and $\nu_i(y)$ correspond to the projections $\dot\pi_i(x)$ and $\dot\pi_i(y)$ respectively. With respect to the standard FO-logic, we can thus rewrite any system of polynomial equation over ${\mathbb{R}_+}^2$ to a system of polynomial equation over $\mathbb{R}_+$, so that the solutions correspond. The unique role of the FO-pair logic is to serve us as an intermediate language for this purpose.

7.4. From Logique des tuples du premier-ordre to Standard Logique du premier-ordre

More generally, we wish to rewrite any FO-tuple formula $\psi \in \mathcal{F}^n_\Sigma$ into a standard FO formula $\tilde\nu(\psi) \in \mathcal{F}_\Sigma$ by introducing fresh variables for projections. For this, we fix n generators of fresh variables $\nu_1, \ldots, \nu_n : \mathcal{V} \to \mathcal{V}$. We then map any expression $o \in \mathcal{O}^n_\Sigma$ with projections to some expressions $\tilde\nu(o) \in \mathcal{E}_\Sigma$ without new variables:

$$\tilde\nu(\dot\pi_i(x)) =_{\text{def}} \nu_i(x), \qquad \tilde\nu(c) =_{\text{def}} c, \qquad \tilde\nu(o \odot o') =_{\text{def}} \tilde\nu(o) \odot \tilde\nu(o').$$

And finally, we map any formula $\psi \in \mathcal{F}^n_\Sigma$ 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 $x \in V$.

Lemma 14. *For any expression $o \in \mathcal{O}^n_\Sigma$ and variable assignment $\beta : V \to dom(S)^n$ with $\mathcal{V}(o) \subseteq V \subseteq \mathcal{V}$ we have $[\![\tilde{\nu}(o)]\!]^{S,\nu(\beta)} = [\![o]\!]^{\beta,S}$.*

Proof sketch. By induction on the structure of Σ-expressions $o \in \mathcal{O}^n_\Sigma$. □

Proposition 15. *For any $\psi \in \mathcal{F}^n_\Sigma$, Σ-structure S, and $n \geq 1$: $n\text{-}sol^S(\psi) = \nu^{-1}(sol^S(\tilde{\nu}(\psi)))$.*

Proof We first prove the following claim is by induction on the structure of Σ-formulas in $\mathcal{F}^n_\Sigma$, where the base case follows from Lemma 14.

Claim 16. *For any variable assignment $\beta : V \to dom(S)^n$ with $V \subseteq \mathcal{V}$ and formula $\psi \in \mathcal{F}^n_\Sigma$ we have $[\![\tilde{\nu}(\psi)]\!]^{S,\nu(\beta)} = [\![\psi]\!]^{\beta,S}$.*

The proof of the claim is straightforward by induction on the structure of Σ-formulas in $\mathcal{F}^n_\Sigma$. Finally, the claim implies the proposition as follows:

$$\begin{array}{rcll} \beta \in n\text{-}sol^S(\psi) & \Leftrightarrow & \nu(\beta) \in n\text{-}sol^S(\tilde{\nu}(\psi)) & \text{previous claim} \\ & \Leftrightarrow & \nu^{-1}(\nu(\beta)) \in \nu^{-1}(n\text{-}sol^S(\tilde{\nu}(\psi))) & \\ & \Leftrightarrow & \beta \in \nu^{-1}(n\text{-}sol^S(\tilde{\nu}(\psi))) & \end{array}$$

□

7.5. Commutation Property

As above, we consider n fresh variable generators $\nu_1, \ldots, \nu_n$ and the operator ν^{-1} that maps object assignments of freshly generates variables to n-tuple assignments. We next show a commutation property of the operator ν^{-1} with Σ-abstractions.

Lemma 17. *For any Σ-abstraction $h : S \to \Delta$ and assignment of fresh variables $\alpha : \uplus_{i=1}^n \nu_i(V) \to dom(S)$:*

$$\nu^{-1}(h \circ \alpha) = h^n \circ \nu^{-1}(\alpha)$$

Proof For any variable $x \in V$ we have:

$$\begin{array}{rcl} \nu^{-1}(h \circ \alpha)(x) & = & (h(\alpha(\nu_1(x))), \ldots, h(\alpha(\nu_n(x)))) \\ & = & h^n((\alpha(\nu_1(x))), \ldots, \alpha(\nu_n(x))) \\ & = & h^n(\nu^{-1}(\alpha)(x)) \\ & = & (h^n \circ \nu^{-1}(\alpha))(x) \end{array}$$

□

Proposition 18. *For any finite set $V \subseteq \mathcal{V}$, subset R of variable assignments of type $\uplus_{i=1}^n \nu_i(V) \to dom(S)$, and Σ-abstraction $h : S \to \Delta$:*

$$\nu^{-1}(h \circ R) = h^n \circ \nu^{-1}(R)$$

Proof By Lemma 17: $\nu^{-1}(h \circ R) = \{\nu^{-1}(h(\alpha)) \mid \alpha \in R\} = \{h^n(\nu^{-1}(\alpha)) \mid \alpha \in R\} = h^n \circ \nu^{-1}(R)$. □

8. Difference Abstraction

We next recast the notions of difference abstractions from [23, 17, 7] by applying our notion of Σ-abstractions to concrete difference in the Σ-algebra $\mathbb{R}^2_+$, where $\Sigma = F^{(2)}_{pos\text{-}arith} \uplus C_{pos\text{-}arith}$.

More generally, let S be a Σ-algebra, such as the algebra $\mathbb{R}^2_+$ of concrete differences, 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:

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

Definition 19. *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 [23] was similar, but did not make the respective roles of *diff* and $h_{\Delta_6} : \mathbb{R}^2_+ \to \Delta_6$ explicit. By having done so, we can now state 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 20. *For any formula $\phi \in \mathcal{F}_\Sigma$ and Σ-algebra S: $\mathit{diff}(sol^S(\phi)) = sol^{S^2}(\phi)$.*

Proof For all $\alpha : \mathcal{V} \to dom(S), \alpha' : \mathcal{V} \to dom(S), \alpha, \alpha' \in sol^S(\phi)$ we can construct the variable assignment $\alpha'' : \mathcal{V} \to dom(S)^2$ with $\mathit{diff}(\alpha, \alpha') = (\alpha(x), \alpha'(x)) = \alpha''(x)$. So we have $\mathit{diff}(sol^S(\phi)) \subseteq sol^{S^2}(\phi)$.

Conversely, for all variable assigment $\alpha'' : \mathcal{V} \to dom(S)^2$, $\alpha'' \in sol^{S^2}(\phi)$ we can generate two variable assignments $\alpha : \mathcal{V} \to dom(S), \alpha' : \mathcal{V} \to dom(S) \in sol^S(\phi)$ with $\forall x \in \mathcal{V}, \alpha(x) = \pi_1(\alpha''(x)) \wedge \alpha'(x) = \pi_2(\alpha''(x))$. So we have $sol^{S^2}(\phi) \subseteq \mathit{diff}(sol^S(\phi))$, and thus finally $\mathit{diff}(sol^S(\phi)) = sol^{S^2}(\phi)$. □

As an immediate consequence, we have for any Σ-abstraction $h : S^2 \to \Delta$ that $sol(\phi)^\Delta = h \circ sol^{S^2}(\phi)$. Our next objective is to show that we can overapproximate the set $sol(\phi)^\Delta$ by $sol^\Delta(\phi)$ (Corollary 24).

Lemma 21. *Let $h : S' \to \Delta$ be a Σ-abstraction and $\alpha : V \to dom(S')$ and a variable assignment. For any expression $e \in \mathcal{E}_\Sigma$ with $V(e) \subseteq V$: $h([\![e]\!]^{S',\alpha}) \subseteq [\![e]\!]^{\Delta, h\circ\alpha}$.*

Proof sketch. Straightforward by induction on the structure of expressions $e \in \mathcal{E}_\Sigma$. □

Proposition 22. *Let $h : S' \to \Delta$ be a Σ-abstraction and $\alpha : V \to dom(S')$ and a variable assignment. For any positive formula $\phi \in \mathcal{F}_\Sigma$ with $V(\phi) \subseteq V$: $[\![\phi]\!]^{S',\alpha} \leq [\![\phi]\!]^{\Delta, h\circ\alpha}$.*

Proof The proof is by induction on the structure positive Σ-formulas ϕ. If ϕ is some equation $e \doteq e'$ then it holds by Lemma 21 that: $h([\![e]\!]^{S',\alpha}) \subseteq [\![e]\!]^{\Delta, h\circ\alpha}$ and $h([\![e']\!]^{S',\alpha}) \subseteq [\![e']\!]^{\Delta, h\circ\alpha}$. Hence:

$$\begin{array}{lll}
[\![e \doteq e']\!]^{S',\alpha} = 1 & \Leftrightarrow & [\![e]\!]^{S',\alpha} \cap [\![e']\!]^{S',\alpha} \neq \emptyset \\
& \Leftrightarrow & h([\![e]\!]^{S',\alpha}) \cap h([\![e']\!]^{S',\alpha}) \neq \emptyset \\
& \Rightarrow & [\![e]\!]^{\Delta, h\circ\alpha} \cap [\![e']\!]^{\Delta, h\circ\alpha} \neq \emptyset \qquad \text{Lemma 21} \\
& \Leftrightarrow & [\![e \doteq e']\!]^{\Delta, h\circ\alpha} = 1
\end{array}$$

This shows that $[\![e \doteq e']\!]^{S',\alpha} \leq [\![e \doteq e']\!]^{\Delta,h\circ\alpha}$ as required. We next consider the case where ϕ is a conjunction of the form $\phi' \wedge \phi''$.

$$\begin{array}{rcll} [\![\phi' \wedge \phi'']\!]^{S',\alpha} & = & [\![\phi']\!]^{S',\alpha} \wedge^{\mathbb{B}} [\![\phi'']\!]^{S',\alpha} & \\ & \leq & [\![\phi']\!]^{\Delta,h\circ\alpha} \wedge^{\mathbb{B}} [\![\phi'']\!]^{\Delta,h\circ\alpha} & \text{induction hypothesis} \\ & = & [\![\phi' \wedge \phi'']\!]^{\Delta,h\circ\alpha} & \end{array}$$

The last case is where ϕ is an existentially quantified formula of the form $\exists x.\phi'$.

$$\begin{array}{rcll} [\![\exists x.\phi']\!]^{S',\alpha} = 1 & \Leftrightarrow & (\text{exists } s \in dom(S').\ [\![\phi']\!]^{\alpha[x/s],S'}) = 1 & \\ & \Rightarrow & (\text{exists } s \in dom(S').\ [\![\phi']\!]^{h\circ\alpha[x/s],\Delta}) = 1 & \text{induction hypothesis} \\ & \Leftrightarrow & [\![\exists x.\phi']\!]^{\Delta,h\circ\alpha} = 1 & \end{array}$$

This shows that $[\![\exists x.\phi']\!]^{S',\alpha} \leq [\![\exists x.\phi']\!]^{\Delta,h\circ\alpha}$ as required. □

Theorem 23. *Let $h : S' \to \Delta$ be a Σ-abstraction and $\alpha : V \to dom(S')$ and a variable assignment. For any positive formula $\phi \in \mathcal{F}_\Sigma$ with $\mathcal{V}(\phi) \subseteq V$:*

$$h \circ sol^{S'}(\phi) \subseteq sol^{\Delta}(\phi)$$

Proof Let h be Σ-abstraction from S' to Δ and $\phi \in \mathcal{F}_\Sigma$ a positive formula. For any variable assignment α to $dom(S')$, we know that $[\![\phi]\!]^{S',\alpha} \leq [\![\phi]\!]^{\Delta,h\circ\alpha}$ by Proposition 22 since ϕ is positive. This is equivalent to $\{h \circ \alpha \mid \alpha \in sol^{S'}(\phi)\} \subseteq sol^{\Delta}(\phi)$ and thus $h \circ sol^{S'}(\phi) \subseteq sol^{\Delta}(\phi)$ as required. □

Corollary 24 (John's Theorem [23, 17]). *For any Σ-abstraction $h : S^2 \to \Delta$ and positive first-order formula $\phi \in \mathcal{F}_\Sigma$:*

$$sol^{S}(\phi)^{\Delta} \subseteq sol^{\Delta}(\phi)$$

Proof With the Σ-stucture $S' = S^2$, this follows from Lemma 20 and Theorem 23.

$$sol^{S}(\phi)^{\Delta} = h \circ \mathit{diff}(sol^{S}(\phi)) = h \circ sol^{S^2}(\phi) \subseteq sol^{\Delta}(\phi)$$

□

If Δ is finite then the set $sol^{\Delta}(\phi)$ is finite, while $sol^{\mathbb{R}_+}(\phi)$ is usually infinite. If furthermore ϕ is a conjunctive formula, we can compute the set $sol^{\Delta}(\phi)$ by a finite domain constraint solver from ϕ and the tables of Δ (such as e.g. Minizinc [22]). 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 homogeneous linear equations and $S = \mathbb{R}_+$, so that the infinite set $sol^{S}(\phi)$ can be finitely represented by a triangular matrix.

This is the core of the problem that we will solve in the present paper. Our approach will be to rewrite formulas ϕ to $\mathbb{R}_+$-equivalent formulas that are h-exact in the following sense:

Definition 25. *Let $h : S \to \Delta$ be a Σ-abstraction. We call a Σ-formula ϕ h-*exact *if:*

$$h(sol^{S}(\phi)) = sol^{\Delta}(\phi).$$

For h-exact formulas ϕ, $h(sol^{S}_{V}(\phi))$ can be computed exactly by computing $sol^{\Delta}_{V}(\phi)$ as described above.

9. Exact Boolean Abstraction

We recall a recent result from [3] that permits to characterize the boolean abstraction of the $\mathbb{R}_+$-solution set of a mixed linear and nonlinear systems by some $h_{\mathbb{B}}$-exact and $\mathbb{B}$-equivalent formula, so that the boolean abstraction can be computed exactly by finite domain constraint programming.

The development of this result was motivated by the needs of the present paper, but was presented independently for two reasons. First, these results require considerable effort with complementary techniques based on elementary modes, and second, they are of interest elsewhere, in particular for computing the sign abstraction as needed for the abstract interpretation of programming languages.

A *linear equation* with natural coefficients and no constant term is a formula of $\mathcal{F}_\Sigma$ with the arithmetic signature $\Sigma = F^{(2)}_{pos\text{-}arith} \uplus C_{pos\text{-}arith}$ of the form:

$$n_1x_1 + \ldots n_mx_m \doteq n_{m+1}x_{m+1} + \ldots + n_px_p$$

where $m, p, n_1, \ldots, n_p \in \mathbb{N}$ and $x_1, \ldots, x_p \in \mathcal{V}$. An equation $e \doteq e'$ is *positive* if the right-hand side e' is equal to 0. It is called *quasi-positive* if it is positive or the right-hand side has the form nx for some natural n and some variable x. A *system of equations* is a conjunction of equations.

Intuition modes lmentaires

Proposition 26 (Elementary Modes Theorem 15 of [3]). *For any system of linear equations L of size n with m variables we can compute in time $O(2^m poly(n))$ an $\mathbb{R}_+$-equivalent formula of the form $\exists \boldsymbol{x}.\ L'$ such that L' is a system of quasi-positive linear equations, in which all variables on the left hand-side belong to $\boldsymbol{x}$ and all variables on the right-hand side occur exactly once.*

Proof sketch. An $\mathbb{R}_+$-EFM of ϕ is a variable assignment in $\alpha \in sol^{\mathbb{R}_+}(\phi)$ with a minimal support, i.e. with a minimal number of variables x such that $\alpha(x) \neq 0$. It is also well-know that the set of all $\mathbb{R}_+$-EFM of ϕ can be computed in $O(2^m poly(n))$ from ϕ by using for example the reverse search method for the enumeration of polytope vertices and extreme rays [4]. Furthermore, any solution in $sol^{\mathbb{R}_+}(\phi)$ is equal to a positive linear combination of $\mathbb{R}_+$-EFM of ϕ. This can be expressed by an $\mathbb{R}_+$-equivalent formula $\exists \mathbf{x}.\phi'$ such that ϕ' is a system of quasi-positive equations, in which all variables on some left-hand side belong to $\mathbf{x}$ and all variables on some right-hand side occur exactly once. □

Definition 27. *Consider the signature of arithmetics $\Sigma = F^{(2)}_{pos\text{-}arith} \uplus C_{pos\text{-}arith}$. A* product-zero-equation *in $\mathcal{F}_\Sigma$ is a positive polynomial equation of the form $z * z' \doteq 0$ where $z, z' \in \mathcal{V}$. A (simple) $h_{\mathbb{B}}$-*mixed system *is a conjunctive formula in $\mathcal{F}_\Sigma$ of the form $\exists \boldsymbol{z}.\ L \wedge P$ where L is a system of linear equations and P a system of product-zero-equations.*

Note that $h_{\mathbb{B}}$-mixed systems may contain non-positive equations in the linear part L and non-linear equations in the positive part P. We could also consider more general $h_{\mathbb{B}}$-mixed systems in which P may be any system of polynomial equations without constant terms, but this will not be needed in the present paper.

Theorem 28 (Theorem 39 of [3]). *For any $h_{\mathbb{B}}$-mixed system ϕ of size n with m variables we can compute in $O(2^m poly(n))$ time an $h_{\mathbb{B}}$-exact formula ϕ' that is $\mathbb{R}_+$-equivalent to ϕ.*

Proof sketch. Consider a $h_{\mathbb{B}}$-mixed system $\phi = \exists \mathbf{z}.(L \wedge P)$. We can then replace L by the $\mathbb{R}_+$-equivalent formula $\exists \mathbf{x}.L'$ from Proposition 26. Since L' is quasi-positive, the variables on the

right hand-side of some equation in L' occur exactly once, and the variables on the left-hand sides belong to $\mathbf{x}$, and P is restricted, it can be shown with considerable effort, that the formula $\exists \mathbf{z}.((\exists \mathbf{x}.L') \wedge P)$ is indeed $h_{\mathbb{B}}$-exact. So we can chose ϕ' equal to this formula. □

In order to compute the $h_{\mathbb{B}}$-abstraction of a $h_{\mathbb{B}}$-mixed system ϕ exactly, we first compute ϕ' along the lines of the sketch of the proof ideas of Theorem 28 and Proposition 26. We can then compute $sol^{\mathbb{B}}(\phi')$ by finite domain constraint programming.

10. Characterizing Difference Abstractions

We next show how to characterize the difference abstractions to Δ_3 and Δ_6 of the solution set of a linear equation system by the solution set of some first-order formulas interpreted over the finite structure $\mathbb{B}$. We do not know how to find exact equivalent formulas as provided by in Theorem 28 in the case of boolean abstraction. Instead, we will use this theorem to find a finitary characterisation of also in the case of difference abstractions. Hereby, we will strongly rely on properties of definition in the first-order tuple logic, so we introduce the first.

10.1. First-Order Definitions

Our strategy for computing difference abstractions to Δ_3 and Δ_6 will be to decompose those into the $\mathbb{B}$-abstraction and first-order definable functions. Therefore, we define next what it means for function or relation to be defined by a formula of logique des tuples du premier-ordre.

Definition 29. *A $\mathcal{F}^n_\Sigma$-definition of arity m is a function $F : \mathcal{V}^m \to \mathcal{F}^n_\Sigma$ for which there exists a formula $\psi \in \mathcal{F}^n_\Sigma$ and a sequence of distinct variables $\boldsymbol{x} \in \mathcal{V}^m$ such that $\mathcal{V}(\psi) = \{\boldsymbol{x}\}$ and $F(\boldsymbol{y}) = \psi[\boldsymbol{x}/\boldsymbol{y}]$ for all $\boldsymbol{y} \in \mathcal{V}^m$. For any Σ-structure S, F defines the following m-ary relation F^{S^n} on $dom(S)^n$:*

$$F^{S^n} = \{(\alpha(\pi_1(\boldsymbol{x})), \ldots, \alpha(\pi_m(\boldsymbol{x}))) \mid \alpha \in n\text{-}sol^S(\psi)\}$$

The formula $F(\mathbf{y})$ states that the values of $\mathbf{y}$ are in the relation defined by the fomula ψ. Which precise sequence $\mathbf{y}$ of distinct variables is chosen, does not matter since $F(\mathbf{y}) = F(\mathbf{x})[\mathbf{x}/\mathbf{y}]$, since the solutions of $F(\mathbf{y})$ and $F(\mathbf{x})$ over structure S correspond to each other modulo renaming of variable

Lemma 30. *For any first-order definition $F : \mathcal{V}^m \to \mathcal{F}^n_\Sigma$ and sequence $\boldsymbol{y} = y_1 \ldots y_m$ of distinct variables:*

$$n\text{-}sol^S(F(\boldsymbol{y})) = \{[y_1/\boldsymbol{s}_1, \ldots, y_m/\boldsymbol{s}_m] \mid (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m) \in F^{S^n}\}.$$

Proof This is straightforward.

10.2. Defining Function Application

We will frequently have to apply functions to relations defined in first-order tuple logic. Let S be the structure of interest. In the simplest case, we are given a FO-definition $F : \mathcal{V}^2 \to \mathcal{F}^{\mathrm{n}}_\Sigma$ that defines a total function $F^{S^n} : dom(S^n) \to dom(S^n)$, and a first-order definition $G : \mathcal{V}^m \to \mathcal{F}^{\mathrm{n}}_\Sigma$ that defines an m-ary relation G^S on $dom(S)^n$. We can then define the application of F^S to all m components G^S by the first-order definition $F^m(G)$ such that for all $\mathbf{y} = y_1 \ldots y_m$:

$$F^m(G)(\mathbf{y}) \quad =_{\mathrm{def}} \quad \exists \mathbf{z}.\ G(\mathbf{z}) \wedge \bigwedge_{i=1}^{m} F(z_i, y_i)$$

where $\mathbf{z} = z_1 \ldots z_m$ are fresh variables. A more general definition where $F^S : dom(S^n)^k \to dom(S^n)^l$ will be needed later on. It will be given in Section 10.1 together with formal properties of such first-order definitions.

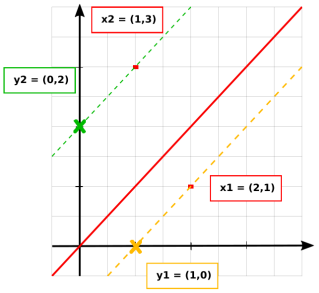


Figure 15: Two examples for the minimal support projection: $y_1 = msp^{\mathbb{R}^2_+}(x_1)$ and $y_2 = msp^{\mathbb{R}^2_+}(x_2)$.

10.3. Exact Δ_3-Abstraction of Linear Systems

We start with the abstraction of Δ_3. We first decompose the abstraction h_{Δ_3} into the boolean abstraction $h_{\mathbb{B}}$ and the *minimal support projection* in $\mathbb{R}^2_+$ defined by the following $h_{\mathbb{B}}$-mixed system in $\mathcal{F}^2_\Sigma$, containing a non-positive linear equation and a product-zero equation, that is non-linear but positive. For any two variables $x, y \in \mathcal{V}$ we define:

$$msp(x,y) \ =_{\text{def}} \ \dot\pi_1(x) + \dot\pi_2(y) \doteq \dot\pi_2(x) + \dot\pi_1(y) \wedge \dot\pi_1(y) * \dot\pi_2(y) \doteq 0$$

The function $msp^{\mathbb{R}^2_+}$ serves for minimal support projection, as illustrated geometrically in Figure 15. The value of $msp^{\mathbb{R}^2_+}(z)$ is the intersection point of the parallel of the diagonal through z with either the x-axis or else the y-axis. For any solution $\alpha \in sol^{\mathbb{R}^2_+}(msp(x,y))$, some component of $\alpha(y)$ must be equal to zero since $\pi_1(\alpha(y)) * \pi_2(\alpha(y)) = 0$. The other component must be equal to $|\pi_1(\alpha(x)) - \pi_2(\alpha(x))|$ since $\pi_1(\alpha(x)) - \pi_2(\alpha(x)) = \pi_1(\alpha(y)) - \pi_2(\alpha(y))$. Hence:

$$\begin{array}{rcl} msp^{\mathbb{R}^2_+} & = & \{((r,r'),(0,r'-r)) \mid r \le r'\} \\ & \cup & \{((r,r'),(r-r',0)) \mid r \ge r'\} \end{array}$$

Lemma 31. *$msp^{\mathbb{R}^2_+}$ is a total function of type $\mathbb{R}^2_+ \to \mathbb{R}^2_+$ satisfying $h_{\Delta_3} = h^2_{\mathbb{B}} \circ msp^{\mathbb{R}^2_+}$.*

Proof By definition $msp^{\mathbb{R}^2_+}$ is a binary relation on $\mathbb{R}^2_+$. This binary relation is a total function satisfying the equation from the lemma due to the equation before the lemma. □

For any first-order definition $G : \mathcal{V}^m \to \mathcal{F}^2_\Sigma$ we defined in Section 10.1 a first-order definition $msp^m(G) : \mathcal{V}^m \to \mathcal{F}^2_\Sigma$ that describes the application of function defined by msp to the m components of the relation defined by G.

Lemma 32. *For any first-order definition $G : \mathcal{V}^m \to \mathcal{F}^2_\Sigma$ and sequence $\boldsymbol{y} \in \mathcal{V}^m$:*

$$msp^{\mathbb{R}^2_+} \circ 2\text{-}sol^{\mathbb{R}_+}(G(\boldsymbol{y})) = 2\text{-}sol^{\mathbb{R}_+}(msp^m(G)(\boldsymbol{y}))$$

Proof This lemma is a consequence of the fact that $msp^{\mathbb{R}^2_+}$ defines a total function by Lemma 31 and a general property of first-order definitions that will be state in Proposition 47 of Section 14. We could have given it directly after the section on the first-order tuple logic, but prefered to do it only at the end, when the full generality of such result needed in this paper has become clear. As parameters for the application of Proposition 47 we choose $F = msp : \mathcal{V}^2 \to \mathcal{F}^2_\Sigma$, $\ell = 1$, $k = 1$, $n = 2$. □

We fix two fresh variable generators $\nu_1, \nu_2 : \mathcal{V} \to \mathcal{V}$ and define $\nu(x)$ and $\nu^{-1}(x)$ as before.

Theorem 33. *For any any linear formula $L(\boldsymbol{y}) \in \mathcal{F}_\Sigma$ with m free variables $\{\boldsymbol{y}\}$ and size n we can compute in time in time $O(2^{2m}poly(n))$ a positive conjunctive formula with existential quantifiers $\phi(\nu(\boldsymbol{y})) \in \mathcal{F}_\Sigma$ with free variables $\{\nu(\boldsymbol{y})\}$ and of size $O(n + m2^{2m})$ such that:*

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

Proof Let $L : \mathcal{V}^m \to \mathcal{F}_\Sigma$ be the first-order definition which when applied to $\mathbf{y}$ returns the linear formula $L(\mathbf{y})$.

	$h_{\Delta_3} \circ \mathit{diff}(sol^{\mathbb{R}_+}(L(\mathbf{y})))$
Proposition 20	$= h_{\Delta_3} \circ sol^{\mathbb{R}_+^2}(L(\mathbf{y}))$
Pair FO Proposition 12	$= h_{\Delta_3} \circ 2\text{-}sol^{\mathbb{R}_+}(L_2(\mathbf{y}))$ *with* $L_2(\mathbf{y}) = \langle L(\mathbf{y})\rangle^2$
Decomposition Lemma 31	$= h^2_{\mathbb{B}} \circ msp^{\mathbb{R}_+^2} \circ 2\text{-}sol^{\mathbb{R}_+}(L_2(\mathbf{y})))$
FO-Definition Lemma 32	$= h^2_{\mathbb{B}} \circ 2\text{-}sol^{\mathbb{R}_+}(msp^m(L_2)(\mathbf{y}))$
Proposition 15	$= h^2_{\mathbb{B}} \circ \nu^{-1}(sol^{\mathbb{R}_+}(\tilde{\nu}(msp^m(L_2)(\mathbf{y}))))$
Proposition 18	$= \nu^{-1}(h_{\mathbb{B}} \circ sol^{\mathbb{R}_+}(\tilde{\nu}(msp^m(L_2)(\mathbf{y}))))$
Definition of $msp^m(L_2(\boldsymbol{y}))$	$= \nu^{-1}(h_{\mathbb{B}} \circ sol^{\mathbb{R}_+}(\tilde{\nu}(\exists \mathbf{z}.\ L_2(\mathbf{z}) \wedge \bigwedge_{i=1}^m msp(z_i, y_i))))$
	where $\mathbf{z} = z_1 \ldots z_m$ *fresh*
$h_{\mathbb{B}}$*-Mixted systems Theorem 28*	$= \nu^{-1}(sol^{\mathbb{B}}(\phi(\nu(\mathbf{y}))))$ *where* $\phi(\nu(\mathbf{y}))$ *is a conjunctive formula that is* $h_{\mathbb{B}}$*-exact and* $\mathbb{R}_+$*-equivalent to the* $h_{\mathbb{B}}$*-mixed system* $\tilde{\nu}(\exists \mathbf{z}.\ L_2(\mathbf{z}) \wedge \bigwedge_{i=1}^m msp(z_i, y_i))$

□

By the $h_{\mathbb{B}}$-mixed systems Theorem 28, the size of $\phi(\nu(\mathbf{y}))$ is in $O(n + m2^{2m})$ and the time of its computation in $O(2^{2m}poly(n))$. □

This theorem induces a new algorithm for the exact computation of $h_{\Delta_3} \circ \mathit{diff}(sol^{\mathbb{R}_+}(L(\mathbf{y})))$ in time $O(poly(n)2^{8m})$ where n is the size of $L(\mathbf{y})$ and m the number of variables in $\mathbf{y}$. Note that this upper bound is simply exponential in the worst case, such as the alternative algorithm sketched in Section 2.

The new algorithm applies Theorem 37 in order to create the formula $\phi(\nu(\mathbf{y}))$ in time $O(2^{2m}poly(n))$. This formula has size $O(n + m2^{2m})$ and $2m$ variables $\nu(\mathbf{y})$. The set of solutions $sol^{\mathbb{B}}(\phi(\nu(\mu(\mathbf{y}))))$ can then be computed by a nave generate and test algorithm in time $O(2^{2m}(n + m2^{2m}))$: Given that this set is of cardinality at most 2^{2m}, we can compute $h_{\Delta_6} \circ \mathit{diff}(sol^{\mathbb{R}_+}(L(\mathbf{y})))$ from $sol^{\mathbb{B}}(\phi(\nu(\mathbf{y})))$ in time 2^{2m} by using the equality of the theorem. The overall time for computing $h_{\Delta_3} \circ \mathit{diff}(sol^{\mathbb{R}_+}(L(\mathbf{y})))$ is in $O(poly(n)2^{2m} + m2^{4m})$ and thus in $O(poly(n)2^{4m})$. In practice, we can improve this algorithm by computing the set boolean solutions of $\phi(\nu(\mathbf{y}))$ by finite domain constraint programming, rather than by a nave generate and test algorithm.

10.4. Exact Δ_6-Abstraction of Linear Systems

The case of Δ_6 following the same approach that for Δ_3, but is considerably more evolved in the usage of first-order definitions.

We consider the abstraction h_{Δ_6} as an element of the Σ-algebra of total functions of type $\mathbb{R}_+^2 \to \mathbb{R}_+^2$, where $\Sigma = F^{(2)}_{pos\text{-}arith} \uplus C_{pos\text{-}arith}$. For any two functions $f, f' : \mathbb{R}_+^2 \to \mathbb{R}_+^2$, the addition is defined by $f +^{\mathbb{R}_+^2 \to \mathbb{R}_+^2} f'(p) = f(p) +^{\mathbb{R}_+^2} f'(p)$ for every $p \in \mathbb{R}_+^2$, and similarly the multiplication is by $f *^{\mathbb{R}_+^2 \to \mathbb{R}_+^2} f'(p) = f(p) *^{\mathbb{R}_+^2} f'(p)$. 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 34. $h_{\Delta_6} = h^2_{\mathbb{B}} + h_{\Delta_3}$ *where* $+ = +^{\mathbb{R}_+^2 \to \mathbb{R}_+^2}$

Proof Let $p = (r, r') \in \mathbb{R}_+^2$. We distinguish the cases for all possible values for $h_{\Delta_6}(p) \in \Delta_6$.

Case $h_{\Delta_6}(p) = \uparrow$**.** Then $0 < r < r'$ so that $h^2_{\mathbb{B}}(p) = (1,1)$ and $h_{\Delta_3}(p) = \triangle = (0,1)$. It follows that $(h^2_{\mathbb{B}} + h_{\Delta_3})(p) = (1,2) = \uparrow = h_{\Delta_6}(p)$.

Case $h_{\Delta_6}(p) = \Uparrow$**.** Then $0 = r < r'$ so that $h^2_{\mathbb{B}}(p) = (0,1)$ and $h_{\Delta_3}(p) = \triangle = (0,1)$. It follows that $(h^2_{\mathbb{B}} + h_{\Delta_3})(p) = (0,2) = \Uparrow = h_{\Delta_6}(p)$.

Case $h_{\Delta_6}(p) = \downarrow$**.** Then $0 < r' < r$ so that $h^2_{\mathbb{B}}(p) = (1,1)$ and $h_{\Delta_3}(p) = \triangledown = (1,0)$. It follows that $(h^2_{\mathbb{B}} + h_{\Delta_3})(p) = (2,1) = \downarrow = h_{\Delta_6}(p)$.

Case $h_{\Delta_6}(p) = \Downarrow$**.** Then $0 = r' < r$ so that $h^2_{\mathbb{B}}(p) = (1,0)$ and $h_{\Delta_3}(p) = \triangledown = (1,0)$. It follows that $(h^2_{\mathbb{B}} + h_{\Delta_3})(p) = (2,0) = \Downarrow = h_{\Delta_6}(p)$.

Case $h_{\Delta_6}(p) = \sim$**.** Then $0 < r = r'$ so that $h^2_{\mathbb{B}}(p) = (1,1)$ and $h_{\Delta_3}(p) = \approx = (0,0)$. It follows that $(h^2_{\mathbb{B}} + h_{\Delta_3})(p) = (1,1) = \sim = h_{\Delta_6}(p)$.

Case $h_{\Delta_6}(p) = \approx$**.** Then $0 = r = r'$ so that $h^2_{\mathbb{B}}(p) = (0,0)$ and $h_{\Delta_3}(p) = \approx = (0,0)$. It follows that $(h^2_{\mathbb{B}} + h_{\Delta_3})(p) = (0,0) = \approx = h_{\Delta_6}(p)$.

□

Let $\mathit{id\text{-}msp}^{\mathbb{R}^2_+} : \mathbb{R}^2_+ \to (\mathbb{R}^2_+)^2$ such that for any $p \in \mathbb{R}^2_+$ $\mathit{id\text{-}msp}^{\mathbb{R}^2_+}(p) = (p, \mathit{msp}^{\mathbb{R}^2_+}(p))$. 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 35 Decomposition. $h_{\Delta_6} = +^{\mathbb{R}^2_+} \bullet (h^2_{\mathbb{B}})^2 \circ \mathit{id\text{-}msp}^{\mathbb{R}^2_+}$.

Proof For any $p \in \mathbb{R}^2_+$, we have:

$$\begin{array}{rcl} h_{\Delta_6}(p) & = & h^2_{\mathbb{B}}(p) +^{\mathbb{R}^2_+} h^2_{\mathbb{B}}(\mathit{msp}^{\mathbb{R}^2_+}(p)) \\ & = & +^{\mathbb{R}^2_+}(h^2_{\mathbb{B}}(p), h^2_{\mathbb{B}}(\mathit{msp}^{\mathbb{R}^2_+}(p))) \\ & = & +^{\mathbb{R}^2_+}((h^2_{\mathbb{B}})^2(p, \mathit{msp}^{\mathbb{R}^2_+}(p))) \\ & = & +^{\mathbb{R}^2_+}((h^2_{\mathbb{B}})^2(\mathit{id}(p), \mathit{msp}^{\mathbb{R}^2_+}(p))) \\ & = & +^{\mathbb{R}^2_+} \bullet (h^2_{\mathbb{B}})^2 \circ \mathit{id\text{-}msp}^{\mathbb{R}^2_+}(p) \end{array}$$

□

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

$$\mathit{id\text{-}msp}(x, y_1, y_2) =_{\text{def}} \langle x = y_1 \rangle^2 \wedge \mathit{msp}(x, y_2)$$

We next define applications of function defined by *id-msp* in the first-order logic. In order to deal with the two output arguments, we use two generators of fresh variables $\mu_1, \mu_2 : \mathcal{V} \to \mathcal{V}$. For any first-order definition $G : \mathcal{V}^m \to \mathcal{F}^2_\Sigma$ we define a first-order definition $\mathit{id\text{-}msp}_\mu{}^m(G) : \mathcal{V}^{2m} \to \mathcal{F}^2_\Sigma$, such that for any sequence of variables $\mathbf{y} \in \mathcal{V}^m$ and with $\mu(\mathbf{y}) = \mu_1(\mathbf{y})\mu_2(\mathbf{y})$:

$$\mathit{id\text{-}msp}_\mu{}^m(G)(\mu(\mathbf{y}))) =_{\text{def}} \exists \mathbf{y}.\ G(\mathbf{y}) \wedge \bigwedge_{i=1}^{m} \mathit{id\text{-}msp}(y_i, \mu_1(y_i), \mu_2(y_i))$$

Lemma 36. $\mathit{id\text{-}msp}^{\mathbb{R}^2_+} \circ 2\text{-}\mathit{sol}^{\mathbb{R}_+}(G(\boldsymbol{y})) = \left\{ \begin{array}{l} [\boldsymbol{y}/(\alpha(\mu_1(\boldsymbol{y})), \alpha(\mu_2(\boldsymbol{y}))] \mid \\ \quad \alpha \in 2\text{-}\mathit{sol}^{\mathbb{R}_+}(\mathit{id\text{-}msp}_\mu{}^m(G)(\mu(\boldsymbol{y})))) \end{array} \right\}$.

Proof This lemma is consequence of the property of first-order definition in the FO-tuple logic that we will state in Proposition 49 of Section 14. Here we choose as parameters the first-order definition $F = \mathit{id\text{-}msp}_\mu : \mathcal{V} \times V^2 \to \mathcal{F}^2_\Sigma$, and $\ell = 1$, $k = 2$, $n = 2$. □

We continue with $\mu_1, \mu_2, \nu_1, \nu_2$ four generators of fresh variables from which we define μ and ν as before.

Theorem 37. *For any linear formula $L(\boldsymbol{y})$ with m free distinct variable $\boldsymbol{y}$ and size n we can compute in time $O(2^{4m}poly(n))$ a positive conjunctive formula with existential quantifiers $\phi(\nu(\mu(\boldsymbol{y}))) \in \mathcal{F}_\Sigma$ with free variables in $\nu(\mu(\boldsymbol{y}))$ and of size $O(n + m2^{4m})$ such that:*

$$h_{\Delta_6} \circ \mathit{diff}(\mathit{sol}^{\mathbb{R}_+}(L(\boldsymbol{y}))) = \{[y/(\beta^2(\nu(\mu_1(y))) +^{\mathbb{R}^2_+} \beta^2(\nu(\mu_2(y))) \mid y \in \{\boldsymbol{y}\}] \mid \beta \in \mathit{sol}^{\mathbb{B}}(\phi(\nu(\mu(\boldsymbol{y}))))\}$$

Proof Let $L : \mathcal{V}^m \to \mathcal{F}_\Sigma$ be the first-order definition which when applied to $\mathbf{y}$ returns the linear formula $L(\mathbf{y})$.

	$h_{\Delta_6} \circ \mathit{diff}(\mathit{sol}^{\mathbb{R}_+}(L(\mathbf{y})))$
Proposition 20	$= h_{\Delta_6} \circ \mathit{sol}^{\mathbb{R}^2_+}(L(\mathbf{y}))$
Proposition 12	$= h_{\Delta_6} \circ 2\text{-}\mathit{sol}^{\mathbb{R}_+}(L_2(\mathbf{y}))$ *with* $L_2(\mathbf{y}) = \langle L(\mathbf{y})\rangle^2$
Dec. Lemma 35	$= +^{\mathbb{R}^2_+} \bullet (h^2_{\mathbb{B}})^2 \circ \mathit{id\text{-}msp}^{\mathbb{R}^2_+} \circ 2\text{-}\mathit{sol}^{\mathbb{R}_+}(L_2(\mathbf{y}))$
FO Lemma 36	$= +^{\mathbb{R}^2_+} \bullet (h^2_{\mathbb{B}})^2 \circ \{[\mathbf{y}/(\alpha(\mu_1(\mathbf{y})), \alpha(\mu_2(\mathbf{y}))] \mid \alpha \in 2\text{-}\mathit{sol}^{\mathbb{R}_+}(\mathit{id\text{-}msp}_\mu{}^m(L_2)(\mu(\mathbf{y})))\}$
	$= +^{h_{\mathbb{R}^2_+}} \bullet \{[\mathbf{y}/(\beta(\mu_1(\mathbf{y})), \beta(\mu_2(\mathbf{y}))] \mid \beta \in h^2_{\mathbb{B}} \circ 2\text{-}\mathit{sol}^{\mathbb{R}_+}(\mathit{id\text{-}msp}_\mu{}^m(L_2)(\mu(\mathbf{y})))\}$
	$= \{[\mathbf{y}/(\beta(\mu_1(\mathbf{y})) +^{\mathbb{R}^2_+} \beta(\mu_2(\mathbf{y}))] \mid \beta \in h^2_{\mathbb{B}} \circ 2\text{-}\mathit{sol}^{\mathbb{R}_+}(\mathit{id\text{-}msp}_\mu{}^m(L_2)(\mu(\mathbf{y})))\}$

We can compute the $h^2_{\mathbb{B}}$ abstraction of the above solution set similarly to the case of Δ_3.

	$h^2_{\mathbb{B}} \circ 2\text{-}\mathit{sol}^{\mathbb{R}_+}(\mathit{id\text{-}msp}_\mu{}^m(L_2)(\mu(\mathbf{y})))$
Proposition 15	$= h^2_{\mathbb{B}} \circ \nu^{-1}(\mathit{sol}^{\mathbb{R}_+}(\tilde{\nu}(\mathit{id\text{-}msp}_\mu{}^m(L_2)(\mu(\mathbf{y})))))$
Proposition 18	$= \nu^{-1}(h_{\mathbb{B}} \circ \mathit{sol}^{\mathbb{R}_+}(\tilde{\nu}(\mathit{id\text{-}msp}_\mu{}^m(L_2)(\mu(\mathbf{y})))))$
Def. of id-msp$_\mu{}^m$	$= \nu^{-1}(h_{\mathbb{B}} \circ \mathit{sol}^{\mathbb{R}_+}(\tilde{\nu}(\exists \mathbf{y}.\ L_2(\mathbf{y}) \wedge \bigwedge_{i=1}^m \mathit{id\text{-}msp}(y_i, \mu_1(y_i), \mu_2(y_i)))))$
Theorem 28 on $h_{\mathbb{B}}$-mixed systems	$= \nu^{-1}(\mathit{sol}^{\mathbb{B}}(\phi(\nu(\mu(\mathbf{y})))))$ *where $\phi(\nu(\mu(\mathbf{y})))$ is a conjunctive formula equivalent to the $h_{\mathbb{B}}$-mixed system $\tilde{\nu}(\exists \mathbf{y}.\ L_2(\mathbf{y}) \wedge \bigwedge_{i=1}^m \mathit{id\text{-}msp}(y_i, \mu_1(y_i), \mu_2(y_i))$*

The combination of the above two calculations and the moving of ν^{-1} to the left yields the equation stated in the theorem. By the $h_{\mathbb{B}}$-mixed systems Theorem 28, the size of $\phi(\nu(\mu(\mathbf{y})))$ is in $O(n + m2^{4m})$ and the time of its computation in $O(2^{4m}poly(n))$. □

Note that upper complexity bound of Theorem 37 is slightly different to that of Theorem 33, since we have to create $4m$ variables for Δ_6, in contrast to $2m$ variables for Δ_3. Theorem 37 induces a new algorithm for the exact computation of $h_{\Delta_6} \circ \mathit{diff}(\mathit{sol}^{\mathbb{R}_+}(L(\mathbf{y})))$. It requires time in $O(poly(n)2^{8m})$ where n is the size of $L(\mathbf{y})$. Hence it is simply exponential in the worst case, such as the existing algorithm sketched in Section 2. But now we can use finite domain constraint programming to avoid the naive generate and test approach. This will prove benefical in practice.

11. Exact Computation of Difference Abstraction with Constraints

We now formalize the general problem of difference abstraction with constraints, and show how to solve it for h_{Δ_3} and h_{Δ_6}.

11.1. General Problem

Let $\Sigma = F^{(2)}_{pos\text{-}arith} \uplus C_{pos\text{-}arith}$. The parameter of the problem is a Σ-abstraction $h : \mathbb{R}^2_+ \to \Delta$ into some finite Σ-structure Δ. We recall that $\Sigma[dom(\Delta)]$ is the extension of signature Σ with additional constants from $dom(\Delta)$.

Definition 38. *The algorithmic problem of* difference abstraction with constraints *is parameterized by a Σ-abstraction $h : \mathbb{R}^2_+ \to \Delta$ and has the following three inputs:*

System of linear equations $L \in \mathcal{F}_\Sigma$: *this system is to be interpreted over $\mathbb{R}_+$.*

Constraint $C \in \mathcal{F}_{\Sigma[dom(\Delta)]}$: *a first-order formula which is to be interpreted over Δ.*

Set of observable variables $V \subseteq \mathcal{V}(L) \cup \mathcal{V}(C)$: *a finite subset of the free variables of the linear equation system and the constraint.*

The output is the h-abstraction of differences of $\mathbb{R}_+$-solutions of L, constrained to the Δ-solutions of C, and projected to the observable variables in V. With $V' = \mathcal{V}(L) \cup \mathcal{V}(C)$ this is:

$$\{\beta_{|V} \mid \beta \in h \circ sol^{\mathbb{R}^2_+}_{V'}(L) \cap sol^{\Delta}_{V'}(C)\}$$

In the example of the introductory reaction network in Figure 3, the system of linear equations $L \in \mathcal{F}_\Sigma$ is given in Figure 4. As non nonlinear constraint $C \in \mathcal{F}_{\Sigma[dom(\Delta)]}$ we can choose the kinetic constraints in Figure 4 in conjunction with the overproduction target $v_{out\text{-}B} \doteq \uparrow$. As set observable variables, we may choose whose values represent changes that are controlled externally, which is the inflow of A and the reactions subject to knockout 3 and 4.

$$V = \{v_{in\text{-}A}, v_3, v_4\}$$

In contrast to the system of linear equation system L, the constraint C may contain arbitrary arithmetic formulas including non-linear polynomial equations and universal quantifiers. This is needed to deal with nonlinear kinetic information. This does not make increase the difficulty of the problem to much, since the constraints are to be interpreted over the finite structure Δ, so that the universal quantifiers in C can be replaced by conjunctions, and the existential quantifieres by disjunctions.

The general problem could be simplified, if we could compute an h-exact formula ϕ that is $\mathbb{R}^2_+$-equivalent to L. In this case, it would be sufficient to compute:

$$sol^{\Delta}(\exists V.\ \phi \wedge C)$$

which can be done by finite domain constraint programming. However, the characterizations of the difference abstractions to Δ_3 and Δ_6 in Theorems 33 and 37 do not provide such h-exact formulas, so further efforts are needed to solve the above problem. This is what we will do next for h_{Δ_3} and h_{Δ_6}.

11.2. Mixed Structures

For adding a treatment of kinetic constraints over Δ_n where $n \in \{3, 6\}$, we consider the union $\mathbb{B} \cup \Delta_n$ as a relational structure, unifying the functionalities of both structures $\mathbb{B}$ and Δ_n. For this, we defined the mixed signature by:

$$\Sigma^{mixed}_n = \{+^{\mathbb{B}}, *^{\mathbb{B}}, +^{\Delta_n}, *^{\Delta_n}\} \cup \mathbb{B} \cup \Delta_n$$

Here we reuse the binary functions of $\mathbb{B}$ and Δ_n as the binary function symbols of Σ^{mixed}_n and the values of $\mathbb{B} \cup \Delta_n$ as the constants of Σ^{mixed}_n.

Definition 39. *For any $n \in \{3, 6\}$, the mixed structure $\mathbb{B} \cup \Delta_n$ is the Σ^{mixed}_n-structure with the mixed domain $\mathbb{B} \cup \Delta_n$ in which all symbols of Σ^{mixed}_n are by themselves, but now with respect to the mixed domain.*

11.3. Difference Abstraction with Constraints for Δ_3

Any pair of booleans in $\mathbb{B}^2 \setminus (1,1)$ is an element in Δ_3, and vice versa. For solving the general problem for Δ_3 we need to capture this relationship in the mixed first-order logic over $\mathbb{B} \cup \Delta_3$. For this we consider the partial function $pair_{\Delta_3} \subseteq (\mathbb{B} \cup \Delta_3)^2 \times (\mathbb{B} \cup \Delta_3)$ such that for all $v_1, v_2 \in \mathbb{B} \cup \Delta_3$:

$$pair_{\Delta_3}(v_1, v_2) = \begin{cases} (v_1, v_2) & \text{if } v_1 *^{\mathbb{B}} v_2 = 0 \text{ so that } (v_1, v_2) \in \Delta_3 \\ \text{undefined} & \text{else} \end{cases}$$

The domain of this partial function is $dom(pair_{\Delta_3}) = \mathbb{B}^2 \setminus \{(1,1)\} = \Delta^3$. Any pair from the domain is mapped to itself. We can define the ternary relation $pair_{\Delta_3}$ in the first-order logic of the mixed structure $B \cup \Delta_3$ by the function $Pair_{\Delta_3} : \mathcal{V}^2 \times \mathcal{V} \to \mathcal{F}_{\Sigma_3^{mixed}}$ such that for all variables $y_1, y_2, y \in \mathcal{V}$:

$$\begin{array}{rcl} Pair_{\Delta_3}(y_1, y_2, y) & =_{\text{def}} & (y_1 = 0 \wedge y_2 = 0 \wedge y = \approx) \\ & \vee & (y_1 = 0 \wedge y_2 = 1 \wedge y = \triangle) \\ & \vee & (y_1 = 1 \wedge y_2 = 0 \wedge y = \triangledown) \end{array}$$

According to definition 29, the first-order definition $Pair_{\Delta_3}$ indeed defines the relation $pair_{\Delta_3}$ in the mixed structure, that is $pair_{\Delta_3} = Pair_{\Delta_3}^{\mathbb{B} \cup \Delta_3}$. In order to deal with the two inputs of $pair_{\Delta_3}$, we reconsider two new variable generators $\nu_1, \nu_2 : \mathcal{V} \to \mathcal{V}$. Recall that for any subset $V \subseteq \mathcal{V}$, structure S, and variable assignment $\alpha : \nu_1(V) \cup \nu_2(V) \to dom(S)$ we defined $\nu^{-1}(\alpha) : V \to dom(S)^2$ such that $\nu^{-1}(\alpha)(y) = (\alpha(\nu_1(y)), \alpha(\nu_2(y)))$ for all $y \in V$. Next we define for any first-order definition $G : \mathcal{V}^{2m} \to \mathcal{F}_{\Sigma_3^{mixed}}$ a first-order definition $Pair^m_{\Delta_3}(G) : \mathcal{V}^m \to \mathcal{F}_{\Sigma_3^{mixed}}$ such that $Pair^m_{\Delta_3}(G)(\mathbf{y})$ describes an application of $Pair_{\Delta_3}$ to the all component of solutions of $G(\mathbf{y})$ for all sequence of variables $\mathbf{y} = y_1 \dots y_m$:

$$Pair^m_{\Delta_3}(G)(\mathbf{y}) = \exists \nu(\mathbf{y}).\ G(\nu(\mathbf{y})) \wedge \bigwedge_{i=1}^{m} Pair_{\Delta_3}(\nu_1(y_i), \nu_2(y_i), y_i)$$

Lemma 40. *Let $m \in \mathbb{N}$, $G : \mathcal{V}^{2m} \to \mathcal{F}_{\Sigma_3^{mixed}}$ be a first-order definition, $\boldsymbol{y}$ be a sequence of variables and $\nu(\boldsymbol{y}) = \nu_1(\boldsymbol{y})\nu_2(\boldsymbol{y})$.*

$$\nu^{-1}(sol^{\mathbb{B}}(G(\nu(\boldsymbol{y}))) \cap \{\alpha : \{\boldsymbol{y}\} \to \Delta_3\} = sol^{\mathbb{B} \cup \Delta_3}(Pair^m_{\Delta_3}(G)(\boldsymbol{y}))$$

Proof The function $pair_{\Delta_3}$ maps all pairs of booleans in Δ_3 to themselves, while being undefined for all elements of $(\Delta_3 \cup \mathbb{B})^2 \setminus \Delta_3$. Hence:

$$\nu^{-1}(sol^{\mathbb{B}}(G(\nu(\mathbf{y}))) \cap \{\alpha : \{\mathbf{y}\} \to \Delta_3\} = pair_{\Delta_3} \circ \nu^{-1}(sol^{\mathbb{B} \cup \Delta_3}(G(\nu(\mathbf{y})))))$$

The composition with the partial function $pair_{\Delta_3}$ on the right is defined in Section 4.1. As stated there, the composition $pair_{\Delta_3} \circ \alpha$ is defined only for total functions α such that $ran(\alpha) \subseteq dom(pair_{\Delta_3}) = \Delta_3$. So a total function $pair_{\Delta_3} \circ \alpha$ may belong to the composition on the right only if α satisfies this condition.

Proposition 48 on first-order definitions with $F = Pair_{\Delta_3} : \mathcal{V}^2 \times \mathcal{V} \to \mathcal{F}_{\Sigma_3^{mixed}}$ shows that:

$$Pair^{\mathbb{B} \cup \Delta_3}_{\Delta_3} \circ \nu^{-1}(sol^{\mathbb{B} \cup \Delta_3}(G(\nu(\mathbf{y})))) = sol^{\mathbb{B} \cup \Delta_3}(Pair^m_{\Delta_3}(G)(\mathbf{y}))$$

Here, the parameters are $\ell = 2$, $k = 1$ and $n = 1$. In combination with $pair_{\Delta_3} = Pair^{\mathbb{B} \cup \Delta_3}_{\Delta_3}$, these two equations yield the lemma. □

Since we will work on the structure $\mathbb{B} \cup \Delta_3$, we need to introduce for any $\phi \in \mathcal{F}_\Sigma$ the formulas $\phi^{\mathbb{B}}$ and ϕ^{Δ_3} that impose the use of $*^{\mathbb{B}}$ and $+^{\mathbb{B}}$, and respectively $*^{\Delta_3}$ and $+^{\Delta_3}$ when interpreting ϕ. It follows that $sol^{\mathbb{B} \cup \Delta_3}(\phi^{\mathbb{B}}) = sol^{\mathbb{B}}(\phi)$.

Theorem 41 (Solving Difference Abstraction with Constraints for h_{Δ_3}). *For any linear formula $L(\boldsymbol{y}) \in \mathcal{F}_{\Sigma}$ with free variable set $\{\boldsymbol{y}\}$, and constraint $C(\boldsymbol{y}') \in \mathcal{F}_{\Sigma[dom(\Delta_3)]}$ with free variable set $\{\boldsymbol{y}'\}$, and $V \subseteq \{\boldsymbol{y}\} \cup \{\boldsymbol{y}'\} = V'$ we can compute at most exponential time a formula over the mixed signature $M \in \mathcal{F}_{\Sigma_3^{mixed}}$ such that:*

$$sol^{\mathbb{B}\cup\Delta_3}(M) = \{\beta_{|V} \mid \beta \in h_{\Delta_3} \circ sol^{\mathbb{R}^2_+}_{V'}(L(\boldsymbol{y})) \cap sol^{\Delta_3}_{V'}(C(\boldsymbol{y}'))\}$$

Proof Without loss of generality we can assume that $\mathbf{y} = \mathbf{y}'$. If not this can be obtained adding by for all $z \in V'$ a redundant equation $\nu(z) \doteq \nu(z)$ conjunctively to both $L(\mathbf{y})$ and $C(\mathbf{y}')$. Once this is done we have $V' = \{\mathbf{y}\} = \{\mathbf{y}'\}$.

By Theorem 33 we can compute in at most exponential time a formula $\phi(\nu(\mathbf{y})) \in \mathcal{F}_{\Sigma}$ such that:

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

The formula $M \in \mathcal{F}_{\Sigma_3^{mixed}}$ can then be defined as follows:

$$M =_{\text{def}} \exists V' \setminus V.\ Pair^m_{\Delta_3}(\phi(\nu(\mathbf{y}))^{\mathbb{B}}) \wedge C(\mathbf{y}')^{\Delta_3}$$

Note that M can be computed in linear time from $\phi(\nu(\mathbf{y}))$ and $C(\mathbf{y}')$. We need to show that formula M satisfies the equation from the theorem. First note that:

$$sol^{\mathbb{B}}(\phi(\mathbf{y})) = sol^{\mathbb{B}\cup\Delta_3}(\phi(\mathbf{y})^{\mathbb{B}})$$

Lemma 40 on the first-order definition of $pair_{\Delta_3}$ shows that:

$$\nu^{\text{-}1}(sol^{\mathbb{B}\cup\Delta_3}(\phi(\mathbf{y})^{\mathbb{B}})) \cap \{\alpha : \{\mathbf{y}\} \to \Delta_3\} = sol^{\mathbb{B}\cup\Delta_3}(Pair^m_{\Delta_3}(\phi(\mathbf{y})^{\mathbb{B}})$$

Hence:

$$\nu^{\text{-}1}(sol^{\mathbb{B}\cup\Delta_3}(\phi(\mathbf{y})^{\mathbb{B}})) \cap sol^{\mathbb{B}\cup\Delta_3}(C(\mathbf{y})^{\Delta_3}) = sol^{\mathbb{B}\cup\Delta_3}(Pair^m_{\Delta_3}(\phi(\mathbf{y})^{\mathbb{B}}) \cap sol^{\mathbb{B}\cup\Delta_3}(C(\mathbf{y})^{\Delta_3})$$

With the equations we can now conclude as follows:

$$\begin{array}{ll} & \{\beta_{|V} \mid \beta \in \nu^{\text{-}1}(sol^{\mathbb{B}}(\phi(\mathbf{y}))) \cap sol^{\Delta_3}(C(\mathbf{y}))\} \\ = & \{\beta_{|V} \mid \beta \in \nu^{\text{-}1}(sol^{\mathbb{B}\cup\Delta_3}(\phi(\mathbf{y})^{\mathbb{B}}) \cap sol^{\mathbb{B}\cup\Delta_3}(C(\mathbf{y})^{\Delta_3})\} \\ = & \{\beta_{|V} \mid \beta \in sol^{\mathbb{B}\cup\Delta_3}(Pair^m_{\Delta_3}(\phi(\mathbf{y})^{\mathbb{B}}) \cap sol^{\mathbb{B}\cup\Delta_3}(C(\mathbf{y})^{\Delta_3})\} \\ = & \{\beta' \mid \beta' \in sol^{\mathbb{B}\cup\Delta_3}(\exists V' \setminus V.\ Pair^m_{\Delta_3}(\phi(\mathbf{y})^{\mathbb{B}}) \wedge C(\mathbf{y})^{\Delta_3})\} \\ = & sol^{\mathbb{B}\cup\Delta_3}(M) \end{array}$$

□

The set $sol^{\mathbb{B}\cup\Delta_3}(M)$ can be computed by a finite domain constraint programming, since $\mathbb{B} \cup \Delta_3$ is a finite structure. Therefore Theorem 41 yields an algorithm for solving the general problem of difference abstraction with constraints in the case of Δ_3.

11.4. Difference Abstraction with Constraints for Δ_6

We next consider the partial function $\mathit{pair\text{-}sum}_{\Delta_6} \subseteq (\mathbb{B} \cup \Delta_6)^4 \times (\mathbb{B} \cup \Delta_6)$ that maps 2 pairs of booleans to abstract difference in Δ_6 in the sense that for all $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{B}^2$:

$$\mathit{pair\text{-}sum}_{\Delta_6}(\mathbf{b}_1, \mathbf{b}_2) = \begin{cases} \mathbf{b}_1 +^{\mathbb{R}^2_+} \mathbf{b}_2 & \text{if } \mathbf{b}_1, \mathbf{b}_2 \in \mathbb{B}^2 \text{ and } \mathbf{b}_1 +^{\mathbb{R}^2_+} \mathbf{b}_2 \in \Delta_6 \\ \text{undefined} & \text{else} \end{cases}$$

By using this partial function and Theorem 37 we can rewrite $h_{\Delta_6} \circ \mathit{diff}(\mathit{sol}^{\mathbb{R}_+}(L(\mathbf{y})))$ for any system of linear equation $L(\mathbf{y})$ as follows where μ, ν, ϕ can be chosen as stated by the theorem:

$$\{[y/\mathit{pair\text{-}sum}_{\Delta_6}(\beta^2(\nu(\mu_1(y))), \beta^2(\nu(\mu_2(y)))) \mid y \in \{\mathbf{y}\}] \mid \beta \in \mathit{sol}^{\mathbb{B}}(\phi(\nu(\mu(\mathbf{y}))))\}$$

We next define the relations on pairs $\mathit{pair\text{-}sum}_{\Delta_6}$ in the mixed pair FO-logic $\mathbb{B} \cup \Delta_6$ by the function $\mathit{Pair\text{-}Sum}_{\Delta_6} : \mathcal{V}^4 \times \mathcal{V} \in \mathcal{F}_{\Sigma_6^{mixed}}$ such that for all variables $y_1, y_2, y_3, y_4, y \in \mathcal{V}$:

$$\begin{array}{rl} \mathit{Pair\text{-}Sum}_{\Delta_6}(y_1, y_2, y_3, y_4, y) =_{\text{def}} & (\ y_1 \doteq 0 \wedge y_2 \doteq 0 \wedge y_3 \doteq 0 \wedge y_4 \doteq 0 \wedge y \doteq \approx) \\ & \vee (y_1 \doteq 1 \wedge y_2 \doteq 1 \wedge y_3 \doteq 0 \wedge y_4 \doteq 0 \wedge y \doteq \sim) \\ & \vee (y_1 \doteq 1 \wedge y_2 \doteq 0 \wedge y_3 \doteq 0 \wedge y_4 \doteq 1 \wedge y \doteq \Uparrow) \\ & \vee (y_1 \doteq 1 \wedge y_2 \doteq 1 \wedge y_3 \doteq 0 \wedge y_4 \doteq 1 \wedge y \doteq \uparrow) \\ & \vee (y_1 \doteq 0 \wedge y_2 \doteq 1 \wedge y_3 \doteq 1 \wedge y_4 \doteq 0 \wedge y \doteq \Downarrow) \\ & \vee (y_1 \doteq 1 \wedge y_2 \doteq 1 \wedge y_3 \doteq 1 \wedge y_4 \doteq 0 \wedge y \doteq \downarrow) \end{array}$$

According to Definition 29, the first-order definition $\mathit{Pair\text{-}Sum}_{\Delta_6}$ indeed defines the partial function $\mathit{pair\text{-}sum}_{\Delta_6}$ in the mixed structure, that is

$$\mathit{pair\text{-}sum}_{\Delta_6} = \mathit{Pair\text{-}Sum}_{\Delta_6}^{\mathbb{B} \cup \Delta_6}$$

We continue with four generators of new variables $\mu_1, \mu_2, \nu_1, \nu_2 : \mathcal{V} \to \mathcal{V}$. For any first-order definition $G : \mathcal{V}^{4m} \to \mathcal{F}_{\Sigma_6^{mixed}}$ we next define a first-order definition $\mathit{Pair\text{-}Sum}^m_{\Delta_6}(G) : \mathcal{V}^m \to \mathcal{F}_{\Sigma_6^{mixed}}$. For all sequence of variables $\mathbf{y} = y_1 \dots y_m$, the formula $\mathit{Pair\text{-}Sum}^m_{\Delta_6}(G)(\mathbf{y})$ describes the application of the partial function defined by $\mathit{Pair\text{-}Sum}_{\Delta_6}$ to the pairs of pairs defined by $G(\nu(\mu(\mathbf{y})))$, i.e.:

$$\mathit{Pair\text{-}Sum}^m_{\Delta_6}(G)(\mathbf{y}) = \exists \nu(\mu(\mathbf{y})).\ G(\nu(\mu(\mathbf{y}))) \wedge \bigwedge_{i=1}^{m} \mathit{Pair\text{-}Sum}_{\Delta_6}(\nu(\mu_1(y_i)), \nu(\mu_2(y_i)), y_i)$$

Lemma 42.

$$\mathit{sol}^{\mathbb{B} \cup \Delta_6}(\mathit{Pair\text{-}Sum}^m_{\Delta_6}(G)(\boldsymbol{y})) = \left\{ \begin{array}{c} [y/\mathit{pair\text{-}sum}_{\Delta_6}(\beta^2(\nu(\mu_1(y))), \beta^2(\nu(\mu_2(y))) \mid y \in \{\boldsymbol{y}\}] \\ \mid \beta \in \mathit{sol}^{\mathbb{B} \cup \Delta_6}(G(\nu(\mu(\boldsymbol{y}))))) \end{array} \right\}$$

Proof We use the fact that $\mathit{pair\text{-}sum}_{\Delta_6} = \mathit{Pair\text{-}Sum}_{\Delta_6}^{\mathbb{B} \cup \Delta_6}$ and the general property of first-order definition from Proposition 48 with $F = \mathit{Pair\text{-}Sum}_{\Delta_6}$, $\ell = 4$, $k = 1$ and $n = 1$. As four new variable generators there we use $\{\nu_i \circ \mu_j \mid i, j \in \{1, 2\}\}$.

$$\mathit{sol}^{\mathbb{B} \cup \Delta_6}(\mathit{Pair\text{-}Sum}^m_{\Delta_6}(G)(\mathbf{y})) = \mathit{pair\text{-}sum}_{\Delta_6} \circ \left\{ \begin{array}{l} \alpha : \{\mathbf{y}\} \to \mathbb{B} \cup \Delta_6 \mid \exists \beta \in \mathit{sol}^{\mathbb{B} \cup \Delta_6}(G(\nu(\mu(\mathbf{y})))).\ \\ \forall y \in \{\mathbf{y}\}.\ \alpha(y) = \beta^2(\nu(\mu_1(y))), \beta^2(\nu(\mu_2(y))) \end{array} \right\}$$

Proposition 43. *For any formula $\phi(\nu(\mu(\boldsymbol{y}))) \in \mathcal{F}_\Sigma$ and constraint $C(\boldsymbol{y}) \in \mathcal{F}_{\Sigma[dom(\Delta_6)]}$ with the same free variables $\{\boldsymbol{y}\}$, and any subset $V \subseteq \{\boldsymbol{y}\}$ we can compute in linear time a formula $M \in \mathcal{F}_{\Sigma_6^{mixed}}$ with $fv(M) = V$ such that:*

$$\mathit{sol}^{\mathbb{B} \cup \Delta_6}(M) = \begin{array}{l} \left\{ \alpha_{|V} \mid \begin{array}{l} \alpha = [y/\mathit{pair\text{-}sum}_{\Delta_6}(\beta^2(\nu(\mu_1(y))), \beta^2(\nu(\mu_2(y)))) \mid y \in \{\boldsymbol{y}\}], \\ \beta \in \mathit{sol}^{\mathbb{B}}(\phi(\nu(\mu(\boldsymbol{y}))))\} \end{array} \right\} \\ \cap \{\alpha_{|V} \mid \alpha \in \mathit{sol}^{\Delta_6}(C(\boldsymbol{y}))\} \end{array}$$

.

Proof We can chose $M = \exists\{\mathbf{y}\} \setminus V.\ \mathit{Pair\text{-}Sum}^m_{\Delta_6}(L(\mathbf{y})) \wedge C(\mathbf{y})^{\Delta_6}$.

$$\begin{array}{ll} \left\{\alpha_{|V} \mid \begin{array}{l}\alpha = [y/\mathit{pair\text{-}sum}_{\Delta_6}(\beta^2(\nu(\mu_1(y))), \beta^2(\nu(\mu_2(y)))) \mid y \in \{\mathbf{y}\}], \\ \beta \in \mathit{sol}^{\mathbb{B}}(\phi(\nu(\mu(\mathbf{y}))))\}\end{array}\right\} \cap \{\alpha_{|V} \mid \alpha \in \mathit{sol}^{\Delta_6}(C(\mathbf{y}))\} & \\ = \mathit{sol}^{\mathbb{B}\cup\Delta_6}(\mathit{Pair\text{-}Sum}^m_{\Delta_6}(\phi(\mathbf{y}))\} \cap \{\alpha_{|V} \mid \alpha \in \mathit{sol}^{\mathbb{B}\cup\Delta_6}(C(\mathbf{y})^{\Delta_6}) & \textit{By Lemma 42} \\ = \mathit{sol}^{\mathbb{B}\cup\Delta_6}(M) & \end{array}$$

□

Theorem 44 (Solving Difference Abstraction with Constraints for h_{Δ_6}). *For any linear formula $L(\boldsymbol{y}) \in \mathcal{F}_\Sigma$ with free variable set $\{\boldsymbol{y}\}$, and constraint $C(\boldsymbol{y}') \in \mathcal{F}_{\Sigma[dom(\Delta_6)]}$ with free variable set $\{\boldsymbol{y}'\}$, and $V \subseteq \{\boldsymbol{y}\} \cup \{\boldsymbol{y}'\} = V'$ we can compute in at most exponential time a formula over the mixed signature $M \in \mathcal{F}_{\Sigma^{mixed}_6}$ such that:*

$$\mathit{sol}^{\mathbb{B}\cup\Delta_6}(M) = \{\beta_{|V} \mid \beta \in h_{\Delta_6} \circ \mathit{sol}^{\mathbb{R}^2_+}_{V'}(L(\boldsymbol{y})) \cap \mathit{sol}^{\Delta_6}_{V'}(C(\boldsymbol{y}'))\}$$

Proof As for the difference abstraction with constraints for Δ_3, without loss of generality we can assume that $\mathbf{y} = \mathbf{y}'$. Once this is done we have $V' = \{\mathbf{y}\} = \{\mathbf{y}'\}$. By Theorem 37, we can compute it at most exponential time a formula $\phi(\nu(\mu(\mathbf{y}))) \in \mathcal{F}_\Sigma$ such that:

$$h_{\Delta_6} \circ \mathit{diff}(\mathit{sol}^{\mathbb{R}_+}(L(\mathbf{y}))) = \{[y/(\beta^2(\nu(\mu_1(y))) +^{\mathbb{R}^2_+} \beta^2(\nu(\mu_2(y))) \mid y \in \{\mathbf{y}\}] \mid \beta \in \mathit{sol}^{\mathbb{B}}(\phi(\nu(\mu(\mathbf{y}))))\}$$

With the definition of $\mathit{pair\text{-}sum}_{\Delta_6}$ we obtain:

$$h_{\Delta_6} \circ \mathit{diff}(\mathit{sol}^{\mathbb{R}_+}(L(\mathbf{y}))) = \left\{\alpha_{|V} \mid \begin{array}{l}\alpha = [y/\mathit{pair\text{-}sum}_{\Delta_6}(\beta^2(\nu(\mu_1(y))), \beta^2(\nu(\mu_2(y)))) \mid y \in \{\mathbf{y}\}], \\ \beta \in \mathit{sol}^{\mathbb{B}}(\phi(\nu(\mu(\mathbf{y}))))\}\end{array}\right\}$$

Finally from Proposition 43, we can compute in linear time a formula $M \in \mathcal{F}_{\Sigma^{mixed}_6}$ such that

$$\begin{array}{ll} \mathit{sol}^{\mathbb{B}\cup\Delta_6}(M) = & \left\{\alpha_{|V} \mid \begin{array}{l}\alpha = [y/\mathit{pair\text{-}sum}_{\Delta_6}(\beta^2(\nu(\mu_1(y))), \beta^2(\nu(\mu_2(y)))) \mid y \in \{\mathbf{y}\}], \\ \beta \in \mathit{sol}^{\mathbb{B}}(\phi(\nu(\mu(\mathbf{y}))))\}\end{array}\right\} \\ & \cap\{\alpha_{|V} \mid \alpha \in \mathit{sol}^{\Delta_6}(C(\mathbf{y}))\} \end{array}$$

. □

The set $\mathit{sol}^{\mathbb{B}\cup\Delta_6}(M)$ can be computed by a finite domain constraint programming, since $\mathbb{B}\cup\Delta_6$ is a finite structure. By combining Theorem 37 and Proposition 43 we obtain an algorithm for solving the general problem of Section 11 in the case of Δ_6.

12. Overapproximation Heuristics with Minimal Support Consequences

We propose a new heuristics for approximating the problem of difference abstractions with constraints. Later on, we will see experimentally later on that this heuristics is close to exact in our main application while requiring considerably less computation time.

Let $h : \mathbb{R}^2_+ \to \Delta$ be some Σ-abstraction into a finite Σ-structure Δ. The general idea of the existing heuristics (see e.g. [23]) for approximating the problem of difference abstractions with constraints from Definition 38 is as follows. Given a linear equation system $L \in \mathcal{F}_\Sigma$, a constraint $C \in \mathcal{F}_{\Sigma[dom(\Delta)]}$ and subsets of variable $V \subseteq V' = \mathcal{V}(L) \cup \mathcal{V}(C)$, we first compute some linear equation system $L' \in \mathcal{F}_\Sigma$ that is a logical consequence of L over $\mathbb{R}_+$ with $\mathcal{V}(L') \subseteq V'$, and in a second step the set of abstract solutions:

$$\mathit{sol}^{\Delta}(\exists V.\ (L \wedge L' \wedge C))$$

by finite domain constraint programming. By John's theorem in Corollary 24, this set is an overapproximation of the target of the problem $\{\beta_{|V} \mid \beta \in sol^{\mathbb{R}_+}_{V'}(L)^{\Delta} \cap sol^{\Delta}_{V'}(C)\}$.

The choice of which $\mathbb{R}_+$-consequence L' of L to add to L is critical. Generally, L' is a finite conjunction of linear equations that are $\mathbb{R}_+$-consequences of L. These are all the linear combinations of equations in L. Unfortunately, there are infinitely many such linear combinations, of which L' has to choose some finite subset.

We call an equation $E \in \mathcal{F}_\Sigma$ *linear* if E has the form $n_1x_1 + \ldots n_kx_k \doteq m_1y_1 + \ldots m_ly_l$ for some pairwise distinct variables x_i and y_j, natural numbers $k, l \geq 0$ and nonzero natural numbers $n_i, m_j > 0$. We call E *nontrivial* if not $k = 0$ and $l = 0$. The support of E is the set of its free variables $\mathcal{V}(E)$. We call E *normalized* if there not exists a natural number $p, n'_1, \ldots n'_l, m'_1, \ldots m'_n$ such that $n_i = p *^{\mathbb{N}} n'_i$ and $m_j = p *^{\mathbb{N}} m'_j$ for all i, j.

Definition 45 (Minimal support linear $\mathbb{R}_+$-consequences). *A linear equation $E \in \mathcal{F}_\Sigma$ is a minimal support linear $\mathbb{R}_+$-consequence of a system of linear equations $L \in \mathcal{F}_\Sigma$ if it satisfies the following three conditions:*

- *E is a nontrivial $\mathbb{R}_+$-consequence of L,*
- *not other nontrivial $\mathbb{R}_+$-consequence of L has a smaller support than E, and*
- *E is normalized.*

It is not difficult to see that no two different minimal support linear $\mathbb{R}_+$-consequences of L may have the same support. Therefore, the set of minimal support linear $\mathbb{R}_+$-consequences of L is finite and of cardinality at most $2^{|\mathcal{V}(E)|}$. Given a system of linear equations $L \in \mathcal{F}_\Sigma$ we denote the conjunction of all its minimal support linear $\mathbb{R}_+$-consequences by:

$$L_{msc} \in \mathcal{F}_\Sigma$$

We next show how to compute L_{msc} from L. First, we transform L into an integer matrix A in linear time, such L is equivalent to $A\mathbf{x} \doteq 0$, where $\mathcal{V}(L) = \{\mathbf{x}\}$. The $\mathbb{R}_+$-consequences of L are thus the linear combination of the rows of A. Since we want to combine the rows of A and not its columns, we consider the transposed matrix A^T. Given a sequence $\mathbf{z}$ of with as many fresh variables as A has rows, the linear combinations of the rows of A (the row space) can be identified with the following set of integer solutions:

$$sol^{\mathbb{Z}}(\exists \mathbf{y}.A^T\mathbf{y} \doteq \mathbf{z})$$

Since the row space is the orthogonal complement of the nullspace of of A,

$$sol^{\mathbb{Z}}(A\mathbf{x} \doteq 0)$$

we have that each vector corresponding to a $\mathbb{R}_+$-consequence of L must be orthogonal to every vector in the nullspace, and in particular to any of its bases. Let $A^{\perp}$ be some basis of the nullspace of A, which can be easily computed by using Gauß algorithm. Then the row space is given by:

$$sol^{\mathbb{Z}}(A^{\perp}\mathbf{z} \doteq 0)$$

Since we are interested only in the subset of solutions of the above system which are nonzero, normalized and with a minimal support, the problem of finding them is simply a particular case of the computation of the elementary modes of the orthogonal complement of the nullspace of A, with basis $A^{\perp}$, but such that nonpositive solutions are considered too. The usual software packages for computing elementary modes can then be applied to $A^{\perp}$ to compute such "reversible" elementary modes [27], or equivalently the problem can be reduced to computing the extreme rays of a cone and solved with any library for the analysis of polytopes, such as [14, 4].

13. Implementation and Experimentation

We have implemented the algorithm solving the differences abstraction problem with constraints in the case of difference abstraction h_{Δ_6} and applied them to change prediction in systems biology

13.1. Implementation

First, we implemented the exact rewriting of linear equation systems for the boolean abstraction $h_{\mathbb{B}}$. We implemented the rewriting in Python while using the libcdd library for computing elementary modes [28].

Second, we implemented a solver for first-order constraint in the mixed structure $\mathbb{B} \cup \Delta_6$ by finite domain constraint programming with the Minizinc tool [22].

Third, we implemented the exact computation of difference abstractions for h_{Δ_6}. This was done in Python based on implementation obtained in the first and second step.

Fourth, we use BioComputing's Reaction-Network tool to represent reaction networks with partial kinetic information in XML-format and to infer the linear equation systems and the nonlinear kinetic constraints. The tools does also support John's overapproximation algorithm. We then integrated our exact algorithm for computing difference abstraction with h_{Δ_6} into the tool, so that it can be applied the systems of linear equation and nonlinear constraints obtained from a reaction network.

Fifth, we implemented the minimal support heuristics again in Python. For this we had to compute "reversible" elementary modes. We again used the libccd library for this, by reducing the computation of reversible elementary modes to the computation of irreversible elementary modes. Beside of this we needed to applying Gauß algorithm to compute the orthogonal matrix $A^{\perp}$. This can be done by using a module of a Python library.

Sixth, we integrated the minimal support heuristics into BioComputing's Reaction-Network tool [1]. We also added a support to compare the solutions sets obtained by John's overapproximation, the minimal support heuristics, and the exact algorithm.

The graphical output of reaction networks is done with BioCompting's Network-Graph tool. It also allows to annotate abstract solution or the elementary flux modes to the network graph, and is integrated into the Reaction-Network tool. BioComputing's Network-Graph tool is publicly available, while the other components of BioComputing Reaction-Network tool are not yet made publicly available.

13.2. Application to Change Prediction of Reaction Networks

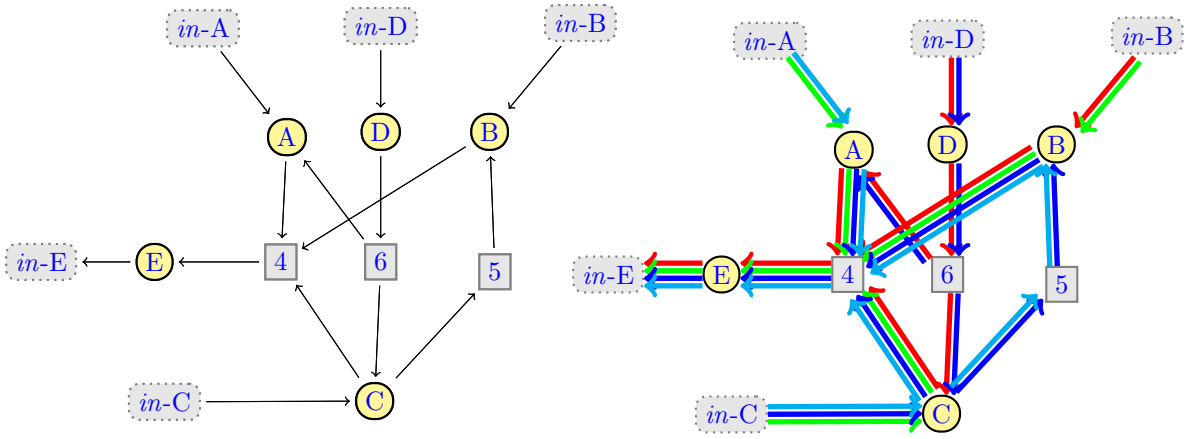
The main application of the change prediction algorithm for reaction networks with partial kinetic information [23, 17] concerns overproduction of the branched chain amino acid Leucine by the reaction network in Figure 20. Leucine is a predecessor of of the surfactin, a nonribosomal peptide, that can be used as a surfactant and produced industrially by the bacteria B. Subtilis. Some of the change predictions obtained for this application where verified successfully in the bioreactor.

We compare the results of John's overapproximation, the minimal support heuristics and the exact algorithm in Figure 16. Beside of the leucine network we also consider the simple loop network and the counter example in Figure 17a.

For the simple loop network, the exact algorithm shows that there are 6 abstract solutions, one for each value of Δ_6. The minimal support heuristic finds the same 6 abstract solutions, while by John's overapproximation returns 19 abstract solutions, of which 13 are not justified.

Network	Count type	John's over-approx.	min. support consequences	exact
Simple loop (Figure 6)	abstract solutions	19	6	6
Leucine overproduction (Figure 20)	knockouts abstract solutions	16 292	14 228	14 228
Counter example (Figure 17a)	abstract solutions	$\geq$ 10000	4454	4374

Figure 16: 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.



(a) Graphical representation.

(b) Elementary flux modes.

Figure 17: Counter example for exactness of the minimal support heuristics for Δ_6.

For the leucine network from Figure 20 the minimal support heuristics finds the same 228 solutions as the exact algorithm. John's overapproximation algorithm produces 292 solutions instead, including the 228 justified solutions.

On the other hand, the minimal support heuristics is remarkably faster than the exact algorithm – in the benchmark on leucine overproduction, we have 5 minutes versus 5 hours.

13.3. Counter Example for the Minimal Support Heuristics

We found and implemented the minimal support heuristics some years before finding the exact algorithm. At that time it was impossible to us to see whether the minimal support heuristics was exact or not. After having developped and implemented the exact algorithm, we could eventually evaluate this question. Our experiments showed that the heuristics is indeed exact for all applications to change prediction of reaction networks in systems biology that we tested. We then tried to prove in the case of Δ_3 that the minimal support heuristics was always exact, but failed to do so.

Next, we tried to find a counter example in the case of Δ_6. For this we developed a random generator of reaction networks and compared the minimal support heuristics with the exact

$2\ v_{in\text{-}E} = v_{in\text{-}B} + v_{in\text{-}C} + v_6$
$2\ v_4 = v_{in\text{-}B} + v_{in\text{-}C} + v_6$
$2\ v_{in\text{-}E} = v_{in\text{-}B} + v_{in\text{-}C} + v_{in\text{-}D}$
$v_{in\text{-}C} + v_6 = v_{in\text{-}B} + 2\ v_5$
$v_{in\text{-}C} + v_{in\text{-}D} = v_{in\text{-}B} + 2\ v_5$
$2\ v_4 = v_{in\text{-}B} + v_{in\text{-}C} + v_{in\text{-}D}$
$v_{in\text{-}E} + v_5 = v_{in\text{-}C} + v_6$
$v_4 + v_5 = v_{in\text{-}C} + v_6$

$v_{in\text{-}E} + v_5 = v_{in\text{-}C} + v_{in\text{-}D}$
$v_4 + v_5 = v_{in\text{-}C} + v_{in\text{-}D}$
$v_{in\text{-}E} = v_{in\text{-}A} + v_{in\text{-}D}$
$v_4 = v_{in\text{-}A} + v_{in\text{-}D}$
$v_{in\text{-}E} = v_{in\text{-}A} + v_6$
$v_4 = v_{in\text{-}A} + v_6$
$v_{in\text{-}B} + v_5 = v_{in\text{-}A} + v_6$
$v_{in\text{-}B} + v_5 = v_{in\text{-}A} + v_{in\text{-}D}$
$v_{in\text{-}B} + v_{in\text{-}C} = 2\ v_{in\text{-}A} + v_6$

$v_{in\text{-}B} + v_{in\text{-}C} = v_{in\text{-}A} + v_{in\text{-}E}$
$v_{in\text{-}B} + v_{in\text{-}C} = 2\ v_{in\text{-}A} + v_{in\text{-}D}$
$v_{in\text{-}B} + v_{in\text{-}C} = v_{in\text{-}A} + v_4$
$v_6 = v_{in\text{-}D}$
$v_{in\text{-}C} = v_{in\text{-}A} + v_5$
$v_4 = v_{in\text{-}E}$
$v_4 = v_{in\text{-}B} + v_5$
$v_{in\text{-}E} = v_{in\text{-}B} + v_5$

Figure 18: Minimal support consequences

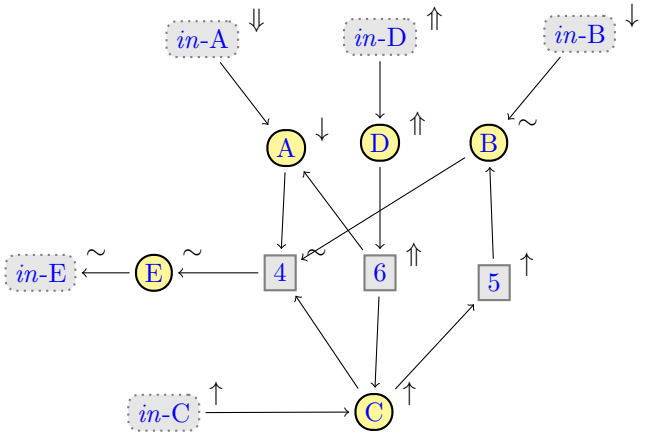


Figure 19: An unjustified solution found with the minimal support heuristics.

algorithm that we implemented for Δ_6 only. This made us indeed find a counter example for Δ_6 that is given in Figure 17a.

Why the minimal support heuristics admits abstract solutions that are not justified is not easy to understand. We can see for instance that $v_4 \doteq v_5 + v_{xB}$. is a minimal support linear consequences, by looking at the elementary flux modes of the counter example network in Figure 17b. The list of all other minimal support linear consequences are given in Figure 18. We can also recognize, which of the abstract solutions are unjustified. An example is given in Figure 19. But we couldn't explain why this solution is unjustified, or find some nonminimal support consequence that it violates.

14. First-Order Function Application

Our next objective is to generalize the definition of function appliction in first-order logic to functions with higher arities, as used already in special cases, and to prove formal properties of such defintions.

We will use vector notation all over. We fix $\ell, k, n \in \mathbb{N}$ and consider first-order definitions $F : \mathcal{V}^\ell \times \mathcal{V}^k \to \mathcal{F}^n_\Sigma$ that define a partial function $F^{S^n} \subseteq dom(S^n)^\ell \times dom(S^n)^k$ for the Σ-structure

S under consideration. For any m, we can lift the first-order definition F to a first-order definition $F^m : \mathcal{V}^{m\ell} \times \mathcal{V}^{mk} \to \mathcal{F}^n_\Sigma$ where F is applied m-times, such that for all sequences $\mathbf{x}^1, \ldots, \mathbf{x}^\ell$, $\mathbf{y}^1, \ldots, \mathbf{y}^k \in \mathcal{V}^m$:

$$F^m(\mathbf{x}^1 \ldots \mathbf{x}^\ell \mathbf{y}^1 \ldots \mathbf{y}^k) \;=_{\text{def}}\; \bigwedge_{i=1}^{m} F(\mathbf{x}^1_i \ldots \mathbf{x}^\ell_i \mathbf{y}^1_i \ldots \mathbf{y}^k_i)$$

For any first-order definition $G : \mathcal{V}^{m\ell} \to \mathcal{F}^n_\Sigma$, we introduce a first-order definition $F^m(G) : \mathcal{V}^{mk} \to \mathcal{F}^n_\Sigma$ such that for all $\overline{\mathbf{y}} \in \mathcal{V}^{mk}$:

$$F^m(G)(\overline{\mathbf{y}}) =_{\text{def}} \exists \overline{\mathbf{x}}.\; G(\overline{\mathbf{x}}) \wedge F^m(\overline{\mathbf{x}}, \overline{\mathbf{y}})$$

where $\overline{\mathbf{x}} = (\mathbf{x}^1 \ldots \mathbf{x}^\ell) \in \mathcal{V}^{m\ell}$ is some sequence of fresh variables. Note that $fv(F^m(G)(\overline{\mathbf{y}})) = \{\overline{\mathbf{y}}\}$ so that the precise choice of $\mathbf{x}$ is irrelevant.

Lemma 46. *Let $F : \mathcal{V}^{\ell+k} \to \mathcal{F}^n_\Sigma$ and $G : \mathcal{V}^{m\ell} \to \mathcal{F}^n_\Sigma$ be first-order definitions and S a Σ-structure S. If the relation $F^{S^n} \subseteq dom(S^n)^{\ell+k}$ is a partial function of type $dom(S^n)^\ell \times dom(S^n)^k$ then the relation $(F^m)^{S^n}$ is a partial function of type $dom(S^n)^{m\ell} \times dom(S^n)^{mk}$ such that:*

$$(F^m)^{S^n}(G^{S^n}) = F^m(G)^{S^n}$$

Proof Let $\overline{\mathbf{x}} = \mathbf{x}^1 \dots \mathbf{x}^\ell \in \mathcal{V}^{m\ell}$ and $\overline{\mathbf{y}} = \mathbf{y}^1 \dots \mathbf{y}^k \in \mathcal{V}^{mk}$ be sequences of variables such that no variables occurs twice in $\overline{\mathbf{x}}\overline{\mathbf{y}}$. Then:

$$\begin{array}{rcl}
(F^m)^{S^n}(G^{S^n}) & = & \{(F^m)^{S^n}(\alpha(\overline{\mathbf{x}})) \mid \alpha \in \textit{n-sol}^S(G(\overline{\mathbf{x}}))\} \\
& = & \{\alpha'(\overline{\mathbf{y}}) \mid \alpha' \in \textit{n-sol}^S(F^m(\overline{\mathbf{x}}\overline{\mathbf{y}})),\ \alpha'_{|\{\overline{\mathbf{x}}\}} \in \textit{n-sol}^S(G(\overline{\mathbf{x}}))\} \\
& = & \{\alpha'(\overline{\mathbf{y}}) \mid \alpha' \in \textit{n-sol}^S(G(\overline{\mathbf{x}}) \wedge F^m(\overline{\mathbf{x}}\overline{\mathbf{y}}))\} \\
& = & \{\alpha''(\overline{\mathbf{y}}) \mid \alpha'' \in \textit{n-sol}^S(\exists \overline{\mathbf{x}}.\ G(\overline{\mathbf{x}}) \wedge F^m(\overline{\mathbf{x}}\overline{\mathbf{y}}))\} \\
& = & F^m(G)^{S^n}
\end{array}$$

For the previous last step note that for any $\alpha' \in \textit{n-sol}^S(G(\overline{\mathbf{x}}) \wedge F^m(\overline{\mathbf{x}}\overline{\mathbf{y}}))$ we can chose α'' as the restriction $\alpha'_{|\mathcal{V}\setminus\{\overline{\mathbf{x}}\}}$. Conversely, for any $\alpha'' \in \textit{n-sol}^S(\exists \overline{\mathbf{x}}.\ G(\overline{\mathbf{x}}) \wedge F^m(\overline{\mathbf{x}}\overline{\mathbf{y}}))$ there must exist a solution $\alpha' \in \textit{n-sol}^S(G(\overline{\mathbf{x}}) \wedge F^m(\overline{\mathbf{x}}\overline{\mathbf{y}}))$ such that α'' is the restriction $\alpha'_{|\mathcal{V}\setminus\{\overline{\mathbf{x}}\}}$. □

For the case $k = 1$ and $\ell = 1$ Lemma 46 yields the following consequence.

Proposition 47 ($\ell = 1$ and $k = 1$). *For any FO definition $G : \mathcal{V}^m \to \mathcal{F}^n_\Sigma$ and $F : \mathcal{V} \times \mathcal{V} \to \mathcal{F}^n_\Sigma$, sequence of variables $\boldsymbol{y} \in \mathcal{V}^m$ and Σ-structure S for which the relation F^{S^n} is a partial function of type $dom(S^n) \times dom(S^n)$:*

$$F^{S^n} \circ \textit{n-sol}^S(G(\boldsymbol{y})) = \textit{n-sol}^S(F^m(G)(\boldsymbol{y}))$$

Proof From Lemmata 30 and 46:

$$\begin{array}{l|l}
F^{S^n} \circ \textit{n-sol}^S(G(\mathbf{y})) & \\
\text{Lemma 30} & = F^{S^n} \circ \{[\mathbf{y}/\mathbf{s}] \mid \mathbf{s} \in G^{S^n}\} \\
& = \{[\mathbf{y}/\mathbf{s}] \mid \mathbf{s} \in (F^m)^{S^n}(G^{S^n})\} \\
\text{Lemma 46} & = \{[\mathbf{y}/\mathbf{s}] \mid \mathbf{s} \in F^m(G)^{S^n}\} \\
\text{Lemma 30} & = \textit{n-sol}^S(F^m(G)(\mathbf{y}))
\end{array}$$

□

For the case of general ℓ and $k = 1$, Proposition 47 can be generalized as follows:

Proposition 48 ($\ell \geq 1$ and $k = 1$). *Let $G : \mathcal{V}^{m\ell} \to \mathcal{F}^n_\Sigma$ and $F : \mathcal{V}^\ell \times \mathcal{V} \to \mathcal{F}^n_\Sigma$ be a first-order definition, and S a Σ-structure such that the relation F^{S^n} is a partial function of type $dom(S^n)^\ell \times dom(S^n)$. Then for any $\boldsymbol{y} \in \mathcal{V}^m$ and fresh variable generators $\nu_1, \dots, \nu_\ell$ the sequence of variables $\nu(\boldsymbol{y}) = \nu_1(\boldsymbol{y}) \dots \nu_\ell(\boldsymbol{y})$ satisfies:*

$$F^{S^n} \circ \nu^{\text{-}1}(\textit{n-sol}^S(G(\nu(\boldsymbol{y})))) = \textit{n-sol}^S(F^m(G)(\boldsymbol{y}))$$

where $\nu^{\text{-}1}(\alpha)(y) = (\alpha(\nu_1(y)), \dots, \alpha(\nu_\ell(y)))$ for all $y \in \mathcal{V}(\boldsymbol{y})$ and $\alpha : \cup_{i=1}^\ell \mathcal{V}(\nu_i(\boldsymbol{y})) \to dom(S^n)$.

Proof Again from Lemmata 30 and 46, by generalizing and adpatation of the proof of Proposition 47:

$$\begin{array}{l|l}
F^{S^n} \circ \nu^{\text{-}1}(\textit{n-sol}^S(G(\nu(\mathbf{y})))) & \\
\text{Lemma 30} & = F^{S^n} \circ \{\nu^{\text{-}1}[\nu(\mathbf{y})/\mathbf{s}] \mid \mathbf{s} \in G^{S^n}\} \\
& = \{[\mathbf{y}/\mathbf{s}] \mid \mathbf{s} \in (F^m)^{S^n}(G^{S^n})\} \\
\text{Lemma 46} & = \{[\mathbf{y}/\mathbf{s}] \mid \mathbf{s} \in F^m(G)^{S^n}\} \\
\text{Lemma 30} & = \textit{n-sol}^S(F^m(G)(\mathbf{y}))
\end{array}$$

□

For general $k \geq 1$ and $l = 1$, Lemma 46 yields the following generalization of Proposition 47:

Proposition 49 ($\ell = 1$ **and** $k \geq 1$)**.** *For any first-order definition* $G : \mathcal{V}^m \to \mathcal{F}^n_\Sigma$ *and* $F : \mathcal{V} \times \mathcal{V}^k \to \mathcal{F}^n_\Sigma$ *and any structure* S *such that the relation* F^{S^n} *is a partial function of type* $dom(S^n) \times dom(S^n)^k$*, and any sequences of fresh variables* $\boldsymbol{y}, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^k \in \mathcal{V}^m$*:*

$$(F^m)^{S^n} \circ n\text{-}sol^S(G(\boldsymbol{y})) = \{[\boldsymbol{y}/(\alpha(\boldsymbol{y}^1), \ldots, \alpha(\boldsymbol{y}^k)] \mid \alpha \in n\text{-}sol^S(F^m(G)(\boldsymbol{y}^1, \ldots, \boldsymbol{y}^k))\}$$

Proof This is another generalization of the proof of Proposition 47:

$$\begin{array}{ll}
 & F^{S^n} \circ n\text{-}sol^S(G(\mathbf{y})) \\
\textit{Lemma 30} & = F^{S^n} \circ \{[\mathbf{y}/\mathbf{s}] \mid \mathbf{s} \in G^{S^n}\} \\
 & = \{[\mathbf{y}/(\mathbf{s}^1, \ldots, \mathbf{s}^k)] \mid (\mathbf{s}^1, \ldots, \mathbf{s}^k) \in (F^m)^{S^n}(G^{S^n})\} \\
\textit{Lemma 46} & = \{[\mathbf{y}/(\mathbf{s}^1, \ldots, \mathbf{s}^k)] \mid (\mathbf{s}^1, \ldots, \mathbf{s}^k) \in F^m(G)^{S^n})\} \\
\textit{Lemma 30} & = \{[\mathbf{y}/(\alpha(\mathbf{y}^1), \ldots, \alpha(\mathbf{y}^k)] \mid \alpha \in n\text{-}sol^S(F^m(G)(\mathbf{y}^1, \ldots, \mathbf{y}^k))\}
\end{array}$$

□

15. Conclusion

We presented a new algorithm for computing the difference abstraction over Δ_3 and Δ_6 of the solution set of a system of linear equation systems with nonlinear constraints on the difference abstractions. The algorithm relies on an exact rewriting of linear equation systems with respect to the boolean abstraction, which can be based on elementary modes. Our reduction uses decompositions of the difference abstractions h_{Δ_3} and h_{Δ_6} into the boolean abstractions and functions on pair algebra $\mathbb{R}^2_+$ that can be defined in first-order logic with pairs. Eventually, we can compute the difference abstractions for systems of linear equation with constraints by finite domain constraint programming. We implemented our algorithm and applied it to change prediction of reaction networks with partial kinetic information in systems biology.

We also presented the minimal support heuristics, for approximating the difference abstraction over Δ_3 and Δ_6 of the solution set of a system of linear equation systems with nonlinear constraints. It turns out that the minimal support heuristics is exact for the prime application of change prediction while requiring much less computation time. It was difficult to find a counter-example shown that the minimal support heuristics is not always exact. We finally succeeded in for the case of Δ_6 by randomly generating and testing reaction networks. In the case of Δ_3 the question remains open though.

We believe that the presented algorithms are fundamental to develop better change prediction methods in the future. For this, it is important to not only deal with cycles in the metabolic parts of reaction networks but also deal with cycles through the regulatory part. An important challeange in practice is to provide multiple changes prediction. The current approaches, however, are not sufficiently precise to do so. This is due to the lack of kinetic information. Furthermore, the current approach can only abstract the differences of steady states, but not account for their relationship to initial states.

In the longer run, it would be of intested to obtain more quantitative predictions and not only qualitative predictions. But this would require more precise kinetic information in the reaction networks and to use more refined difference abstractions for the abstract interpretation.

References

[1] Biocomputing's network-graph tool. http://researchers.lille.inria.fr/~niehren/BioComputing/Network-Graph/doc.html. 2018-07-10.

[2] E. Allart, J. Niehren, and C. Versari. Computing difference abstractions of metabolic networks under kinetic constraints. In L. Bortolussi and G. Sanguinetti, editors, *Computational Methods in Systems Biology - 17th International Conference, CMSB 2019, Trieste, Italy, September 18-20, 2019, Proceedings*, volume 11773 of *Lecture Notes in Computer Science*, pages 266–285. Springer, 2019.

[3] E. Allart, J. Niehren, and C. Versari. Abstracting Linear Equation Systems. Submitted., May 2020.

[4] D. Avis. A revised implementation of the reverse search vertex enumeration algorithm. In *Polytopescombinatorics and computation*, pages 177–198. Springer, 2000.

[5] 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.

[6] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *POPL*, pages 269–282, 1979.

[7] 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.

[8] 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.

[9] G. Facchetti and C. Altafini. Partial inhibition and bilevel optimization in flux balance analysis. *BMC Bioinformatics*, pages 344–344, 2013.

[10] F. Fages, S. Gay, and S. Soliman. Inferring reaction systems from ordinary differential equations. *Theor. Comput. Sci.*, 599:64–78, 2015.

[11] F. Fages and S. Soliman. Abstract interpretation and types for systems biology. *Theor. Comput. Sci.*, 403(1):52–70, 2008.

[12] M. Feinberg. Chemical reaction network structure and the stability of complex isothermal reactors. *Chemical Engineering Science*, 42(10):2229 – 2268, 1987.

[13] K. D. Forbus. Qualitative reasoning. In A. B. Tucker, editor, *The Computer Science and Engineering Handbook*, pages 715–733. CRC Press, 1997.

[14] K. Fukuda and A. Prodon. Double description method revisited. In *Franco-Japanese and Franco-Chinese Conference on Combinatorics and Computer Science*, pages 91–111. Springer, 1995.

[15] J. Gagneur and S. Klamt. Computation of elementary modes: a unifying framework and the new binary approach. *BMC bioinformatics*, 5(1):1, 2004.

[16] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. Copasia complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, 2006.

[17] 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.

[18] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Comb.*, 4(4):373–396, 1984.

[19] 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.

[20] C. D. Maranas and A. R. Zomorrodi. *Flux Balance Analysis and LP Problems*, chapter 3, pages 53–80. Wiley-Blackwell, 2016.

[21] K. Marriott, P. J. Stuckey, and M. Wallace. Constraint logic programming. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 409–452. Elsevier, 2006.

[22] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack. Minizinc: Towards a standard CP modelling language. In C. Bessiere, editor, *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*, pages 529–543. Springer, 2007.

[23] 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.

[24] J. D. Orth, I. Thiele, and B. O. Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–248, 2010.

[25] 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.

[26] S. Schuster, T. Dandekar, and D. A. Fell. Detection of elementary flux modes in biochemical networks: a promising tool for pathway analysis and metabolic engineering. *Trends in biotechnology*, 17(2):53–60, 1999.

[27] M. Terzer and J. Stelling. Large-scale computation of elementary flux modes with bit pattern trees. *Bioinformatics*, 24(19):2229–2235, 2008.

[28] M. Troffaes. pycddlib-a python wrapper for komei fukudals cddlib, 2018.

[29] 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.

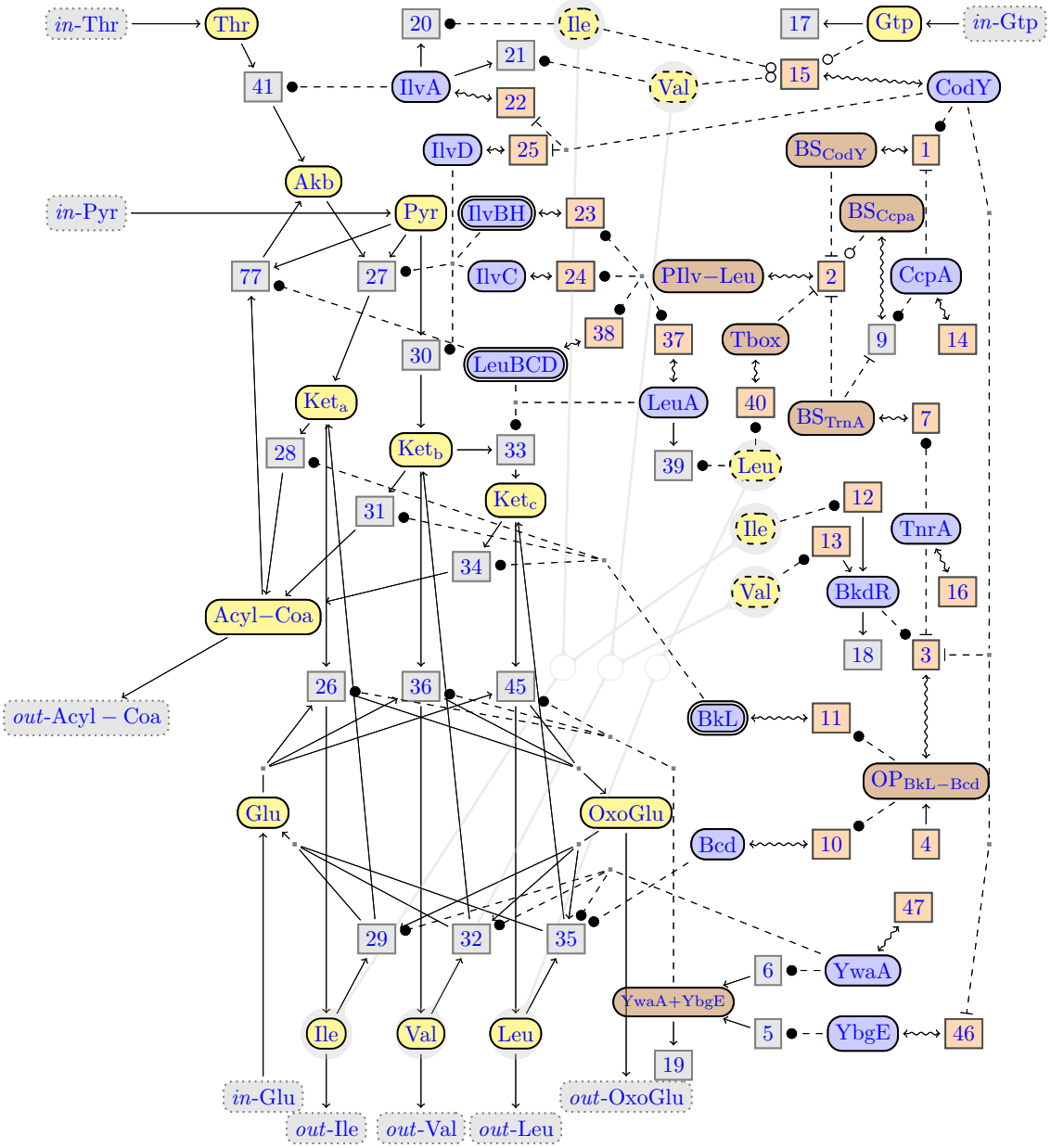


Figure 20: Graphical representation of the model that represents a part of the metabolism of the bacteria B. Subtilis, the branched chain amino-acid: Isoleucine, Leucine and Valine.

Appendix A. Proofs for Section 7 (First-Order Logic)

Lemma 11. *For any $e \in \mathcal{E}_\Sigma$, Σ-algebra S, $n \geq 1$, and $\beta : V \to dom(S)^n$ with $\mathcal{V}(\phi) \subseteq V \subseteq \mathcal{V}$:*

$$[\![e]\!]^{\beta,S^n} = [\![(\Pi_1(e), \ldots, \Pi_n(e))]\!]^{\beta,S}$$

Proof. By induction on the structure of expressions in $\mathcal{E}_\Sigma$.
Cas of constants $c \in C$.

$$[\![c]\!]^{\beta,S^n} = c^{S^n} = (c^S, \ldots, c^S) = [\![(\Pi_1(c), \ldots, \Pi_n(c))]\!]^{\beta,S}$$

Cas of variables $x \in \mathcal{V}$.

$$[\![x]\!]^{\beta,S^n} = \beta(x) = \beta((\pi_1(x), \ldots, \pi_n(x))) = [\![(\Pi_1(x), \ldots, \Pi_n(x))]\!]^{\beta,S}$$

Cas of expressions $e_1 \odot e_2$ where $e_1, e_2 \in \mathcal{E}_\Sigma$ and $\odot \in F^{(2)}$.

$$\begin{array}{ll} [\![e_1 \odot e_2]\!]^{\beta,S^n} & = \cup\{(s_1 \odot^{S^n} s_2) \mid s_j \in [\![e_j]\!]^{\beta,S^n}\} \\ \textit{ind.hyp.} & = \cup\{(s_1 \odot^{S^n} s_2) \mid s_j \in [\![(\Pi_1(e_j), \ldots, \Pi_n(e_j))]\!]^{\beta,S}\} \\ & = [\![(\Pi_1(e_1) \odot^S \Pi_1(e_2), \ldots, \Pi_n(e_1) \odot^S \Pi_n(e_2)]\!]^{\beta,S} \end{array}$$

□

Proposition 12. *For any $\phi \in \mathcal{F}_\Sigma$, Σ-algebra S, and $n \geq 1$: $sol^{S^n}(\phi) = n\text{-}sol^S(\langle\phi\rangle^n)$.*

Proof. By induction on the structure of formulas in $\mathcal{F}_\Sigma$. The base case of Σ-equations follows essentially from Lemma 11. Let β be an assignment variables $\beta : V \to dom(S)^n$ with $V \subseteq \mathcal{V}$.
Cas $e \doteq e'$ where $e, e' \in \mathcal{E}_\Sigma$.

$$\begin{array}{ll} sol^{S^n}(e \doteq e') & = \{\beta \mid [\![e \doteq e']\!]^{\beta,S^n} = 1\} \\ \textit{by prev. claim} & = \{\beta \mid [\![\bigwedge_{i=1}^n \Pi_i(e) \doteq \Pi_i(e')]\!]^{\beta,S} = 1\} \\ & = \{\beta \mid [\![\langle e \doteq e'\rangle^n]\!]^{\beta,S} = 1\} \\ & = n\text{-}sol^S(\langle e \doteq e'\rangle^n) \end{array}$$

Cas $\phi \wedge \phi'$ where $\phi, \phi' \in \mathcal{F}_\Sigma$.

$$\begin{array}{ll} sol^{S^n}(\phi \wedge \phi') & = \{\beta \mid [\![\phi \wedge \phi']\!]^{\beta,S^n} = 1\} \\ & = \{\beta \mid [\![\phi]\!]^{\beta,S^n} \wedge [\![\phi']\!]^{\beta,S^n} = 1\} \\ \textit{ind.hyp.} & = \{\beta \mid [\![\langle\phi\rangle^n]\!]^{\beta,S} \wedge [\![\langle\phi'\rangle^n]\!]^{\beta,S} = 1\} \\ & = \{\beta \mid [\![\langle\phi \wedge \phi'\rangle^n]\!]^{\beta,S} = 1\} \\ & = n\text{-}sol^S(\langle\phi \wedge \phi'\rangle^n) \end{array}$$

Cas $\neg\phi$ were $\phi \in \mathcal{F}_\Sigma$.

$$\begin{array}{ll} sol^{S^n}(\neg\phi) & = \{\beta \mid [\![\neg\phi]\!]^{\beta,S^n} = 1\} \\ & = \{\beta \mid \neg[\![\phi]\!]^{\beta,S^n} = 1\} \\ \textit{ind.hyp.} & = \{\beta \mid \neg[\![\langle\phi\rangle^n]\!]^{\beta,S} = 1\} \\ & = \{\beta \mid [\![\neg\langle\phi\rangle^n]\!]^{\beta,S} = 1\} \\ & = \{\beta \mid [\![\langle\neg\phi\rangle^n]\!]^{\beta,S} = 1\} \\ & = n\text{-}sol^S(\langle\neg\phi\rangle^n) \end{array}$$

Cas $\exists x.\phi$ were $\phi \in \mathcal{F}_\Sigma$.

$$\begin{array}{ll} sol^{S^n}(\exists x.\ \phi) & = \{\beta \mid [\![\exists x.\ \phi]\!]^{\beta,S^n} = 1\} \\ & = \{\beta \mid \mathit{exists}\ s \in \mathit{dom}(S)^n.\ [\![\phi]\!]^{\beta[x/s],S^n} = 1\} \\ \textit{ind.hyp.} & = \{\beta \mid \mathit{exists}\ s \in \mathit{dom}(S)^n.\ [\![\langle\phi\rangle^n]\!]^{\beta[x/s],S} = 1\} \\ & = \{\beta \mid [\![\exists x.\ \langle\phi\rangle^n]\!]^{\beta,S} = 1\} \\ & = \{\beta \mid [\![\langle\exists x.\ \phi\rangle^n]\!]^{\beta,S} = 1\} \\ & = \mathit{n\text{-}sol}^S(\langle\exists x.\ \phi\rangle^n) \end{array}$$

□

Lemma 14. *For any expression $o \in \mathcal{O}^n_\Sigma$ and variable assignment $\beta : V \to \mathit{dom}(S)^n$ with $\mathcal{V}(o) \subseteq V \subseteq \mathcal{V}$ we have $[\![\tilde{\nu}(o)]\!]^{S,\nu(\beta)} = [\![o]\!]^{\beta,S}$.*

Proof. By induction on the structure of Σ-expressions $o \in \mathcal{O}^n_\Sigma$.
Cas of constants $c \in C$.

$$[\![\tilde{\nu}(c)]\!]^{S,\nu(\beta)} = [\![c]\!]^{\beta,S}$$

Cas $\dot{\pi}_i(x)$ where $x \in \mathcal{V}$ and $1 \leq i \leq n$.

$$\begin{array}{ll} [\![\tilde{\nu}(\dot{\pi}_i(x))]\!]^{S,\nu(\beta)} & = \{\nu(\beta)(\nu_i(x))\} \\ & = \{\pi_i(\beta(x))\} \\ & = [\![o]\!]^{\beta,S} \end{array}$$

Cas $o_1 \odot o_2$ where $o_1, o_2 \in \mathcal{O}^n_\Sigma$ and $\odot \in F^{(2)}$.

$$\begin{array}{ll} [\![\tilde{\nu}(o_1 \odot o_2)]\!]^{S,\nu(\beta)} & = [\![\tilde{\nu}(o_1) \odot \tilde{\nu}(o_2)]\!]^{S,\nu(\beta)} \\ & = \cup\{(s_1 \odot^S s_2 \mid s_i \in [\![\tilde{\nu}(o_i)]\!]^{S,\nu(\beta)}\} \\ \textit{ind.hyp.} & = \cup\{(s_1 \odot^S s_2 \mid s_i \in [\![o_i]\!]^{\beta,S}\} \\ & = [\![o_1 \odot o_2]\!]^{\beta,S} \end{array}$$

Cas $\dot{\pi}_i(o_1, \ldots, o_n)$ where $o_1, \ldots, o_n \in \mathcal{O}^n_\Sigma$.

$$\begin{array}{ll} [\![\tilde{\nu}(\dot{\pi}_i(o_1, \ldots o_n))]\!]^{S,\nu(\beta)} & = [\![\tilde{\nu}(o_i)]\!]^{S,\nu(\beta)} \\ \textit{ind.hyp.} & = [\![o_i]\!]^{\beta,S} \\ & = [\![\dot{\pi}_i(o_1, \ldots o_n)]\!]^{\beta,S} \end{array}$$

□

Claim 16. *For any variable assignment $\beta : V \to \mathit{dom}(S)^n$ with $V \subseteq \mathcal{V}$ and formula $\psi \in \mathcal{F}^n_\Sigma$ we have $[\![\tilde{\nu}(\psi)]\!]^{S,\nu(\beta)} = [\![\psi]\!]^{\beta,S}$.*

Proof. The proof of the claim is by induction on the structure of Σ-formulas in $\mathcal{F}^n_\Sigma$.
Cas $o \doteq o'$ where $o, o' \in \mathcal{O}^n_\Sigma$.

$$\begin{array}{ll} [\![o \doteq o']\!]^{\beta,S} = 1 & \Leftrightarrow [\![o]\!]^{\beta,S} \cap [\![o]\!]^{\beta,S} \neq \emptyset \\ \text{Lemma 14} & \Leftrightarrow [\![\tilde{\nu}(o)]\!]^{S,\nu(\beta)} \cap [\![\tilde{\nu}(o)]\!]^{S,\nu(\beta)} \neq \emptyset \\ & \Leftrightarrow [\![\tilde{\nu}(o) \doteq \tilde{\nu}(o')]\!]^{S,\nu(\beta)} = 1 \\ & \Leftrightarrow [\![\tilde{\nu}(o \doteq o')]\!]^{S,\nu(\beta)} = 1 \end{array}$$

Cas $\psi \wedge \psi'$ where $\psi, \psi' \in \mathcal{F}^n_\Sigma$.

$$\begin{array}{ll} [\![\psi \wedge \psi']\!]^{\beta,S} = 1 & \Leftrightarrow [\![\psi]\!]^{\beta,S} \wedge [\![\psi']\!]^{\beta,S} = 1 \\ \textit{ind.hyp} & \Leftrightarrow [\![\tilde{\nu}(\psi)]\!]^{S,\nu(\beta)} \wedge [\![\tilde{\nu}(\psi)']\!]^{S,\nu(\beta)} = 1 \\ & \Leftrightarrow [\![\tilde{\nu}(\psi) \wedge \tilde{\nu}(\psi)']\!]^{S,\nu(\beta)} = 1 \\ & \Leftrightarrow [\![\tilde{\nu}(\psi \wedge \psi')]\!]^{S,\nu(\beta)} = 1 \end{array}$$

Cas $\neg\psi$ where $\psi \in \mathcal{F}^n_\Sigma$.

$$\begin{array}{ll} [\![\neg\psi]\!]^{\beta,S} = 1 & \Leftrightarrow \neg[\![\psi]\!]^{\beta,S} = 1 \\ \textit{ind.hyp.} & \Leftrightarrow \neg[\![\tilde{\nu}(\psi)]\!]^{S,\nu(\beta)} = 1 \\ & \Leftrightarrow [\![\neg\tilde{\nu}(\psi)]\!]^{S,\nu(\beta)} = 1 \\ & \Leftrightarrow [\![\tilde{\nu}(\neg\psi)]\!]^{S,\nu(\beta)} = 1 \end{array}$$

Cas $\exists x.\psi$ where $\psi \in \mathcal{F}^n_\Sigma$.

$$\begin{array}{ll} [\![\exists x.\psi]\!]^{\beta,S} = 1 & \Leftrightarrow \textit{exist } s \in \textit{dom}(S)^n.[\![\psi]\!]^{S,\beta[x/s]} = 1 \\ \textit{ind.hyp.} & \Leftrightarrow \textit{exist } s \in \textit{dom}(S)^n.[\![\tilde{\nu}(\psi)]\!]^{S,\nu(\beta[x/s])} = 1 \\ & \Leftrightarrow \textit{exist } s_1 \in \textit{dom}(S) \ldots s_n \in \textit{dom}(S).[\![\psi]\!]^{S,\nu(\beta[\nu_i(x)/s_i])} = 1 \\ & \Leftrightarrow [\![\exists\nu_1(x) \ldots \exists\nu_n(x).\tilde{\nu}(\psi)]\!]^{S,\nu(\beta)} = 1 \\ & \Leftrightarrow [\![\exists x.\tilde{\nu}(\psi)]\!]^{S,\nu(\beta)} = 1 \end{array}$$

□

Appendix B. Proofs for Section 8 (Difference Abstraction)

Lemma 21. *Let $h : S' \to \Delta$ be a Σ-abstraction and $\alpha : V \to \textit{dom}(S')$ and a variable assignment. For any expression $e \in \mathcal{E}_\Sigma$ with $V(e) \subseteq V$: $h([\![e]\!]^{S',\alpha}) \subseteq [\![e]\!]^{\Delta,h\circ\alpha}$.*

Proof. The proof is by induction on the structure of expressions $e \in \mathcal{E}_\Sigma$. Let α be a variable assignment into $\textit{dom}(S')$. For any expressions $e = e_1 \odot e_2$ where $\odot \in F^{(2)}$ we have:

$$\begin{array}{lcll} h'([\![e_1 \odot e_2]\!]^{S',\alpha}) & = & h'([\![e_1]\!]^{S',\alpha} \odot^{S'} [\![e_2]\!]^{S',\alpha}) & \\ & \subseteq & h'([\![e_1]\!]^{S',\alpha}) \odot^{\Delta} h'([\![e_2]\!]^{S',\alpha}) & \text{homomorph.} \\ & \subseteq & [\![e_1]\!]^{\Delta,h\circ\alpha} \odot^{\Delta} [\![e_2]\!]^{\Delta,h\circ\alpha} & \text{ind. hyp.} \\ & = & [\![e_1 \odot e_2]\!]^{\Delta,h\circ\alpha} & \end{array}$$

For any expression $e = x \in V$ we have:

$$h'([\![x]\!]^{S',\alpha}) = h'(\{\alpha(x)\}) = [\![x]\!]^{\Delta,h\circ\alpha}$$

For constant expressions $e = c \in C$ we have:

$$h'([\![c]\!]^{S',\alpha}) = h(c^{S'}) = c^{\Delta} = [\![c]\!]^{\Delta,h\circ\alpha} \qquad \text{homomorphism}$$

□