



HAL
open science

Exact Boolean Abstraction of Linear Equation Systems

Emilie Allart, Joachim Niehren, Cristian Versari

► **To cite this version:**

Emilie Allart, Joachim Niehren, Cristian Versari. Exact Boolean Abstraction of Linear Equation Systems. *Computation*, 2021, 9 (11), pp.32. 10.3390/COMPUTATION9110113 . hal-03384058v2

HAL Id: hal-03384058

<https://inria.hal.science/hal-03384058v2>

Submitted on 20 Oct 2021

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.

Exact Boolean Abstraction of Linear Equation Systems

Emilie Allart^{1,2,‡}  0000-0003-4965-1819, Joachim Niehren^{1,3,‡} and Cristian Versari^{1,2,‡*}

¹ BioComputing Team, CRISAL Lab, Lille,

² University of Lille,

³ Inria Lille

* Correspondence: cristian.versari@univ-lille.fr

‡ These authors contributed equally to this work.

Abstract: We study the problem of how to compute the boolean abstraction of the solution set of a linear equation system over the positive reals. We call a linear equation system ϕ exact for the boolean abstraction if the abstract interpretation of ϕ over the structure of booleans is equal to the boolean abstraction of the solution set of ϕ over the positive reals. Abstract interpretation over the booleans is thus complete for the boolean abstraction when restricted to exact linear equation systems, while it is not complete more generally. We present a new rewriting algorithm that makes linear equation systems exact for the boolean abstraction while preserving the solutions over the positive reals. The rewriting algorithm is based on the elementary modes of the linear equation system. The computation of the elementary modes may require exponential time in the worst case, but is often feasible in practice with freely available tools. For exact linear equation systems we can compute the boolean abstraction by finite domain constraint programming. This yields a solution of the initial problem that is often feasible in practice. Our exact rewriting algorithm has two further applications. First, it can be used to compute the sign abstraction of linear equation systems over the reals, as needed for analysing function programs with linear arithmetics. And second it can be applied to compute the difference abstraction of a linear equation system as used in change prediction algorithms for flux networks in systems biology.

Keywords: Linear equation systems; abstract interpretation; program analysis; systems biology.

1. Introduction

We develop approaches to remedy the incompleteness of abstract interpretation [1] of linear equation systems over the reals, in the algebra of booleans $\mathbb{B} = \{0, 1\}$ and the structure of signs $\mathbb{S} = \{-1, 0, 1\}$. These abstractions have various applications: In systems biology, the boolean abstraction underlies abstractions of chemical reactions networks into Boolean networks [2,3]. In program analysis, the sign abstraction can be applied to functional programs with arithmetics, for analysing the signs of the possible values of floating-point variables [4,5].

The soundness of abstract interpretations of first-order logic formulas without negation was shown by John [6–8]. It applies to the interpretation in any concrete structure S , as long as it is connected by a homomorphism $h : S \rightarrow \Delta$ to the abstract structure Δ . The concrete interpretation of a first-order formula ϕ is the set of concrete solutions $sol^S(\phi)$ and its abstract interpretation is the set of its abstract solutions $sol^\Delta(\phi)$. John’s soundness theorem (see Theorem 1 below) states that the set of abstract solutions of overapproximates the abstraction by h of the set of concrete solutions:

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

When choosing the operators in $\Sigma_{bool} = \{+, *, 0, 1\}$, the class of negation-free first-order formulas with operators in Σ_{bool} extends on the classes of linear and polynomial equation systems. In this article, we consider the boolean abstraction which is the unique homomorphism $h_{\mathbb{B}} : \mathbb{R}_+ \rightarrow \mathbb{B}$, and the sign abstraction which is the unique homomorphism $h_{\mathbb{S}} : \mathbb{R} \rightarrow \mathbb{S}$ with respect to the operators in Σ_{bool} . The boolean abstraction maps any strictly positive real to 1 and 0 to 0. The sign abstraction extends on the boolean

Citation: Allart, E.; Niehren, J.; Versari, C. Exact Boolean Abstraction of Linear Equation Systems. *Preprints* 2021, 1, 0. <https://doi.org/>

Received:

Accepted:

Published:

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

32 abstraction while mapping all strictly negative reals to -1 . We note that the structure of
 33 signs \mathbb{S} is *not* an algebra since the sum of a positive and a negative number may have
 34 any sign.

35 *Problematics*

A natural question is whether abstract interpretation is complete [9] for the ab-
 straction of formulas induced by a homomorphisms $h : S \rightarrow \Delta$, i.e, whether for all
 negation-free first-order formulas ϕ with the same operators:

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

36 We call a formula ϕ *h-exact* if it satisfies this property. A counter example against the
 37 completeness of abstraction interpretation for the boolean and the sign abstraction is
 38 the linear equation ϕ_0 equal to $x + y \overset{\circ}{=} x + z$. Here we write $\overset{\circ}{=}$ for the equality symbol
 39 inside the logic, in order to point out its difference from equality in the language of
 40 mathematics. Formula ϕ_0 is neither $h_{\mathbb{B}}$ -exact nor $h_{\mathbb{S}}$ -exact. This can be seen as follows.
 41 Over the reals ϕ_0 is equivalent to $y \overset{\circ}{=} z$, so that all assignments τ that are abstractions
 42 of concrete solutions of ϕ_0 must satisfy $\tau(y) = \tau(z)$. When interpreted abstractly over
 43 \mathbb{B} or \mathbb{S} , however, ϕ_0 admits the abstract solution $\tau = [x/1, y/1, z/0]$ which is not the
 44 abstraction of any concrete solution since $\tau(y) \neq \tau(z)$.

45 In order to deal with the incompleteness of abstract interpretation, we propose to
 46 study the following two questions for homomorphism $h : S \rightarrow \Delta$ where h is either the
 47 boolean abstraction $h_{\mathbb{B}}$ or the sign abstraction $h_{\mathbb{S}}$.

48 **Exact Rewriting** Can we rewrite linear equation systems to *h-exact* formulas?

49 **Computing Abstractions** Can we can compute the abstraction $h \circ \text{sol}^S(\phi)$ exactly for a
 50 given system of homogeneous linear equations ϕ ?

51 Geometrically speaking, the concrete solution sets $\text{sol}^{\mathbb{R}^+}(\phi)$ and $\text{sol}^{\mathbb{R}}(\phi)$ of homogeneous
 52 linear equation systems ϕ are polytopes – i.e. finite intersections of half spaces in $\mathbb{R}^{fv(\phi)}$.
 53 The problem of computing boolean abstractions or sign abstraction is thus to compute
 54 the h_{Δ} abstraction of a polytope given by a linear equation system.

55 For any *h-exact* formula ϕ , the computation of abstractions $h \circ \text{sol}^S(\phi)$ is equivalent
 56 to the computation of $\text{sol}^\Delta(\phi)$. Since the abstract structure Δ is finite for the boolean and
 57 sign abstraction, we can compute the set of abstract solutions in at most exponential
 58 time by a naive generate and test algorithm. Finite domain constraint programming
 59 [10] can be used to avoid the naive generation of all variable assignments to Δ in many
 60 practical cases. Therefore, any algorithm for exact rewriting induces an algorithm for
 61 computing abstractions that may be feasible in practice.

62 *Contributions*

63 Our main result is the first algorithm for exact rewriting that applies to linear
 64 equation systems for the Boolean abstraction. Based on this algorithm, we present a
 65 novel algorithm for computing the sign abstraction of linear equation systems.

66 *Exact Rewriting for the Boolean Abstraction.* In the first step, we study exact rewriting of
 67 (homogeneous) linear equation systems for boolean abstraction. The counter example ϕ_0 ,
 68 for instance, can be rewritten to $h_{\mathbb{B}}$ -exact formula $y \overset{\circ}{=} z$. The idea is to take the system of
 69 all linear consequences over \mathbb{R}_+ of the linear equation system. There may be infinitely
 70 many such consequences, but all of them are linear combinations of the extreme rays of
 71 the cone $\text{sol}^{\mathbb{R}^+}(\phi_0)$. Up to normalization, there are only finitely many extreme directions,
 72 which are known as the elementary modes of the linear equation system [11–14]. These
 73 can be computed by libraries from computational geometry [15] in at most exponential
 74 time. Nevertheless, the computation is often well-behaved in practice.

75 Based on the elementary modes (Folklore Theorem 2), we can rewrite any (ho-
 76 mogeneous) linear equation system into quasi-positive and strongly-triangular linear

77 equation system that is equivalent over \mathbb{R}_+ (Corollary 1), that can be computed in at
 78 most exponential time. As we will prove, such systems are always $h_{\mathbb{B}}$ -exact (Theorem 3).
 79 Hence, any system of linear equations can be converted in at most exponential time to
 80 some \mathbb{R}_+ -equivalent $h_{\mathbb{B}}$ -exact formula.

81 Note that $h_{\mathbb{B}}$ -exact formulas may still not be \mathbb{S} -exact. A counter example is the
 82 strongly-triangular quasi-positive linear system $u + v \stackrel{\circ}{=} x \wedge u + v \stackrel{\circ}{=} y$. It is not $h_{\mathbb{S}}$ -exact,
 83 since it entails $x \stackrel{\circ}{=} y$ over \mathbb{R} but still has the abstract solution $[u/1, v/-1, x/1, y/-1]$
 84 over \mathbb{S} which maps x and y to distinct signs. Indeed, we don't have any idea of how to
 85 do exact rewriting for the sign abstraction. The problem is that positivity is essential
 86 for our approach. And since the addition of positive and negative numbers may have
 87 any sign, \mathbb{S} fails to be an algebra, making the analogous argument as in the proof of
 88 \mathbb{B} -exactness fail.

89 *Extension to $h_{\mathbb{B}}$ -Mixed Systems.* In the second step, we introduce $h_{\mathbb{B}}$ -mixed systems, which
 90 by Theorem 4 generalize on systems of 1. linear equations, 2. positive polynomial
 91 equations $p \stackrel{\circ}{=} 0$, and 3. positive polynomial inequations $p \stackrel{\circ}{\neq} 0$, where p is a positive
 92 polynomial without constant term. We then show our main result:

93 **Theorem 5 (Main).** Any $h_{\mathbb{B}}$ -mixed systems can be converted to a $h_{\mathbb{B}}$ -exact formula by
 94 converting its linear subsystem to an $h_{\mathbb{B}}$ -exact formula.

95 The correctness of the algorithm for $h_{\mathbb{B}}$ -mixed systems relies on the notion of $h_{\mathbb{B}}$ -
 96 invariant formulas that we introduce. The class of $h_{\mathbb{B}}$ -invariant formulas subsume
 97 systems of positive polynomial equations $p \stackrel{\circ}{=} 0$ and inequations $p \stackrel{\circ}{\neq} 0$, where p is a
 98 positive polynomials without constant terms.

99 *Computing Sign Abstractions.* In the third step, we present an algorithm for computing the
 100 sign abstraction of (homogeneous) systems of linear equations based on exact rewriting
 101 for the boolean abstraction (Theorem 6). For this, we decompose the sign abstraction
 102 into the boolean abstraction based on a function that is definable in first-order logic. This
 103 function decomposes real numbers into their positive part x and negative part y . At least
 104 one of these two parts must be zero, which can be expressed by the polynomial equation
 105 $x * y \stackrel{\circ}{=} 0$. The positivity of x can be expressed by $\exists z. x \stackrel{\circ}{=} z * z$ and the positivity of y
 106 in analogy. In this way, we can reduce the problem of computing $h_{\mathbb{S}} \circ \text{sol}^{\mathbb{R}}(\phi)$ to the
 107 problem of computing $h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}_+}(\phi')$ for some existentially quantified $h_{\mathbb{B}}$ -mixed system
 108 ϕ' that we can make $h_{\mathbb{B}}$ -exact based on our main Theorem 5.

109 *Application to Program Analysis.* We show how to apply the computation of the sign
 110 abstraction of linear equation systems in order to improve the analysis of functional
 111 programs with arithmetics. For finding program errors there it can be useful to know
 112 about the possible signs of the values of program variables. We elaborate an example in
 113 the final Section 10.

114 *Implementation.* We implemented the $h_{\mathbb{B}}$ -exact rewriting algorithm for $h_{\mathbb{B}}$ -mixed systems
 115 from the main Theorem 5 in Python and plan to freely publish our tool soon. For this we
 116 used the cddlib tool from computational geometry [16] for computing elementary modes.
 117 We also used finite domain constraint programming with Minizinc [17] for computing
 118 the set of boolean solutions over logical formulas. Some successful experiments are
 119 mentioned in the related work subsection below. We did not yet implemented the
 120 algorithm for computing sign abstractions, nor its application to program analysis
 121 though.

122 Related Work

123 We start with related work by the authors and then move to related work by others.

124 *Change Prediction of Reaction Networks.* Our main Theorem 5 was recently applied to the
 125 change prediction of reaction networks in systems biology [6]. Indeed, the development
 126 of the present article was originally motivated by this application. The problem there is to

127 compute the difference abstraction of linear equation systems, expressing the steady state
 128 semantics of chemical reaction networks. Two difference abstractions were considered,
 129 $h_{\Delta_3} : \mathbb{R}_+^2 \rightarrow \{\Delta, \nabla, \approx\}$ and a refinement thereof $h_{\Delta_6} : \mathbb{R}_+^2 \rightarrow \{\uparrow, \downarrow, \sim, \uparrow, \downarrow, \approx\}$. In
 130 analogy to the approach adopted for computing sign abstractions (step three above), the
 131 algorithmic approach presented there is to decompose the difference abstractions h_{Δ_3}
 132 and h_{Δ_6} into the boolean abstraction $h_{\mathbb{B}}$ and functions that are definable in first-order
 133 logic. The elaboration of this approach, however, is quite different for reflecting the
 134 nature of the difference abstractions.

135 *Experimentation.* We tested our implementation of the exact rewriting algorithm for the
 136 boolean abstraction successfully for computing difference abstractions in the application
 137 of change prediction in systems biology. The experimental results are presented in [6] are
 138 generally encouraging. They show that $h_{\mathbb{B}}$ -exact rewriting based on elementary modes
 139 in combination with finite domain constraint programming may indeed avoid naive
 140 generate and test in many practical examples. In some of these examples, however, the
 141 overall computation time still took some hours.

142 *Abstracting Reaction Networks to Boolean Networks.* Independently, the authors proposed an
 143 abstraction of chemical reaction networks to boolean networks in [18], whose precision
 144 can be improved by using the $h_{\mathbb{B}}$ -exact rewriting of $h_{\mathbb{B}}$ -mixed equation systems.

145 *Alternative Algorithm for Computing Sign Abstractions.* An alternative algorithm for com-
 146 puting the sign abstraction of linear equation systems (and thus also the boolean ab-
 147 straction) can be obtained by John’s overapproximation theorem [6]. It shows that it
 148 is sufficient to generate the finitely many abstract solutions in $\tau \in \text{sol}^{\mathbb{S}}(\phi)$ and then to
 149 check for each of them whether there exists a concrete solution σ such that $\tau = h_{\mathbb{S}} \circ \sigma$. In
 150 order to perform this latter test, note that $h_{\Delta}(x) \stackrel{\circ}{=} 1$ is equivalent to the strict inequation
 151 $x > 0$ and $h_{\mathbb{S}}(x) \stackrel{\circ}{=} 0$ by the equation $x \stackrel{\circ}{=} 0$. Similarly, $h_{\mathbb{S}}(x) \stackrel{\circ}{=} -1$ can be defined
 152 by the strict inequation $x < 0$. Whether there exists a concrete solution $\sigma \in \text{sol}^{\mathbb{R}}(\phi)$
 153 such $\tau = h \circ \sigma$ is thus equivalent to the satisfiability of $\phi \wedge \bigwedge_{x \in \text{fv}(\phi)} h_{\mathbb{S}}(x) \stackrel{\circ}{=} \tau(x)$ over
 154 \mathbb{R} , where $\text{fv}(\phi)$ is the set of free variables of ϕ . The satisfiability of systems of strict
 155 linear inequations and homogeneous linear equations without constant terms over \mathbb{R}
 156 are known to be decidable since at least 1926 [19]. But still one has to generate the set
 157 of all abstract solutions $\text{sol}^{\mathbb{S}}(\phi)$. The new algorithm presented above avoids generating
 158 this set.

159 *Abstract Program Interpretation over Numerical Domains.* In abstract interpretation [20],
 160 non relational domains permit to approximate the set of values of program variables
 161 while ignoring the relationship to the values of others. Well-known non-relational
 162 numerical domains include the interval domain [21] describing invariants of the form
 163 $\bigwedge_{i=1}^m x_i \in [r_i, r'_i]$ with reals $r_i \leq r'_i$ and the constant propagation domain for invariants of
 164 the form $\bigwedge_{i=1}^m x_i \stackrel{\circ}{=} r_i$ [22].

165 Abstract interpretation of relational domains may yield better approximations
 166 that with non relational domains, since relationships between the values of different
 167 variables can be taken into account. Well-known relational numerical domains include
 168 the polyhedral domain [4]. A polyhedron is the solution set of systems of inhomogeneous
 169 linear inequations of the form $n_1x_1 + \dots + n_mx_m \leq r$. Alternatively, the linear equality
 170 domain [23] was considered. These are defined by system of inhomogeneous linear
 171 equations $n_1x_1 + \dots + n_mx_n \stackrel{\circ}{=} r$.

172 In the present paper, we study the problem of computing the sign abstraction of
 173 polytopes represented by homogeneous linear equation systems. The polytopes can be
 174 obtained by existing methods for the abstract program interpretation over the polyhedral
 175 domain. One weakness of our approach is that we study the homogeneous case only, so
 176 that we can only abstract polytope and not more general polyhedrons.

177 *Outline*

178 In Section 2 we recall preliminaries on homomorphisms between Σ -structures. In
 179 Section 3 the first-order logic is recalled. John's theorem and its relation to the soundness
 180 and completeness of abstract interpretation in the classical framework are discussed in
 181 Section 4. We discuss how to make linear equation system quasi-positive and strongly
 182 triangular based on elementary modes in Section 5. These properties can be used to
 183 prove $h_{\mathbb{B}}$ -exactness as we show in Section 6, and thus to obtain an $h_{\mathbb{B}}$ -exact rewriting of
 184 linear equation systems. We introduce the notion of $h_{\mathbb{B}}$ -invariance in Section 7 and apply
 185 it in Section 8 to lift the $h_{\mathbb{B}}$ -exact rewriting algorithm from linear to $h_{\mathbb{B}}$ -mixed systems.
 186 This allows us to define the sign abstraction of linear equation systems on Section 9. We
 187 finally apply this result in Section 10 to the sign analysis of functional programs with
 188 arithmetic.

189 **2. Homomorphisms on Σ -Structures**

190 We need some basic notation from set theory and standard notion of universal
 191 algebra such as Σ -algebras, Σ -structures and homomorphism.

192 For any set A and $n \in \mathbb{N}$, the set of n -tuples of elements in A is denoted by A^n .
 193 For finite sets A the number of elements of A is denoted by $|A|$. Furthermore, for any
 194 function $f : A \rightarrow B$ we define the function $f^2 : A^2 \rightarrow B^2$ such that $f^2(a, a') = (f(a), f(a'))$
 195 for all $a, a' \in A$.

196 *2.1. Σ -Algebras*

197 We next recall the notion of Σ -algebras. Let $\Sigma = \cup_{n \geq 0} F^{(n)} \uplus C$ be a ranked signature.
 198 We call the elements of $f \in F^{(n)}$ are called n -ary function symbols, even though we may
 199 also use them as $n + 1$ -ary relation symbols later on when moving to Σ -structures. The
 200 elements in $c \in C$ are called the constants of Σ .

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

203 Let $\mathbb{B} = \{0, 1\}$ be the set of booleans, \mathbb{N} the set of natural numbers including 0, \mathbb{Z}
 204 the set of integers, \mathbb{R} the set of real numbers, and \mathbb{R}_+ the set of positive real numbers
 205 including 0. Note that $\mathbb{B} \subseteq \mathbb{N} \subseteq \mathbb{R}_+ \subseteq \mathbb{R}$ and $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{R}$. Let the addition on the
 206 reals be the binary function $+^{\mathbb{R}} : \mathbb{R}^2 \rightarrow \mathbb{R}$ and the multiplication the binary function
 207 $*^{\mathbb{R}} : \mathbb{R}^2 \rightarrow \mathbb{R}$. Let the addition on the positive real numbers $+^{\mathbb{R}_+} : \mathbb{R}_+^2 \rightarrow \mathbb{R}_+$ be equal
 208 to the restriction $+^{\mathbb{R}}|_{\mathbb{R}_+ \times \mathbb{R}_+}$ and the multiplication $*^{\mathbb{R}_+} : \mathbb{R}_+^2 \rightarrow \mathbb{R}_+$ be the restriction
 209 $*^{\mathbb{R}}|_{\mathbb{R}_+ \times \mathbb{R}_+}$.

210 Let $\Sigma_{bool} = \{+, *\} \cup \{0, 1\}$ be the set of boolean operators where $+$ and $*$ are binary
 211 function symbols and 0 and 1 constants. Note that constant 0 is freely overloaded with
 212 the boolean 0 and the constant 1 with the boolean 1.

213 **Example 2.** The set of positive reals \mathbb{R}_+ can be turned into a Σ_{bool} -algebra, in which the functions
 214 symbols are interpreted as binary functions $+^{\mathbb{R}_+}$ and $*^{\mathbb{R}_+}$. The constants are interpreted by
 215 themselves $0^{\mathbb{R}_+} = 0$ and $1^{\mathbb{R}_+} = 1$.

216 **Example 3.** The set of Booleans $\mathbb{B} = \{0, 1\} \subseteq \mathbb{R}_+$ equally defines a Σ_{bool} -algebra. There,
 217 the function symbols are interpreted as a disjunction $+^{\mathbb{B}} = \vee^{\mathbb{B}}$ and conjunction $*^{\mathbb{B}} = \wedge^{\mathbb{B}}$ on
 218 Booleans. The constants are interpreted by themselves $0^{\mathbb{B}} = 0$ and $1^{\mathbb{B}} = 1$.

219 *2.2. Σ -Structures*

220 We next recall the usual generalization of Σ -algebras to Σ -structures. The objective
 221 is to generalize from functions to relations. For this, we consider n -ary function symbols
 222 as $n+1$ -ary relation symbols.

d	d'	$d +^{\mathbb{S}} d'$	$d *^{\mathbb{S}} d'$	$d -^{\mathbb{S}} d'$	$d /^{\mathbb{S}} d'$
-1	1	$\{-1, 0, 1\}$	$\{-1\}$	$\{-1\}$	$\{-1\}$
-1	0	$\{-1\}$	$\{0\}$	$\{-1\}$	\emptyset
-1	-1	$\{-1\}$	$\{1\}$	$\{-1, 0, 1\}$	$\{1\}$

d	d'	$d +^{\mathbb{S}} d'$	$d *^{\mathbb{S}} d'$	$d -^{\mathbb{S}} d'$	$d /^{\mathbb{S}} d'$
0	1	$\{1\}$	$\{0\}$	$\{-1\}$	$\{0\}$
0	0	$\{0\}$	$\{0\}$	$\{0\}$	\emptyset
0	-1	$\{-1\}$	$\{0\}$	$\{1\}$	$\{0\}$

d	d'	$d +^{\mathbb{S}} d'$	$d *^{\mathbb{S}} d'$	$d -^{\mathbb{S}} d'$	$d /^{\mathbb{S}} d'$
1	1	$\{1\}$	$\{1\}$	$\{-1, 0, 1\}$	$\{1\}$
1	0	$\{1\}$	$\{0\}$	$\{1\}$	\emptyset
1	-1	$\{-1, 0, 1\}$	$\{-1\}$	$\{1\}$	$\{-1\}$

Figure 1. Evaluation in the Σ_{arith} -structure of signs \mathbb{S} .

Definition 4. A Σ -structure $\Delta = (dom(\Delta), \cdot^{\Delta})$ consists of a set $dom(\Delta)$ and an interpretation \cdot^{Δ} such that $c^{\Delta} \in dom(\Delta)$ for all $c \in C$ and $f^{\Delta} \subseteq dom(\Delta)^{n+1}$ for all $f \in F^{(n)}$.

Clearly, any Σ -algebra is also a Σ -structure. Note also that symbols in $F^{(0)}$ are interpreted as monadic relations, i.e., as subsets of the domain, in contrast to constants in C that are interpreted as elements of the domain.

We denote the subtraction on the reals by the binary function $-^{\mathbb{R}} : \mathbb{R}^2 \rightarrow \mathbb{R}$ and the division on the reals by the ternary relation $/^{\mathbb{R}} \subseteq \mathbb{R}^2 \times \mathbb{R}$. Note that division by zero is undefined. Note also that subtraction on \mathbb{R}_+ would yield only a partial function.

Let $\Sigma_{arith} = \{+, *, -, /\} \cup \{0, 1\}$ be the arithmetic signature, where 0 and 1 are constants, and all other operators are binary function symbols. Again, we freely overload to constant 0 with real number 0 and the constant 1 with the real number 1.

Example 5. The set of reals \mathbb{R} can be turned into a Σ_{arith} -structure, with the interpretation of the binary functions symbols as the ternary relations $+^{\mathbb{R}}, *^{\mathbb{R}}, -^{\mathbb{R}}, /^{\mathbb{R}}$. The constants are interpreted by themselves $0^{\mathbb{R}} = 0$ and $1^{\mathbb{R}} = 1$. Note that $/^{\mathbb{R}}$ is a partial but not a total function, since division by 0 is not defined. So we must see $/^{\mathbb{R}}$ as a ternary relation, so that \mathbb{R} is not a Σ_{arith} -algebra. It still is a Σ_{bool} -algebra though.

Example 6. The set of signs $\{-1, 0, 1\} \subseteq \mathbb{R}$ can be turned into a Σ_{arith} -structure $\mathbb{S} = (\{-1, 0, 1\}, \cdot^{\mathbb{S}})$ with the interpretation $+^{\mathbb{S}}, -^{\mathbb{S}}, *^{\mathbb{S}}$ and $/^{\mathbb{S}}$ given in Fig. 1. The constants are interpreted by themselves $0^{\mathbb{S}} = 0$ and $1^{\mathbb{S}} = 1$. Note that all $+^{\mathbb{S}}$ contains $(-1, 1, -1)$, $(-1, 1, 1)$ and $(-1, 1, 0)$ meaning that the sum of a strictly negative and a strictly positive real has a sign in $-1 +^{\mathbb{S}} 1$, so it may either be strictly positive, strictly negative, or zero. So \mathbb{S} is a Σ_{arith} -structure and even when restricting the signature to Σ_{bool} it remains a Σ_{bool} -structure that is not a Σ_{bool} -algebra.

2.3. Homomorphisms

We recall the standard notion of homomorphism for Σ -structures which can also be applied to Σ -algebras.

Definition 7. A homomorphism between two Σ -structures S and Δ is a function $h : dom(S) \rightarrow dom(\Delta)$ such that for $c \in C$, $n \in \mathbb{N}$, $f \in F^{(n)}$, and $s_1, \dots, s_{n+1} \in dom(S)$:

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

$$\begin{aligned} \llbracket c \rrbracket^{\sigma, S} &= \{c^S\} \\ \llbracket x \rrbracket^{\sigma, S} &= \{\sigma(x)\} \\ \llbracket f(e_1, \dots, e_n) \rrbracket^{\sigma, S} &= \cup \{f^S(s_1, \dots, s_n) \mid s_i \in \llbracket e_i \rrbracket^{\sigma, S} \text{ for } 1 \leq i \leq n\} \end{aligned}$$

Figure 2. Set-valued interpretation of expressions $\llbracket e \rrbracket^{\sigma, S} \subseteq \text{dom}(S)$.

253 We can convert any $n + 1$ -ary relation to a n -ary set valued functions. In this
254 way any n -function is converted to a n -ary set valued n -functions. In other words,
255 functions of type $D^n \rightarrow D$ are converted to functions of type $D^n \rightarrow 2^D$ where $D =$
256 $\text{dom}(\Delta)$. In set-valued notation, the second condition on homomorphism can then be
257 rewritten equivalently as $h(f^S(s_1, \dots, s_n)) \subseteq f^\Delta(h(s_1), \dots, h(s_n))$. A homomorphism
258 for Σ -algebras thus satisfies $h(c^S) = c^\Delta$ and for all function symbols $f \in F^{(n)}$ and
259 $s_1, \dots, s_n \in \text{dom}(S)$ it satisfies $h(f^S(s_1, \dots, s_n)) = f^\Delta(h(s_1), \dots, h(s_n))$.

260 The boolean abstraction is the function $h_{\mathbb{B}} : \mathbb{R}_+ \rightarrow \mathbb{B}$ with $h_{\mathbb{B}}(0) = 0$ and $h_{\mathbb{B}}(r) = 1$
261 if $r > 0$.

262 **Lemma 8.** *The boolean abstraction $h_{\mathbb{B}}$ is a homomorphism between Σ_{bool} -algebras.*

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

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

263 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
264 both constants $c \in C$ we have that $h_{\mathbb{B}}(c^{\mathbb{R}_+}) = h_{\mathbb{B}}(c) = c = c^{\mathbb{B}}$.

265 The sign abstraction is the function $h_{\mathbb{S}} : \mathbb{R} \rightarrow \mathbb{S}$ with $h_{\mathbb{S}}(0) = 0$, $h_{\mathbb{S}}(r) = -1$ for all
266 strictly negative reals $r < 0$ and $h_{\mathbb{S}}(r) = 1$ for all strictly positive reals $r > 0$.

267 **Lemma 9.** *The sign abstraction $h_{\mathbb{S}}$ is a homomorphism between Σ_{arith} -structures.*

268 **Proof** Let $r, r' \in \mathbb{R}$. For the multiplication we have $h_{\mathbb{S}}(r *^{\mathbb{R}} r') = h_{\mathbb{S}}(r) *^{\mathbb{R}} h_{\mathbb{S}}(r')$ and
269 thus $h_{\mathbb{S}}(r *^{\mathbb{R}} r') \in \{h_{\mathbb{S}}(r) *^{\mathbb{R}} h_{\mathbb{S}}(r')\} = h_{\mathbb{S}}(r) *^{\mathbb{S}} h_{\mathbb{S}}(r')$. For the addition, we have to
270 distinguish cases. If r and r' have the same sign, so $r +^{\mathbb{R}} r'$ has the same sign, so that
271 we have $h_{\mathbb{S}}(r +^{\mathbb{R}} r') \in h_{\mathbb{S}}(r) +^{\mathbb{S}} h_{\mathbb{S}}(r')$. If $r > 0$ and $r' < 0$ or vice versa then we have
272 $h_{\mathbb{S}}(r) +^{\mathbb{S}} h_{\mathbb{S}}(r') = \mathbb{S}$ so that $h_{\mathbb{S}}(r +^{\mathbb{R}} r') \in \mathbb{S} = h_{\mathbb{S}}(r) +^{\mathbb{S}} h_{\mathbb{S}}(r')$. The treatment of $-^{\mathbb{S}}$ and
273 $/^{\mathbb{S}}$ is similar. For the constants we have $h_{\mathbb{S}}(0^{\mathbb{R}}) = 0^{\mathbb{S}}$ and $h_{\mathbb{S}}(1^{\mathbb{R}}) = 1^{\mathbb{S}}$.

274 3. First-Order Logic

275 We recall the syntax and semantics of first-order logic with equality. For this, we fix
276 a countably infinite set of variables \mathcal{V} that will be ranged over by x, y, z .

277 3.1. Expressions

278 Given a ranked signature with constants and function symbols $\Sigma = C \cup \bigcup_{n \geq 0} F^{(n)}$,
279 the set of Σ -expressions contains all terms that can be constructed from constants and
280 variables by using function symbols:

$$281 \quad e_1, \dots, e_n \in \mathcal{E}_{\Sigma} \quad ::= x \mid c \mid f(e_1, \dots, e_n) \quad \text{where } c \in C, n \geq 0, f \in F^{(n)}, x \in \mathcal{V}$$

282 Let $fv(e)$ be the set of all variables that occur in e . Given a subset $V \subseteq \mathcal{V}$ let $\mathcal{E}_{\Sigma}(V)$ be the
283 subset of expression $e \in \mathcal{E}_{\Sigma}$ with $fv(e) \subseteq V$.

284 The semantics of Σ -expressions is defined in Fig. 2. For any Σ -structure S and
285 variable assignment $\sigma : V \rightarrow \text{dom}(S)$, any expression $e \in \mathcal{E}_{\Sigma}(V)$ denotes a set of values
286 $\llbracket e \rrbracket^{\sigma, S} \subseteq \text{dom}(S)$. This set is defined recursively by set-valued interpretation of the
287 operators of the expressions in the structure S . If S is a Σ -algebra, then the result will
288 always be a singleton.

$$\begin{aligned}
\llbracket e \stackrel{\circ}{=} e' \rrbracket^{\sigma, S} &= \begin{cases} 1 & \text{if } \llbracket e \rrbracket^{\sigma, S} \cap \llbracket e' \rrbracket^{\sigma, S} \neq \emptyset \\ 0 & \text{else} \end{cases} & \llbracket \phi \wedge \phi' \rrbracket^{\sigma, S} &= \llbracket \phi \rrbracket^{\sigma, S} \wedge^{\mathbb{B}} \llbracket \phi' \rrbracket^{\sigma, S} \\
\llbracket \neg \phi \rrbracket^{\sigma, S} &= \neg^{\mathbb{B}}(\llbracket \phi \rrbracket^{\sigma, S}) & \llbracket \exists x. \phi \rrbracket^{\sigma, S} &= \begin{cases} 1 & \text{if exists } s \in \text{dom}(S). \\ & \llbracket \phi \rrbracket^{\sigma[x/s], S} = 1 \\ 0 & \text{else} \end{cases}
\end{aligned}$$

Figure 3. Interpretation of formulas $\phi \in \mathcal{F}_{\Sigma}(V)$ as truth values $\llbracket \phi \rrbracket^{\sigma, S} \in \mathbb{B}$ over a Σ -structure S given a variable assignment $\sigma : V \rightarrow \text{dom}(S)$.

289 3.2. Logic Formulas

290 The set of first-order formulas is the set of terms constructed with the usual first-
291 order connectives from equations with symbols in Σ and variables in \mathcal{V} :

$$292 \quad \phi \in \mathcal{F}_{\Sigma} ::= e \stackrel{\circ}{=} e' \mid \exists x. \phi \mid \phi \wedge \phi' \mid \neg \phi \quad \text{where } e, e' \in \mathcal{E}_{\Sigma} \text{ and } x \in \mathcal{V}$$

293 A Σ -formula $\phi \in \mathcal{F}_{\Sigma}$ is a term, which either is a Σ -equation $e \stackrel{\circ}{=} e'$ with variables
294 in \mathcal{V} , an existentially quantified formula $\exists x. \phi$, a conjunction $\phi \wedge \phi'$, or a negation $\neg \phi$.
295 A system of Σ -equations is a conjunction of equations $e_1 \stackrel{\circ}{=} e'_1 \wedge \dots \wedge e_n \stackrel{\circ}{=} e'_n$ where
296 $e_1, e'_1, \dots, e_n, e'_n \in \mathcal{E}_{\Sigma}$.

297 The set of free variables $fv(\phi)$ contains all those variables of ϕ that occur outside
298 the scope of any existential quantifier in ϕ . Given a subset $V \subseteq \mathcal{V}$ we write $\mathcal{F}_{\Sigma}(V)$ for
299 the subset of formulas $\phi \in \mathcal{F}_{\Sigma}$ such that $fv(\phi) \subseteq V$.

300 First-order formulas can be defined for providing the missing logical operators.
301 First, we can define disjunctions $\phi \vee \phi' =_{\text{def}} \neg(\neg \phi \wedge \neg \phi')$ and implications $\phi \rightarrow \phi' =_{\text{def}}$
302 $\neg \phi \vee \phi'$, and second universally quantified formulas $\forall x. \phi =_{\text{def}} \neg \exists x. \neg \phi$. Note that these
303 formulas are not negation-free (and thus John's theorem cannot be applied to them).
304 Third, we define the valid formula $true =_{\text{def}} \exists x. x \stackrel{\circ}{=} x$. Fourth, we write $\bigwedge_{i=1}^n \phi_i$ instead
305 of $\phi_1 \wedge \dots \wedge \phi_n$. In the case $n = 0$ the conjunction is $true$. Fourth, for any vector of
306 variables $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_n) \in \mathcal{V}^n$ we will write $\exists \mathbf{y}. \phi$ instead of $\exists \mathbf{y}_1 \dots \exists \mathbf{y}_n. \phi$.

307 For any $V \subseteq \mathcal{V}$, the semantics of first-order formulas $\phi \in \mathcal{F}_{\Sigma}(V)$ for a Σ -structure S
308 and a variable assignment $\sigma : V \rightarrow \text{dom}(S)$ is the truth value $\llbracket \phi \rrbracket^{\sigma, S} \in \mathbb{B}$ defined in Fig. 3.
309 Note that the equality symbol $\stackrel{\circ}{=}$ is interpreted as nondisjointness, i.e., an equation $e \stackrel{\circ}{=} e'$
310 is true if and only if $\llbracket e \rrbracket^{\sigma, S} \cap \llbracket e' \rrbracket^{\sigma, S} \neq \emptyset$. In the case of Σ -algebras, the equality symbol $\stackrel{\circ}{=}$
311 is indeed interpreted as equality of singletons. In the case of more general Σ -structures,
312 though, it is *not* interpreted as set equality.

The set of solutions with domain V of a formula $\phi \in \mathcal{F}_{\Sigma}(V)$ over a Σ -algebra S is:

$$\text{sol}_V^S(\phi) = \{ \sigma : V \rightarrow \text{dom}(S) \mid \llbracket \phi \rrbracket^{\sigma, S} = 1 \}$$

313 If $V = fv(\phi)$ we omit the index V , i.e., $\text{sol}^S(\phi) = \text{sol}_V^S(\phi)$.

314 Two formulas $\phi, \phi' \in \mathcal{F}_{\Sigma}$ are called S -equivalent if they have the same solution sets
315 over S on $V = fv(\phi) \cup fv(\phi')$, that is $\text{sol}_V^S(\phi) = \text{sol}_V^S(\phi')$. Note that $y \stackrel{\circ}{=} y$ is equivalent to
316 $z \stackrel{\circ}{=} z$ and also to $true$, i.e., to $\exists x. x \stackrel{\circ}{=} x$.

317 3.3. Examples

Since $\mathbb{B} \subseteq \mathbb{R}_+$ we can define the boolean abstraction by a formula $y \stackrel{\circ}{=} h_{\mathbb{B}}(x)$ in
 $\mathcal{F}_{\Sigma_{\text{bool}}}$ over \mathbb{R}_+ with two variables $x, y \in \mathcal{V}$:

$$(x \stackrel{\circ}{=} 0 \wedge y \stackrel{\circ}{=} 0) \vee (\neg x \stackrel{\circ}{=} 0 \wedge y \stackrel{\circ}{=} 1)$$

Since $\mathbb{S} \subseteq \mathbb{R}$ we can define the sign abstraction by a formula $y \overset{\circ}{=} h_{\mathbb{S}}(x)$ in $\mathcal{F}_{\Sigma_{bool}}$ over \mathbb{R} with two variables $x, y \in \mathcal{V}$:

$$(x \overset{\circ}{=} 0 \wedge y \overset{\circ}{=} 0) \vee (x > 0 \wedge y \overset{\circ}{=} 1) \vee (x < 0 \wedge y + 1 \overset{\circ}{=} 0)$$

where:

$$\begin{aligned} x \geq 0 &=_{\text{def}} \exists z. x \overset{\circ}{=} z * z \\ x > 0 &=_{\text{def}} x \geq 0 \wedge \neg(x \overset{\circ}{=} 0) \\ x < 0 &=_{\text{def}} \neg x \geq 0 \end{aligned}$$

318 These definitions illustrate that both abstraction are closely related to strict inequations
319 $x > 0$ and $x < 0$. The boolean abstraction is concerned with strict positivity $x > 0$ while
320 the sign abstraction is also concerned with strict negativity $x < 0$.

321 3.4. Semantic Properties of Free and Bound Variables

322 We will need the following two standard lemmas on the rôle of free and bound
323 variables for the solution sets of logic formulas. For any subset of variable assignments R
324 of type $V' \rightarrow \text{dom}(S)$ and any disjoint sets of variables $V \cap V' = \emptyset$ we define $\text{ext}_V^S(R) =$
325 $\{\sigma \cup \sigma' \mid \sigma : V \rightarrow \text{dom}(S), \sigma' \in R\}$.

326 **Lemma 10 Cylindrification.** *If $V \cap \text{fv}(\phi) = \emptyset$ then $\text{sol}_{V \cup \text{fv}(\phi)}^S(\phi) = \text{ext}_V^S(\text{sol}^S(\phi))$.*

327 **Proof** We can show that $\llbracket e \rrbracket^{\sigma, S} = \llbracket e \rrbracket^{\sigma|_{\text{fv}(e)}, S}$ for all expressions $e \in \mathcal{E}_{\Sigma}$ with $\text{fv}(e)$ disjoint
328 to V and any variable assignment $\sigma : \text{fv}(e) \cup V \rightarrow \text{dom}(S)$ by induction on the structure
329 of expressions. From this we can prove for all formulas $\phi \in \mathcal{F}_{\Sigma}$ such that $\text{fv}(\phi)$ is disjoint
330 from V and $\sigma : \text{fv}(\phi) \cup V \rightarrow \text{dom}(S)$ that $\llbracket \phi \rrbracket^{\sigma, S} = \llbracket \phi \rrbracket^{\sigma|_{\text{fv}(\phi)}, S}$ by induction on the structure
331 of Σ -formulas. This implies the lemma.

332 The projection $\pi_a(f)$ of a function $f : A \rightarrow B$ is its restriction $f|_{A \setminus \{a\}}$. The projection
333 of a set F of functions $f : A \rightarrow B$ is $\pi_a(F) = \{\pi_a(f) \mid f \in F\}$.

334 **Lemma 11 Quantification is projection.** $\text{sol}^S(\exists x. \phi) = \pi_x(\text{sol}^S(\phi))$.

Proof This is follows from the semantics of existential quantified formulas as follows:

$$\text{sol}^S(\exists x. \phi) = \{\sigma|_{\text{fv}(\phi) \setminus \{x\}} \mid \sigma \in \text{sol}^S(\phi)\} = \pi_x(\text{sol}^S(\phi))$$

335

336 4. Abstract Interpretation

337 We recall the notion of Σ -abstractions and use them for abstracting set of concrete
338 solutions of logic formulas within the usual framework of abstract interpretation. Due
339 to John's theorem, this abstraction of logic formulas can be soundly approximated by
340 abstract interpretation over target structure of the Σ -abstraction. We then argue that the
341 notion of soundness shown by John's theorem [8] does indeed coincide with the notion of
342 soundness abstract interpretation in the classical framework of Cousot & Cousot [1]. We
343 then introduce the notion of exactness of a logic formula with respect to a Σ -abstraction
344 and relate it to the completeness of abstract interpretation.

345 4.1. John's Overapproximation for Σ -Abstractions

346 The notion of Σ -abstraction from [6] generalizes at the same time over the boolean
347 abstraction and the sign abstraction.

348 **Definition 12.** *A Σ -abstraction is a homomorphism $h : S \rightarrow \Delta$ between Σ -structures such that*
349 $\text{dom}(\Delta) \subseteq \text{dom}(S)$.

350 The boolean abstraction $h_{\mathbb{B}}$ is a Σ_{bool} -abstraction by Lemma 8. The sign abstraction
 351 $h_{\mathbb{S}}$ is a Σ_{bool} -abstraction by Lemma 9.

Let $h : S \rightarrow \Delta$ be a Σ -abstraction and $V \subseteq \mathcal{V}$. For any subset of assignments R of type $V \rightarrow \text{dom}(S)$ we define the abstraction:

$$h \circ R = \{h \circ \sigma : V \rightarrow \text{dom}(\Delta) \mid \sigma \in R\}$$

Theorem 1 John's Overapproximation [6–8]. For any Σ -abstraction $h : S \rightarrow \Delta$ between Σ -structures and any negation-free Σ -formula $\phi \in \mathcal{F}_{\Sigma}$:

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

352 John's theorem states that the abstraction with respect to h of the concrete solution
 353 set of a first-order formula can be overapproximated by abstract interpretation of the
 354 formula in the target structure of h .

355 We only give a brief sketch of the full proof which can be found in [6]. Let $V = \text{fv}(\phi)$
 356 and $\sigma : V \rightarrow \text{dom}(S)$. For any expression $e \in \mathcal{E}_{\Sigma}(V)$ we can show $h(\llbracket e \rrbracket^{\sigma, S}) = \llbracket e \rrbracket^{h \circ \sigma, \Delta}$ by
 357 induction on the structure of e . It then follows for any negation-free formula $\phi \in \mathcal{F}_{\Sigma}(V)$
 358 that $\llbracket \phi \rrbracket^{\sigma, S} \leq \llbracket \phi \rrbracket^{h \circ \sigma, \Delta}$. This is equivalent to that $\{h \circ \sigma \mid \sigma \in \text{sol}_V^S(\phi)\} \subseteq \text{sol}_V^{\Delta}(\phi)$ and
 359 thus $h \circ \text{sol}_V^S(\phi) \subseteq \text{sol}_V^{\Delta}(\phi)$. Since $V = \text{fv}(\phi)$ it follows that $h \circ \text{sol}^S(\phi) \subseteq \text{sol}^{\Delta}(\phi)$ as
 360 required.

361 4.2. Exactness of Σ -Formulas for Σ -Abstractions

362 As a new contribution, we introduce the notion of exactness of first-order formulas
 363 with respect to a Σ -abstraction.

364 **Definition 13 h -Exactness.** Let $h : S \rightarrow \Delta$ be a Σ -abstraction and $\phi \in \mathcal{F}_{\Sigma}(V)$ a formula. We
 365 call ϕ h -exact with respect to V if $h \circ \text{sol}_V^S(\phi) = \text{sol}_V^{\Delta}(\phi)$. We call ϕ h -exact if ϕ is h -exact
 366 with respect to $\text{fv}(\phi)$.

367 For instance, the linear equation system ϕ equal to $x + y \stackrel{\circ}{=} x + z$ is neither $h_{\mathbb{B}}$ -exact
 368 nor $h_{\mathbb{S}}$ -exact. However it is equivalent to $y \stackrel{\circ}{=} z$ which is both $h_{\mathbb{B}}$ -exact and $h_{\mathbb{S}}$ -exact. To
 369 see this note that $\tau = [x/1, y/1, z/0]$ belongs to $\text{sol}^{\mathbb{B}}(\phi)$ but not to $h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}^+}(\phi)$ since
 370 $\tau(y) \neq \tau(z)$. The same assignment also belongs to $\text{sol}^{\mathbb{S}}(\phi)$ but not to $h_{\mathbb{S}} \circ \text{sol}^{\mathbb{R}}(\phi)$ since
 371 $\tau(y) \neq \tau(z)$.

372 4.3. Soundness and Completeness of Abstract Interpretation

373 John's theorem is related to the soundness of abstract interpretation and the notion
 374 of exactness to its completeness. In order to state the precise relationship, we need to
 375 embed our setting into the classical framework of abstract interpretation [1,9].

When considering formulas as programs, the usual framework of abstract interpretation of programs applies to the interpretation of the formulas (programs) in the target structure of the Σ -abstraction. More formally, we fix a finite subset of variables $V \subseteq \mathcal{V}$ and consider the subset of formulas as programs:

$$\mathcal{P} = \{\phi \in \mathcal{F}_{\Sigma}(V) \mid \phi \text{ is negation-free}\}$$

The semantics of a program $\phi \in \mathcal{P}$ over a given Σ -structure S is the set of its solutions over S :

$$\llbracket \phi \rrbracket = \text{sol}^S(\phi)$$

The range of the semantics mapping is the space of concrete values $C = 2^{\{\sigma \mid \sigma : V \rightarrow \text{dom}(S)\}}$. Note that $(C, \subseteq, \cap, \cup)$ is a complete lattice. An abstract interpretation of a program $\phi \in \mathcal{P}$ maps ϕ to the set of its solutions over Δ :

$$\llbracket \phi \rrbracket^{\#} = \text{sol}^{\Delta}(\phi)$$

The range of the abstract interpretation is the abstract domain $A = 2^{\{\tau | \tau: V \rightarrow \text{dom}(\Delta)\}}$. Clearly, $(A, \subseteq, \cap, \cup)$ is also a complete lattice. We define the abstraction function $\alpha_h : C \rightarrow A$ of our Galois connection such that for subsets of concrete assignments $R \subseteq C$:

$$\alpha_h(R) = h \circ R$$

376 **Definition 14 Cousot & Cousot [1], Giacobazzi, Ranzato & Scozzari [9].** *An abstract*
 377 *interpretation $\llbracket \cdot \rrbracket^\sharp : \mathcal{P} \rightarrow A$ is sound for an abstraction $\alpha : C \rightarrow A$ with respect to the program*
 378 *semantics $\llbracket \cdot \rrbracket : \mathcal{P} \rightarrow C$ if for all programs $\phi \in \mathcal{P}$ it holds that $\alpha(\llbracket \phi \rrbracket) \subseteq \llbracket \phi \rrbracket^\sharp$. It is complete if*
 379 *all programs $\phi \in \mathcal{P}$ satisfy $\alpha(\llbracket \phi \rrbracket) = \llbracket \phi \rrbracket^\sharp$.*

380 John's theorem states that the abstract interpretation α_h of negation free-formulas
 381 $\phi \in \mathcal{P}$ over Δ is sound for the abstraction of $\text{sol}^S(\phi)$ with respect to the Σ -abstraction
 382 $h : S \rightarrow \Delta$. Furthermore, if all formulas of \mathcal{P} are h -exact then abstract interpretation
 383 over Δ is complete for abstraction α_h . As illustrated above abstract interpretation over \mathbb{B}
 384 fails to be complete for the abstraction $\alpha_{h_{\mathbb{B}}}$, and similarly, abstract interpretation over
 385 \mathbb{S} fails to be complete for the abstraction $\alpha_{h_{\mathbb{S}}}$. Note that the completeness of abstract
 386 interpretations has been largely studied in the context of program analysis (see e.g.
 387 Section 8 of [9] for an overview).

388 In the present article, we study the problem of exact rewriting for $h_{\mathbb{B}}$. The question
 389 is how to rewrite a Σ_{bool} -formula into a $h_{\mathbb{B}}$ -exact formula that is \mathbb{R}_+ -equivalent. Note
 390 that exact rewriting of linear equation system for $h_{\mathbb{B}}$ is a different problem than to decide
 391 whether abstract interpretation is complete for $\alpha_{h_{\mathbb{B}}}$ on linear equation systems. Still, both
 392 notions are closely related: exact rewriting can help to improve the precision of abstract
 393 interpretation just in the case where it is not already complete, i.e., maximally precise.
 394 Otherwise, exact rewriting is trivial.

395 In the case of the sign abstraction, we do not have any algorithmic idea of how to
 396 do exact rewriting for linear equation systems. Therefore, we study the easier problem
 397 of exact rewriting for the boolean abstraction of linear equation systems in the first place.
 398 Given an $h_{\mathbb{B}}$ -exact formula ϕ , we can compute the abstraction $h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}_+}(\phi) = \text{sol}^{\mathbb{B}}(\phi)$
 399 by finite domain constraints programming. We then use exact rewriting for the boolean
 400 abstraction to compute sign abstractions of linear equation systems $h_{\mathbb{S}} \circ \text{sol}^{\mathbb{R}}(\phi)$, rather
 401 than relying on exact rewriting for the sign abstraction itself. For this we use first-order
 402 definitions beside of finite domain constraint programming.

4.03 4.4. Galois Connection

4.04 We finally introduce the concretization operation that corresponds to the abstraction
 4.05 of the solution set of a logic formula with respect to a Σ -abstraction, and show that the
 4.06 pair of abstraction and concretization forms a Galois connection.

Given a Σ -abstraction $h : S \rightarrow \Delta$, and a set R of variable assignments to $\text{dom}(\Delta)$, we define the left-decomposition of R with respect to h as the following set of variable assignments to $\text{dom}(S)$:

$$h \ominus R =_{\text{def}} \{\sigma \mid h \circ \sigma \in R\}$$

So let $\alpha_h : C \rightarrow A$ be the abstraction induced by Σ -abstraction h . We define the corresponding concretization function $\gamma_h : A \rightarrow C$ such that for all abstract assignments $R \subseteq A$:

$$\gamma_h(R) = h \ominus R =_{\text{def}} \{\sigma \in C \mid h \circ \sigma \in R\}$$

Lemma 15. $(A, C, \alpha_h, \gamma_h)$ is a Galois connection, i.e. for all $R \in C$ and $T \in A$:

$$\alpha_h(R) \subseteq T \text{ if and only if } R \subseteq \gamma_h(T)$$

407 **Proof** If $h \circ R \subseteq T$ then $h \circ h \circ R \subseteq h \circ T$ and since $R \subseteq h \circ h \circ R$ we have $R \subseteq h \circ T$.
 408 If conversely $R \subseteq h \circ T$ then $h \circ R \subseteq h \circ h \circ T$ and since $h \circ h \circ T = T$ it follows that
 409 $h \circ R \subseteq T$.

410 5. Equation Systems, Positivity, and Triangularity

411 We study systems of Σ_{bool} -equations for positivity and triangularity. These notions
 412 will be essential for showing \mathbb{B} -exactness. We are not only interested in homogeneous
 413 linear equations but also in more general polynomial equations without constant term.

414 5.1. Classes of Equation Systems

Let $e_1, \dots, e_n \in \mathcal{E}_{\Sigma_{bool}}$ be a sequence of expressions and $n \in \mathbb{N}$. If $n \neq 0$ we define
 $\sum_{i=1}^n e_i =_{\text{def}} e_1 + \dots + e_n$ and $\prod_{i=1}^n e_i =_{\text{def}} e_1 * \dots * e_n$. For $n = 0$, we define $\sum_{i=1}^n e_i = 0$
 and $\prod_{i=1}^n e_i = 1$. Furthermore, for any expression $e \in \mathcal{E}_{\Sigma_{bool}}$ we define:

$$ne =_{\text{def}} \sum_{i=1}^n e \quad \text{and} \quad e^n =_{\text{def}} \prod_{i=1}^n e$$

A *polynomial (with natural coefficients)* is a Σ_{bool} -expression of the following form:

$$\sum_{j=1}^l n_j \prod_{k=1}^{i_j} x_{j,k}^{m_{j,k}}$$

415 where l and i_j are natural numbers, $x_{1,1}, \dots, x_{l,i_l}$ variables, all $n_j \neq 0$ are natural numbers
 416 called the *coefficients*, and all $m_{j,k} \neq 0$ are natural numbers called the *exponents*. The
 417 products $\prod_{k=1}^{i_j} x_{j,k}^{m_{j,k}}$ are called the *monomials* of the polynomial.

418 **Definition 16.** A polynomial $\sum_{j=1}^l n_j \prod_{k=1}^{i_j} x_{j,k}^{m_{j,k}}$ with natural coefficients $n_j \neq 0$ has no
 419 constant term if none of its monomials is equal to 1, i.e., $i_j \neq 0$ for all $1 \leq j \leq l$. It is linear if
 420 all its monomials are variables, i.e. $i_j = 1$ and $m^{j,1} = \dots = m^{j,i_j} = 1$ for all $1 \leq j \leq l$.

421 A *polynomial equation* is a Σ_{bool} -equation $p \stackrel{\circ}{=} p'$ between polynomials. A *polynomial*
 422 *equation system* is a system of polynomial equations.

423 Linear polynomials have the form $\sum_{j=1}^l n_j x_{j,1}$ where l and all $n_j \neq 0$ are naturals
 424 and all $x_{j,1}$ are variables. In particular, linear polynomials do not have a constant term.
 425 Note that the constant 0 is equal to the linear polynomial with $l = 0$. A (*homogeneous*)
 426 *linear equation* is a polynomial equation with linear polynomials, so without constant
 427 terms. A (*homogeneous*) *linear equation system* is a system of linear equations.

A (*homogeneous*) *integer matrix equation* has the form $A\mathbf{y} \stackrel{\circ}{=} \mathbf{0}$ where A is a $n \times m$
 matrix of integers for some naturals m, n such that $\mathbf{y} \in \mathcal{V}^m$ and $\mathbf{0} \in \{0\}^n$. Any integer
 matrix equation can be turned into a linear equation system with natural coefficients,
 by bringing the negative coefficients positively on the right-hand side. For instance, the
 linear integer matrix equation:

$$\begin{pmatrix} 3 & 0 \\ 2 & -5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \stackrel{\circ}{=} \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

corresponds to the following system of linear Σ_{bool} -equations:

$$3x \stackrel{\circ}{=} 0 \wedge 2x \stackrel{\circ}{=} 5y$$

428 Therefore, we will sometimes confuse an integer matrix equations with the correspond-
 429 ing system of linear Σ_{bool} -equations. Conversely any system of linear Σ_{bool} -equations
 430 can be converted into a integer matrix equation, by moving the positive right-hand sides

4.31 negatively to the left and factorizing the expressions for the different occurrences of the
4.32 same variable.

4.33 5.2. Positivity and Triangularity

4.34 We next define positivity and triangularity properties for equation systems. These
4.35 will be key properties to show \mathbb{B} -exactness of linear equation systems.

4.36 **Definition 17.** A Σ_{bool} -equation is called positive if it has the form $e \stackrel{\circ}{=} 0$ and quasi-positive if
4.37 it has the form $e \stackrel{\circ}{=} ny$, where $n \in \mathbb{N}$, $y \in \mathcal{V}$, and $e \in \mathcal{E}_{\Sigma_{\text{bool}}}$. We call a system of Σ_{bool} -equations
4.38 positive respectively quasi-positive if all its equations are.

4.39 This definition makes sense, since all constants in Σ_{bool} -expressions are positive
4.40 and all operators of Σ_{bool} -expressions preserve positivity. Note also that any positive
4.41 equation is quasi-positive since the constant 0 is equal to the polynomial $0y$.

4.42 This above system of linear equations is quasi-positive, but not positive since $5y$
4.43 appears on a right-hand side. More generally, the linear equation system for a integer
4.44 matrix equation $Ay \stackrel{\circ}{=} \mathbf{0}$ is positive if and only if all integers in A are positive, and
4.45 quasi-positive, if each line of A contains at most one negative integer.

4.46 **Definition 18.** We call a quasi-positive system of Σ_{bool} -equations triangular if it has the form
4.47 $\bigwedge_{i=1}^n e_i \stackrel{\circ}{=} n_i y_i$ such that the variables y_l are l -fresh for all $1 \leq l \leq n$, i.e., $y_l \notin \text{fv}(\bigwedge_{i=1}^{l-1} e_i \stackrel{\circ}{=} e'_i)$
4.48 and if $n_l \neq 0$ then $y_l \notin \text{fv}(e_l)$. We call the quasi-positive polynomial system strongly-triangular
4.49 if it is triangular and satisfies $n_l \neq 0$ for all $1 \leq l \leq n$.

4.50 The above linear equation system is triangular, but not strongly triangular since the
4.51 right-hand side of the first equation is 0. Consider an integer matrix equation $Ay \stackrel{\circ}{=} \mathbf{0}$. If
4.52 A is positive and triangular, then the corresponding linear equation system is positive
4.53 and triangular too. For being quasi-positive and strongly-triangular, the integers below
4.54 the diagonal of A must be negative, those on the diagonal must be strictly negative, and
4.55 those on the right of the diagonal must be positive.

4.56 5.3. Linear Equation Systems and Elementary Modes

4.57 We next show that elementary modes [11–14] can be used to transform systems of
4.58 linear equations into \mathbb{R}_+ -equivalent systems that are quasi-positive and strongly-triangular.

4.59 We first recall the necessary definitions and folklore results on elementary modes
4.60 and the double description method. We will limit the presentation to equations with in-
4.61 teger coefficients solved in \mathbb{R}_+ , since more general definitions and results for elementary
4.62 modes in \mathbb{R} are not needed for this paper.

4.63 **Definition 19.** The support of a function $\sigma : V \rightarrow \mathbb{R}$ is $\text{supp}(\sigma) = \{y \in V \mid \sigma(y) \neq 0\}$.

4.64 **Definition 20 Elementary Modes.** An elementary mode of an integer matrix $A \in \mathbb{Z}^{n,m}$ is
4.65 a vector $\mathbf{n} \in \mathbb{N}^n$ such that for any sequence of pairwise distinct variables $\mathbf{y} \in \mathcal{V}^n$ the function
4.66 $\sigma = [\mathbf{y}/\mathbf{n}]$ is a solution in $\text{sol}^{\mathbb{R}_+}(Ay \stackrel{\circ}{=} \mathbf{0})$ such that:

- 4.67 • $\text{supp}(\sigma)$ is minimal, i.e. there exist no $\sigma' \in \text{sol}^S(\phi)$ such that $\text{supp}(\sigma') \subsetneq \text{supp}(\sigma)$,
- 4.68 • σ is normalized, i.e. there exist variables y, y' in \mathbf{y} such that $\sigma(y)$ and $\sigma(y')$ are coprimes
4.69 (their greatest common divisor is 1).

4.70 The elementary modes of a matrix A are the extreme directions of the polyhedral
4.71 cone $\text{sol}^{\mathbb{R}_+}(Ay \stackrel{\circ}{=} \mathbf{0})$. This implies that any solution of the linear system can be expressed
4.72 as a weighted sum of its elementary modes, where all the weights are non negative. Due
4.73 to normalization, the number of elementary modes is finite for all integer matrices.

$$\phi_0 =_{\text{def}} \left\{ \begin{array}{l} y_1 = y_2 + y_3 \\ \wedge \\ y_1 = y_2 + y_4 \end{array} \right. \quad \left(\begin{array}{cccc} -1 & 1 & 1 & 0 \\ 1 & -1 & 0 & -1 \end{array} \right) \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} \stackrel{\circ}{=} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Figure 4. :A linear equation system and the corresponding integer matrix equation.

$$\text{emr}(\phi_0) =_{\text{def}} \exists x_0. \exists x_1. \left\{ \begin{array}{l} x_0 + x_1 \stackrel{\circ}{=} y_1 \\ \wedge \\ x_1 \stackrel{\circ}{=} y_2 \\ \wedge \\ x_0 \stackrel{\circ}{=} y_3 \\ \wedge \\ x_0 \stackrel{\circ}{=} y_4 \end{array} \right. \quad \left(\begin{array}{cc} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{array} \right) \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \stackrel{\circ}{=} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

Figure 5. The elementary mode rewriting and the corresponding matrix equation.

474 **Theorem 2 (Folklore).** For any integer matrix $A \in \mathbb{Z}^{m,n}$ one can compute a matrix of natural
 475 numbers $E \in \mathbb{N}^{n,o}$ in at most exponential time, such that the Σ_{bool} -formulas for $A\mathbf{y} \stackrel{\circ}{=} \mathbf{0}$ and
 476 $\exists \mathbf{x}. E\mathbf{x} \stackrel{\circ}{=} \mathbf{y}$ are \mathbb{R}_+ -equivalent for all vectors $\mathbf{y} \in \mathcal{V}^n$ and $\mathbf{x} \in \mathcal{V}^o$ of pairwise distinct variables.
 477 Furthermore, the o columns of E are the elementary modes of A .

478 The pair of matrices (A, E) is called a double description in Motzkin's double
 479 description method [11]. We note that Theorem 2 can be lifted to matrices of rational
 480 numbers \mathbb{Q} , since any rational matrix equation $A\mathbf{y} \stackrel{\circ}{=} \mathbf{0}$ can be rewritten to a integer
 481 matrix equation with the same \mathbb{R}_+ -solution set, by multiplying with the natural numbers
 482 in the denominators of the rational numbers. The freely available cddlib tool [16] in the
 483 rational mode inputs a matrix $A \in \mathbb{Q}^{n,m}$, and outputs the list of (integer) elementary
 484 modes of A . From this list, we can construct the matrix E for A by aligning the elementary
 485 modes of A as the columns of E .

486 The service of the cddlib tool is even more general, in that it applies to rational
 487 matrix inequations interpreted over the reals, rather than to rational matrix equations
 488 interpreted over the positive reals: it permits to compute the normalized extreme di-
 489 rections of the polyedral cones $\text{sol}^{\mathbb{R}}(B\mathbf{y} \geq \mathbf{0})$ for any rational matrix inequation B . If
 490 one wants to compute the elementary modes of integer matrix equations A – that is
 491 the normalized extreme directions of the polyhedral cone of a rational matrix equation
 492 over the positive reals $\text{sol}^{\mathbb{R}_+}(A\mathbf{y} \stackrel{\circ}{=} \mathbf{0})$ –, then one can chose $B = \begin{pmatrix} A \\ -A \\ Id \end{pmatrix}$ where Id
 493 is the identity matrix with as many columns than A , since $\text{sol}^{\mathbb{R}}(B\mathbf{y} \geq \mathbf{0}) = \text{sol}^{\mathbb{R}}(A\mathbf{y} \geq$
 494 $\mathbf{0} \wedge A\mathbf{y} \leq \mathbf{0} \wedge \mathbf{y} \geq \mathbf{0}) = \text{sol}^{\mathbb{R}_+}(A\mathbf{y} \stackrel{\circ}{=} \mathbf{0})$.

495 **Corollary 1 Elementary Mode Rewriting.** For any system of linear equations $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}$
 496 one can compute in at most exponential time an \mathbb{R}_+ -equivalent formula $\text{emr}(\phi) \in \mathcal{F}_{\Sigma_{\text{bool}}}$ that
 497 has the form $\exists \mathbf{x}. \phi'$ where ϕ' is quasi-positive and strongly-triangular system of equations.

498 **Proof** Any system of linear equations $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}$ can be converted into an \mathbb{R}_+ -equivalent
 499 integer matrix equation $A\mathbf{y} \stackrel{\circ}{=} \mathbf{0}$ where \mathbf{y} is a vector that contains all variables in $\text{fv}(\phi)$
 500 exactly once. Let E be a matrix of elementary modes of A from Theorem 2. The theorem
 501 states that $A\mathbf{y} \stackrel{\circ}{=} \mathbf{0}$ is \mathbb{R}_+ -equivalent to $\exists \mathbf{x}. E\mathbf{x} \stackrel{\circ}{=} \mathbf{y}$ for some vector of fresh variables \mathbf{x} . So
 502 let $\text{emr}(\phi)$ be $\exists \mathbf{x}. \phi'$ and ϕ' be $E\mathbf{x} \stackrel{\circ}{=} \mathbf{y}$. Since all entries of E are positive, the variables in \mathbf{y}
 503 are pairwise distinct, and the variables in \mathbf{x} are chosen freshly, it follows that ϕ' is both
 504 quasi-positive and strongly-triangular. \square

505 We have implemented the elementary mode rewriting in Python based on the
 506 cddlib tool [16] and plan to freely publish our tool soon. An example input is the system
 507 of linear Σ_{bool} -equations ϕ_0 given in Fig. 4. The corresponding integer matrix equation
 508 system is given there too. The elementary modes of the matrix of this system are the
 509 vectors $(1, 0, 1, 1)$ and $(1, 1, 0, 0)$. When putting these vectors in the columns of a new
 510 matrix, our tool returns the elementary mode rewriting $emr(\phi_0)$ in Fig. 5.

511 6. $h_{\mathbb{B}}$ -Exact Rewriting of Linear Equation Systems

512 Our next objective is to study the preservation of h -exactness by logical operators.
 513 The main difficulty of this paper is the fact that h -exactness is not preserved by conjunc-
 514 tion. Nevertheless, as we will show next, it is preserved by disjunction and existential
 515 quantification.

516 In order to do so we first show that h -exactness is preserved when adding variables.
 517 For this we have to assume that the Σ -abstraction h is surjective, which will be the case
 518 of all Σ -abstractions of interest.

519 **Lemma 21 Variable extension preserves exactness.** *Let $h : S \rightarrow \Delta$ be a Σ -abstraction that*
 520 *is surjective and $\phi \in \mathcal{F}_{\Sigma}(V)$ a formula. Then the h -exactness of ϕ implies the h -exactness of ϕ*
 521 *with respect to V .*

522 **Proof** This follows from that abstractions of solutions of ϕ can be extended arbitrarily to
 523 variables that do not appear freely in ϕ as stated by the following claim.

524 **Claim 22.** *For all $\sigma : V \rightarrow \Delta$: $\sigma \in h \circ sol^S(\phi)$ iff $\sigma_{|_{fv(\phi)}} \in h \circ sol^S(\phi)$.*

525 For the one direction let $\sigma \in h \circ sol_V^S(\phi)$. Then there exists $\sigma \in sol_V^S(\phi)$ such that
 526 $\sigma = h \circ \sigma$. Since $V \supseteq fv(\phi)$ it follows that $\sigma_{|_{fv(\phi)}} \in sol^S(\phi)$. Furthermore $\sigma_{|_{fv(\phi)}} =$
 527 $h \circ \sigma_{|_{fv(\phi)}}$ and thus $\sigma_{|_{fv(\phi)}} \in h \circ sol^S(\phi)$.

528 For the other direction let $\sigma_{|_{fv(\phi)}} \in h \circ sol^S(\phi)$. Then there exists $\sigma \in sol^S(\phi)$ such
 529 that $\sigma_{|_{fv(\phi)}} = h \circ \sigma$. For any $y \in V \setminus fv(\phi)$ let $s_y \in dom(S)$ be such that $h(s_y) = \sigma(y)$.
 530 Such values exists since h is surjective. Now define $\sigma' = \sigma[y/s_y \mid y \in V \setminus fv(\phi)]$. Since
 531 $V \supseteq fv(\phi)$ it follows that $\sigma' \in sol_V^S(\phi)$. Furthermore, $\sigma = h \circ \sigma'$, so $\sigma \in h \circ sol_V^S(\phi)$. \square

532 For the case of disjunction, we need a basic property of unions (joins) which fails
 533 for intersections (meets).

Lemma 23 Abstraction α_h preserves joins. *Let V be a set of variables, R_1 and R_2 be subsets*
of assignments of type $V \rightarrow dom(S)$ and $h : S \rightarrow \Delta$ be a Σ -abstraction. Then:

$$h \circ (R_1 \cup R_2) = h \circ R_1 \cup h \circ R_2$$

Proof This lemma follows from the following equivalences:

$$\begin{aligned} \tau \in h \circ (R_1 \cup R_2) &\Leftrightarrow \exists \sigma. \sigma \in R_1 \cup R_2 \wedge \tau = h \circ \sigma \\ &\Leftrightarrow \exists \sigma. (\sigma \in R_1 \vee \sigma \in R_2) \wedge \tau = h \circ \sigma \\ &\Leftrightarrow \exists \sigma. (\sigma \in R_1 \wedge \tau = h \circ \sigma) \vee (\sigma \in R_2 \wedge \tau = h \circ \sigma) \\ &\Leftrightarrow \tau \in h \circ R_1 \vee \tau \in h \circ R_2 \\ &\Leftrightarrow \tau \in h \circ R_1 \cup h \circ R_2 \end{aligned}$$

534

535 **Proposition 24.** *The disjunction of h -exact formulas is h -exact.*

Proof Let ϕ_1 and ϕ_2 be negation free formulas that are h -exact. Let $V = fv(\phi_1) \cup fv(\phi_2)$. Lemma 21 shows that ϕ_1 and ϕ_2 are also h -exact with respect to the extended variable set V , i.e., for both $i \in \{1, 2\}$:

$$h \circ sol_V^S(\phi_i) = sol_V^\Delta(\phi_i)$$

The h -exactness of the disjunction $\phi_1 \vee \phi_2$ can now be shown as follows:

$$\begin{aligned} h \circ sol^S(\phi_1 \vee \phi_2) &= h \circ (sol_V^S(\phi_1) \cup sol_V^S(\phi_2)) \\ &= h \circ sol_V^S(\phi_1) \cup h \circ sol_V^S(\phi_2) \quad \text{by Lemma 23} \\ &= sol_V^\Delta(\phi_1) \cup sol_V^\Delta(\phi_2) \quad \text{by } h\text{-exactness of } \phi_1 \text{ and } \phi_2 \text{ wrt. } V \\ &= sol^\Delta(\phi_1 \vee \phi_2) \end{aligned}$$

536

Lemma 25 Projection commutes with abstraction. For any Σ -abstraction $h : S \rightarrow \Delta$, subset R of assignments of type $V \rightarrow S$, and variable $x \in \mathcal{V}$: $h \circ \pi_x(R) = \pi_x(h \circ R)$.

Proof For all $\sigma : V \rightarrow dom(S)$ we have $h \circ \pi_x(\sigma) = h \circ \sigma|_{V \setminus \{x\}} = (h \circ \sigma)|_{V \setminus \{x\}} = \pi_x(h \circ \sigma)$.

Proposition 26 Quantification preserves exactness. For any surjective Σ -abstraction $h : S \rightarrow \Delta$ and formula $\exists x.\phi \in \mathcal{F}_\Sigma$, if ϕ is h -exact then $\exists x.\phi$ is h -exact.

Proof Let ϕ be h -exact. By definition ϕ is h -exact with respect to $V = fv(\phi)$. Since h is assumed to be surjective, Lemma 21 implies that ϕ is h -exact with respect to $V \cup \{x\}$ (independently of whether x occurs freely in ϕ or not). Hence:

$$\begin{aligned} h(sol^S(\exists x.\phi)) &= h(\pi_x(sol^S(\phi))) \quad \text{by Lemma 11} \\ &= \pi_x(h(sol^S(\phi))) \quad \text{by Lemma 25} \\ &= \pi_x(sol^\Delta(\phi)) \quad \text{since } \phi \text{ is } h\text{-exact} \\ &= sol^\Delta(\exists x.\phi) \quad \text{by Lemma 11} \end{aligned}$$

543

We next study the h -exactness for strongly-triangular systems of Σ_{bool} -equations, under the condition that h is an abstraction between Σ_{bool} -algebras with unique division (see Definition 28).

Lemma 27 Singleton property. If S is a Σ -algebra, $e \in \mathcal{E}_\Sigma(V)$, and $\sigma : V \rightarrow S$ a variable assignment, then the set $\llbracket e \rrbracket^{\sigma, S}$ is a singleton.

Proof By induction on the structure of expressions $e \in \mathcal{E}_\Sigma(V)$:

Case of constants $c \in C$. The set $\llbracket c \rrbracket^{\sigma, S} = \{c^S\}$ is a singleton.

Case of variables $x \in V$. The set $\llbracket x \rrbracket^{\sigma, S} = \{\sigma(x)\}$ is a singleton.

Case $f(e_1, \dots, e_n)$ where $e_i \in \mathcal{E}_\Sigma(V)$ and $f \in F^{(n)}$.

$$\llbracket f(e_1, \dots, e_n) \rrbracket^{\sigma, S} = \{f^S(s_1, \dots, s_n) \mid s_i \in \llbracket e_i \rrbracket^{\sigma, S}\}$$

This set is a singleton since $\llbracket e_i \rrbracket^{\sigma, S}$ are singletons by induction hypothesis, meaning that $f^S(\llbracket e_1 \rrbracket^{\sigma, S}, \dots, \llbracket e_n \rrbracket^{\sigma, S})$ is also a singleton since S is a Σ -algebra.

A Σ -algebra is a Σ -structure with the singleton property. Let ele be the function that maps any singleton to the element that it contains.

Definition 28. We say that a Σ_{bool} -structure S has unique division, if it satisfies the first-order formula $\forall x.\exists^{=1}y.ny \stackrel{\circ}{=} x$ for all nonzero natural numbers $n \in \mathbb{N}$.

557

558 Clearly, the Σ_{bool} -structures \mathbb{R}_+ , \mathbb{B} , and \mathbb{S} have unique division. Note however,
 559 that \mathbb{S} is not a Σ_{bool} -algebra, so that the following two Propositions 30 and 33 cannot be
 560 applied to \mathbb{S} instead of \mathbb{B} .

561 For any element s of the domain of a Σ_{bool} -structure S with unique division and
 562 any nonzero natural number $n \in \mathbb{N}$, we denote by $\frac{s}{n}$ the unique element of $\{\sigma(y) \mid \sigma \in$
 563 $sol^S(ny \overset{\circ}{=} z), \sigma(z) = s\}$.

Lemma 29. *Let $\phi \in \mathcal{F}_{\Sigma_{bool}}$ be a formula and S a Σ_{bool} -algebra with unique division. For nonzero natural number n , variable $y \notin fv(\phi)$, and expression $e \in \mathcal{E}_{\Sigma}(fv(\phi))$:*

$$sol^S(\phi \wedge ny \overset{\circ}{=} e) = \{\sigma[y / \frac{ele(\llbracket e \rrbracket^{\sigma, S})}{n}] \mid \sigma \in sol^S(\phi)\}$$

564 **Proof** We fix some $\sigma : fv(\phi) \rightarrow dom(S)$ arbitrarily. Since S is a Σ_{bool} -algebra, $\llbracket e \rrbracket^{\sigma, S}$
 565 is a singleton and $fv(e) \subseteq V(\phi)$, $ele(\llbracket e \rrbracket^{\sigma, S})$ is defined uniquely. Furthermore S has
 566 unique division, so that $\frac{ele(\llbracket e \rrbracket^{\sigma, S})}{n}$ is well defined element of $dom(S)$. Therefore and since
 567 $y \notin fv(\phi)$, $\sigma[y / \frac{ele(\llbracket e \rrbracket^{\sigma, S})}{n}]$ is the unique solution of the equation $ny \overset{\circ}{=} e$ that extends on σ .

568 First we prove the inclusion " \supseteq ". Let $\sigma \in sol^S(\phi)$, $y \notin fv(\phi)$, and $\sigma[y / \frac{ele(\llbracket e \rrbracket^{\sigma, S})}{n}]$ is a
 569 solution of $ny \overset{\circ}{=} e$, it follows that $\sigma[y / \frac{ele(\llbracket e \rrbracket^{\sigma, S})}{n}]$ is a solution of $\phi \wedge ny \overset{\circ}{=} e$.

570 Second, we prove the inverse inclusion " \subseteq ". Let $\sigma \in sol^S(\phi \wedge ny \overset{\circ}{=} e)$. Since
 571 $\sigma[y / \frac{ele(\llbracket e \rrbracket^{\sigma, S})}{n}]$ is the unique solution of the equation $ny \overset{\circ}{=} e$ that extends on $\sigma' = \sigma|_{fv(\phi)}$ it
 572 follows that $\sigma(y) = \frac{ele(\llbracket e \rrbracket^{\sigma, S})}{n}$ so that $\sigma = \sigma'[y / \frac{ele(\llbracket e \rrbracket^{\sigma, S})}{n}]$ while $\sigma' \in sol^S(\phi)$.

573 **Proposition 30.** *Let $\phi \in \mathcal{F}_{\Sigma_{bool}}(V)$ a formula, $n \neq 0$ a natural number, $e \in \mathcal{E}_{\Sigma_{bool}}(V)$ an
 574 expression, $y \notin V$, and $h : S \rightarrow \Delta$ a Σ_{bool} -abstraction between Σ_{bool} -algebras with unique
 575 division. Under these conditions, if ϕ is h -exact then $\phi \wedge e \overset{\circ}{=} ny$ is h -exact.*

576 **Proof** Let $e \in \mathcal{E}_{\Sigma_{bool}}(V)$ an expression.

577 **Claim 31.** *For any $\sigma : V \rightarrow \mathbb{R}_+$: $h(ele(\llbracket e \rrbracket^{\sigma, S})) = ele(\llbracket e \rrbracket^{h \circ \sigma, \Delta})$.*

578 This can be seen as follows. For any $\sigma : V \rightarrow S$ Theorem 1 on homomorphism
 579 yields $h(\llbracket e \rrbracket^{\sigma, S}) \subseteq \llbracket e \rrbracket^{h \circ \sigma, \Delta}$. Since S and Δ are both Σ -algebras, the sets $\llbracket e \rrbracket^{\sigma, S}$ and $\llbracket e \rrbracket^{h \circ \sigma, \Delta}$
 580 are both singletons by Lemma 27, so that $h(ele(\llbracket e \rrbracket^{\sigma, S})) = ele(\llbracket e \rrbracket^{h \circ \sigma, \Delta})$.

581 **Claim 32.** *For any $s \in dom(S)$ and $n \neq 0$ a natural number: $h(\frac{s}{n}) = \frac{h(s)}{n}$.*

582 Since S is assumed to have unique division $s' = \frac{s}{n}$ is well-defined as the unique
 583 element of $dom(S)$ such that $\underbrace{s' +^S \dots +^S s'}_n = s$. Hence, $h(\underbrace{s' +^S \dots +^S s'}_n) = h(s)$ and
 584 since h is a homomorphism, it follows that $\underbrace{h(s') +^\Delta \dots +^\Delta h(s')}_n = h(s)$. Since Δ is
 585 assumed to have unique division, this implies that $h(s') = \frac{h(s)}{n}$.

The Proposition can now be shown based on these two claims. Let ϕ be h -exact, $y \notin V$, and $fv(e) \subseteq V$. We have to show that $\phi \wedge ny \stackrel{\circ}{=} e$ is h -exact too:

$$\begin{aligned}
h \circ sol^S(\phi \wedge e \stackrel{\circ}{=} ny) &= h \circ \{ \sigma[y / \frac{ele(\llbracket e \rrbracket^{\sigma,S})}{n}] \mid \sigma \in sol^S(\phi) \} && \text{by Lemma 29} \\
&= \{ (h \circ \sigma)[y / h(\frac{ele(\llbracket e \rrbracket^{\sigma,S})}{n})] \mid \sigma \in sol^S(\phi) \} && \text{elementary} \\
&= \{ \sigma[y / h(\frac{ele(\llbracket e \rrbracket^{\sigma,S})}{n})] \mid \sigma \in sol^\Delta(\phi) \} && h\text{-exactness of } \phi \\
&= \{ \sigma[y / \frac{h(ele(\llbracket e \rrbracket^{\sigma,S}))}{n}] \mid \sigma \in sol^\Delta(\phi) \} && \text{by Claim 32} \\
&= \{ \sigma[y / \frac{ele(\llbracket e \rrbracket^{h\sigma,\Delta})}{n}] \mid \sigma \in sol^\Delta(\phi) \} && \text{by Claim 31} \\
&= sol^\Delta(\phi \wedge e \stackrel{\circ}{=} ny) && \text{by Lemma 29} \quad \square
\end{aligned}$$

586 **Proposition 33.** *Let $h : S \rightarrow \Delta$ be a Σ_{bool} -abstraction between algebras with unique division.*
587 *Then any strongly-triangular system of Σ_{bool} -equations is h -exact.*

588 **Proof** Any strongly-triangular system of equations has the form $\bigwedge_{i=1}^n e_i \stackrel{\circ}{=} n_i y_i$ where n
589 and $n_i \neq 0$ are naturals and y_i is i -fresh for all $1 \leq i \leq n$. The proof is by induction on n .
590 In the case $n = 0$, the conjunction is equal to *true* which is h -exact since $h(sol^S(\text{true})) =$
591 $sol^\Delta(\text{true})$. In the case $n > 0$, we have by induction hypothesis that $\bigwedge_{j=1}^{i-1} e_j \stackrel{\circ}{=} n_j y_j$ is
592 h -exact. Since $n_i \neq 0$ it follows from Proposition 30 that that $e_i \stackrel{\circ}{=} n_i y_i \wedge \bigwedge_{j=1}^{i-1} e_j \stackrel{\circ}{=} n_j y_j$ is
593 h -exact. \square

594 We notice that Proposition 33 remains true for triangular systems that are not
595 strongly-triangular. This will follow from results that we can only present in the next
596 section (Theorem 4 and Proposition 41), since they require an additional argument.

597 **Theorem 3 $h_{\mathbb{B}}$ -Exactness.** *Quasi-positive strongly-triangular polynomial systems are $h_{\mathbb{B}}$ -exact.*

598 **Proof** The Σ_{bool} -algebras \mathbb{R}_+ and \mathbb{B} have unique division, so we can apply Proposition
599 33 for proving the theorem. \square

We note that the analogous statement for \mathbb{S} instead of \mathbb{B} fails, even though \mathbb{S} has unique division. The problem is that \mathbb{S} is not a Σ_{bool} -algebra. As a counter-example, reconsider the strongly-triangular system of quasi-positive system equations:

$$u + v \stackrel{\circ}{=} x \wedge u + v \stackrel{\circ}{=} y$$

600 This system implies $x \stackrel{\circ}{=} y$ over \mathbb{R} but accept the abstract solution $[u/1, v/-1, x/1, y/-1]$
601 mapping x and y to distinct signs, so it is not $h_{\mathbb{S}}$ -exact. Nevertheless it is $h_{\mathbb{B}}$ -exact by
602 Theorem 3.

603 **Corollary 2 $h_{\mathbb{B}}$ -exact rewriting of linear equation systems.** *For any linear Σ_{bool} -equations*
604 *ϕ the elementary mode rewriting $emr(\phi) \in \mathcal{F}_{\Sigma_{bool}}$ is \mathbb{R}_+ -equivalent, $h_{\mathbb{B}}$ -exact, and can be*
605 *computed in at most exponential time from ϕ .*

606 **Proof** The elementary modes rewriting Corollary 1 shows that any linear Σ_{bool} -equation
607 system ϕ is \mathbb{R}_+ -equivalent a formula $emr(\phi)$ of the form $\exists \mathbf{z}.\phi'$ such that ϕ' is a quasi-
608 positive strongly-triangular linear equation system. Theorem 3 shows that any quasi-
609 positive strongly-triangular linear equation system is $h_{\mathbb{B}}$ -exact, so is ϕ' . Existential
610 quantification preserves $h_{\mathbb{B}}$ -exactness by Proposition 26, so $emr(\phi)$ is $h_{\mathbb{B}}$ -exact too. \square

611 This $h_{\mathbb{B}}$ -exact rewriting permits us to compute the boolean abstraction of any system
612 of linear Σ_{bool} -equations by computing the \mathbb{B} -solutions of the \mathbb{R}_+ -equivalent $h_{\mathbb{B}}$ -exact
613 formula. The latter can be done by finite domain constraint programming.

614 Our objective to find an algorithm for computing the sign abstraction of a system of
615 linear Σ_{bool} -equations remains open. We will finally approach it in Section 9. While the
616 idea is to use the $h_{\mathbb{B}}$ -exact rewriting algorithm, we first need to generalize it from linear

617 systems to mixed systems. This will be done in Section 8. The generalization will rely on
618 the notion of $h_{\mathbb{B}}$ -invariance that we discuss next in Section 7.

619 7. Invariance

620 A problem that we need to overcome is that conjunctions of two h -exact formulas
621 may not be h -exact. The situation changes when assuming the following notion of
622 h -invariance for at least one of the two formulas.

Definition 34 Invariance. Let $h : S \rightarrow \Delta$ be a Σ -abstraction and $V \subseteq \mathcal{V}$ a subset of variables. We call a subset R of variable assignments of type $V \rightarrow \text{dom}(S)$ h -invariant iff:

$$\forall \sigma, \sigma' : V \rightarrow \text{dom}(S). (\sigma \in R \wedge h \circ \sigma = h \circ \sigma' \implies \sigma' \in R).$$

623 We call a Σ -formula ϕ h -invariant if its solution set $\text{sol}^S(\phi)$ is.

624 The relevance of the notion of invariance for exactness of conjunctions – that we
625 will formalize in Proposition 41 – is due to the the following lemma:

626 **Lemma 35.** *If either R_1 or R_2 are h -invariant then: $h \circ (R_1 \cap R_2) = h \circ R_1 \cap h \circ R_2$.*

Proof The one inclusion is straightforward without invariance:

$$\begin{aligned} h \circ (R_1 \cap R_2) &= \{h \circ \sigma \mid \sigma \in R_1, \sigma \in R_2\} \\ &\subseteq \{h \circ \sigma \mid \sigma \in R_1\} \cap \{h \circ \sigma \mid \sigma \in R_2\} \\ &= h \circ R_1 \cap h \circ R_2 \end{aligned}$$

627 For the other inclusion, we can assume without loss of generality that R_1 is h -invariant.
628 So let $\tau \in h \circ R_1 \cap h \circ R_2$. Then there exist $\sigma_1 \in R_1$ and $\sigma_2 \in R_2$ such that $\tau = h \circ \sigma_1 =$
629 $h \circ \sigma_2$. By h -invariance of R_1 it follows that $\sigma_1 \in R_2$. So $\sigma_1 \in R_1 \cap R_2$, and hence,
630 $\tau \in h \circ (R_1 \cap R_2)$.

631 We can now present the algebraic characterization of h -invariance based on the
632 concretization function γ_h of the Galois connection of h . Recall that $R \subseteq h \circ (h \circ R)$ for
633 all subsets of concrete variable assignments R . The inverse inclusion characterizes the
634 h -invariance of R .

635 **Lemma 36 Algebraic characterization.** *Let $h : S \rightarrow \Delta$ be a Σ -abstraction. A subset R of
636 concrete variable assignment $V \rightarrow \text{dom}(S)$ is h -invariant for h iff $h \circ (h \circ R) \subseteq R$.*

637 **Proof “ \implies ”.** Let R be h -invariant and $\sigma \in h \circ (h \circ R)$. Then there exists $\sigma' \in R$ such that
638 $h \circ \sigma = h \circ \sigma'$. The h -invariance of R thus implies that $\sigma \in R$.

639 “ \impliedby ”. Suppose that $h \circ (h \circ R) \subseteq R$. Let $\sigma, \sigma' : V \rightarrow \text{dom}(S)$ such that $h \circ \sigma = h \circ \sigma'$ and
640 $\sigma \in R$. We have to show that $\sigma' \in R$. From $h \circ \sigma = h \circ \sigma'$ and $\sigma \in R$ it follows that
641 $\sigma' \in h \circ (h \circ R)$ and thus $\sigma' \in R$ as required.

642 **Lemma 37 Variable extension preserves invariance.** *Let h be a surjective abstraction and
643 R a subset of functions of type $V' \rightarrow \text{dom}(S)$ and V a subset of variables disjoint from V' . If R
644 is h -invariant then $\text{ext}_V^S(R)$ is h -invariant too.*

645 **Proof** This will follow straightforwardly from the characterization of h -invariance in
646 Lemma 36 and the following two claims:

647 **Claim 38.** *If h is surjective then $h \circ \text{ext}_V^S(R) = \text{ext}_V^\Delta(h \circ R)$.*

648 This follows from $h \circ \text{ext}_V^S(R) = \{h \circ \sigma \mid \sigma \in \text{ext}_V^S(R)\} = \text{ext}_V^\Delta(\{h \circ \sigma' \mid \sigma' \in R\})$
649 where we use the surjectivity of h in the last step.

650 **Claim 39.** $h \circ \text{ext}_V^\Delta(R') = \text{ext}_V^S(h \circ R')$ for any subset R' of functions of type $V' \rightarrow \text{dom}(\Delta)$.

$$\begin{aligned}
h \circ \text{ext}_V^\Delta(R') &= \{\sigma : V \cup V' \rightarrow \text{dom}(S) \mid h \circ \sigma \in \text{ext}_V^\Delta(R')\} \\
&= \{\sigma : V \cup V' \rightarrow \text{dom}(S) \mid h \circ \sigma|_{V'} \in R'\} \\
&= \text{ext}_V^S(\{\sigma' : V' \rightarrow \text{dom}(S) \mid h \circ \sigma' \in R'\}) \\
&= \text{ext}_V^S(h \circ R')
\end{aligned}$$

651

652 **Lemma 40.** Let $h : S \rightarrow \Delta$ be a surjective Σ -abstraction, ϕ be a Σ -formula, and $V \supseteq \text{fv}(\phi)$.
653 Then the h -invariance of ϕ implies the h -invariance of $\text{sol}_V^S(\phi)$.

654 **Proof** This follows from the cylindrification Lemma 10 and that extension preserves
655 h -invariance as shown in Lemma 37.

656 **Proposition 41 Exactness is preserved by conjunction when assuming invariance.** Let
657 h be a surjective Σ -abstraction. If ϕ_1 and ϕ_2 are h -exact Σ -formulas and ϕ_1 or ϕ_2 are h -invariant
658 then the conjunction $\phi_1 \wedge \phi_2$ is h -exact.

Proof Let ϕ_1 and ϕ_2 be h -exact Σ -formulas. We assume without loss of generality that
 ϕ_1 is h -invariant. Let $V = \text{fv}(\phi_1 \wedge \phi_2)$. Since $\text{fv}(\phi_2) \subseteq V$ the set $\text{sol}_V^S(\phi_2)$ is h -invariant
too by Lemma 40. We can now show that $\phi_1 \wedge \phi_2$ is h -exact as follows:

$$\begin{aligned}
h \circ \text{sol}^S(\phi_1 \wedge \phi_2) &= h \circ (\text{sol}_V^S(\phi_1) \cap \text{sol}_V^S(\phi_2)) \\
&= h \circ \text{sol}_V^S(\phi_1) \cap h \circ \text{sol}_V^S(\phi_2) \quad \text{by Lemma 35} \\
&= \text{sol}_V^\Delta(\phi_1) \cap \text{sol}_V^\Delta(\phi_2) \quad \text{by } h\text{-exactness of } \phi_1 \text{ and } \phi_2 \text{ wrt } V \\
&= \text{sol}^\Delta(\phi_1 \wedge \phi_2)
\end{aligned}$$

659

660 Our next objective is to show that h -invariant formulas are closed under conjunction,
661 disjunction, and existential quantification. The two former closure properties rely on the
662 following two algebraic properties of abstraction decomposition.

663 **Lemma 42 Concretization γ_h preserves join and meet.** For any Σ -abstraction $h : S \rightarrow \Delta$,
664 any subsets of assignments of type $V \rightarrow \text{dom}(S)$ R_1 and R_2 and V a subset of variables:

- 665 • $h \circ (R_1 \cap R_2) = h \circ R_1 \cap h \circ R_2$.
666 • $h \circ (R_1 \cup R_2) = h \circ R_1 \cup h \circ R_2$.

667 For general Galois connections, concretization is well-known to preserve joins but
668 may not preserve meets. Still, meets are preserved for any Galois connections where the
669 the concrete and abstract domains C and A are powersets as in our setting, so that joins
670 are unions and meets intersections.

Proof The case of unions follows straightforwardly from the definitions:

$$\begin{aligned}
h \circ (R_1 \cup R_2) &= \{\sigma \mid h \circ \sigma \in R_1 \cup R_2\} \\
&= \{\sigma \mid h \circ \sigma \in R_1 \vee h \circ \sigma \in R_2\} \\
&= \{\sigma \mid h \circ \sigma \in R_1\} \cup \{\sigma \mid h \circ \sigma \in R_2\} \\
&= h \circ R_1 \cup h \circ R_2
\end{aligned}$$

The case of intersection is symmetric:

$$\begin{aligned}
h \circ (R_1 \cap R_2) &= \{\sigma \mid h \circ \sigma \in R_1 \cap R_2\} \\
&= \{\sigma \mid h \circ \sigma \in R_1 \wedge h \circ \sigma \in R_2\} \\
&= \{\sigma \mid h \circ \sigma \in R_1\} \cap \{\sigma \mid h \circ \sigma \in R_2\} \\
&= h \circ R_1 \cap h \circ R_2
\end{aligned}$$

671

672 **Lemma 43 Intersection and union preserve invariance.** Let $h : S \rightarrow \Delta$ be a Σ -abstraction.
 673 Then the intersection and union of any two h -invariant subsets R_1 and R_2 of variables assign-
 674 ments of type $V \rightarrow \text{dom}(S)$ is h -invariant.

675 **Proof** This follows from the algebraic characterization Lemma 36 for invariance, in
 676 combination with the algebraic properties of composition and decomposition given in
 677 Lemmas 23, 35, and 42.

678 **Lemma 44 Concretization γ_h commutes with projection..** $h \circ \pi_x(R) = \pi_x(h \circ R)$.

679 **Proof** For all $\sigma : V \rightarrow \text{dom}(\Delta)$ we have $h \circ \pi_x(\sigma) = h \circ \sigma|_{V \setminus \{x\}} = (h \circ \sigma)|_{V \setminus \{x\}} =$
 680 $\pi_x(h \circ \sigma)$.

681 **Proposition 45 Invariance is preserved by conjunction, disjunction, and quantifica-**
 682 **tion.** If h is a surjective abstraction then the class of h -invariant FO-formulas is closed under
 683 conjunction, disjunction, and existential quantification.

684 **Proof** Let $h : S \rightarrow \Delta$ be a Σ -abstraction.

Case of conjunction: Let ϕ_1 and ϕ_2 be h -invariant and $V = \text{fv}(\phi_1 \wedge \phi_2)$. By Lemma 40 the
 sets $\text{sol}_V^S(\phi_1)$ and $\text{sol}_V^S(\phi_2)$ are both h -invariant, and so by Lemma 43 is their intersection.
 Hence:

$$\begin{aligned} h \circ (h \circ \text{sol}^S(\phi_1 \wedge \phi_2)) &= h \circ (h \circ (\text{sol}_V^S(\phi_1) \cap \text{sol}_V^S(\phi_2))) \\ &\subseteq \text{sol}_V^S(\phi_1) \cap \text{sol}_V^S(\phi_2) && \text{by } h\text{-invariance and Lemma 36} \\ &= \text{sol}^S(\phi_1 \wedge \phi_2) \end{aligned}$$

685 By Lemma 36 in the other direction, this implies that $\phi_1 \wedge \phi_2$ is h -invariant.

686 **Case of disjunction:** Analogous to the case of conjunction.

Case of existential quantification:

$$\begin{aligned} h \circ (h \circ \text{sol}^S(\exists x.\phi_1)) &= h \circ (h \circ \pi_x(\text{sol}^S(\phi_1))) && \text{by Lemma 11} \\ &= h \circ (\pi_x(h \circ \text{sol}^S(\phi_1))) && \text{by Lemma 25} \\ &= \pi_x(h \circ (h \circ \text{sol}^S(\phi_1))) && \text{by Lemma 44} \\ &\subseteq \pi_x(\text{sol}^S(\phi_1)) && \text{by } h\text{-invariance of } \phi_1 \text{ and Lemma 36} \\ &= \text{sol}^S(\exists x.\phi_1) && \text{by Lemma 11} \end{aligned}$$

687 By Lemma 36, this implies that $\exists x.\phi_1$ is h -invariant. \square

688 We do not know whether negation preserves h -invariance in general, but for finite
 689 Δ it can be shown that if ϕ is h -exact and h -invariant, then $\neg\phi$ is h -exact and h -invariant
 690 too.

691 **Proposition 46.** Let h be a surjective Σ -abstraction. Then the class of h -exact and h -invariant
 692 Σ -formulas is closed under conjunction, disjunction and existential quantification.

693 **Proof** Closure under conjunction follows from Propositions 41 and 45, closure under
 694 disjunction from Propositions 24 and 45, and closure under existential quantification by
 695 Propositions 26 and 45.

696 **Theorem 4 $h_{\mathbb{B}}$ -invariance and $h_{\mathbb{B}}$ -exactness of polynomial equations.** Any positive
 697 polynomial equation $p \stackrel{\circ}{=} 0$ such that p has no constant term is $h_{\mathbb{B}}$ -exact and $h_{\mathbb{B}}$ -invariant.

698 **Proof** Consider a positive polynomial equation $p \stackrel{\circ}{=} 0$ such that p has no constant term
 699 and only positive coefficients. Thus p has the form $\sum_{j=1}^l n_j \prod_{k=1}^{i_j} x_{j,k}^{m_{j,k}} \stackrel{\circ}{=} 0$ where $l \geq 0$,
 700 and $n_j, i_j, m_{j,k} > 0$.

701 **Claim 47.** For both algebras $S \in \{\mathbb{B}, \mathbb{R}_+\}$: $\text{sol}^S(p \stackrel{\circ}{=} 0) = \text{sol}^S(\bigwedge_{j=1}^l \bigvee_{k=1}^{i_j} x_{j,k} \stackrel{\circ}{=} 0)$.

702 The polynomial has values zero if and only if all its monomials do, that is: $\prod_{k=1}^{i_j} x_{j,k}^{m_{j,k}} =$
 703 0 for all $1 \leq j \leq l$. Since constant terms are ruled out, we have $i_j \neq 0$. Furthermore, we
 704 assumed for all polynomials that $m_{j,k} \neq 0$. So for all $1 \leq j \leq l$ there must exist $1 \leq k \leq i_j$
 705 such that $x_{j,k} = 0$.

706 **Claim 48.** The equation $x \stackrel{\circ}{=} 0$ is $h_{\mathbb{B}}$ -exact and $h_{\mathbb{B}}$ -invariant.

707 This proof of this claim is straightforward from the definitions.

With these two claims we are now in the position to prove the Theorem 4. Since the class of $h_{\mathbb{B}}$ -exact and $h_{\mathbb{B}}$ -invariant formulas is closed under conjunction and disjunction by Proposition 46, it follows from Claim 48 that $\bigwedge_{j=1}^l \bigvee_{k=1}^{i_j} x_{j,k} \stackrel{\circ}{=} 0$ is both $h_{\mathbb{B}}$ -exact and $h_{\mathbb{B}}$ -invariant. Since this formula is equivalent over \mathbb{R}_+ to the polynomial equation by Claim 47, the $h_{\mathbb{B}}$ -invariance carries over to $p \stackrel{\circ}{=} 0$. The $h_{\mathbb{B}}$ -exactness also carries over based on the equivalence for both structures \mathbb{R}_+ and \mathbb{B} :

$$\begin{aligned} h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}_+}(p \stackrel{\circ}{=} 0) &= h_{\mathbb{B}} \circ \text{sol}_{V^+}^{\mathbb{R}_+}(\bigwedge_{j=1}^l \bigvee_{k=1}^{i_j} x_{j,k} \stackrel{\circ}{=} 0) && \text{by Claim 47 for } \mathbb{R}_+ \\ &= \text{sol}^{\mathbb{B}}(\bigwedge_{j=1}^l \bigvee_{k=1}^{i_j} x_{j,k} \stackrel{\circ}{=} 0) && \text{by } h_{\mathbb{B}} \text{ exactness} \\ &= \text{sol}^{\mathbb{B}}(p \stackrel{\circ}{=} 0) && \text{by Claim 47 for } \mathbb{B}. \quad \square \end{aligned}$$

708

709 8. $h_{\mathbb{B}}$ -Exact Rewriting of $h_{\mathbb{B}}$ -Mixed Systems

710 In this section, we lift our main result to $h_{\mathbb{B}}$ -mixed system, presenting a rewrite
 711 algorithm that makes any $h_{\mathbb{B}}$ -mixed system $h_{\mathbb{B}}$ -exact.

712 **Definition 49.** A $h_{\mathbb{B}}$ -mixed system is a formula in $\mathcal{F}_{\Sigma_{\text{bool}}}$ of the form $\exists \mathbf{z}. \phi \wedge \phi'$ where ϕ is a
 713 system of linear Σ_{bool} -equations and ϕ' a $h_{\mathbb{B}}$ -invariant and $h_{\mathbb{B}}$ -exact first-order formula.

714 Note that linear equation systems $A\mathbf{y} \stackrel{\circ}{=} \mathbf{0}$, with A an integer matrix and \mathbf{y} a
 715 sequence of pairwise distinct variables, need not to be $h_{\mathbb{B}}$ -exact, if A is not positive.
 716 However, as shown by the elementary mode rewriting Corollary 1 any linear equation
 717 systems is \mathbb{R}_+ -equivalent to some quasi-positive strongly-triangular linear system, that
 718 is $h_{\mathbb{B}}$ -exact by Theorem 3.

719 Our next objective is to rewrite formulas in order to reduce the overapproximation
 720 coming with the abstract interpretation over the Booleans by John's theorem. The idea is
 721 to make a linear equation system $h_{\mathbb{B}}$ -exact that are used as subformulas as for instance
 722 of $h_{\mathbb{B}}$ -mixed systems.

We recall from Corollary 1 that the elementary mode rewriting $\text{emr}(\phi)$ of a linear equation system is an $h_{\mathbb{B}}$ -exact formula that is \mathbb{R}_+ -equivalent to ϕ . We now introduce the boolean rewriting by lifting the elementary mode rewriting to a richer class of formulas. Given a vector $\mathbf{z} \in \mathcal{V}^*$, a linear equation system $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}$, and a formula $\phi' \in \mathcal{F}_{\Sigma_{\text{bool}}}$, the boolean rewriting is defined by:

$$\text{br}(\exists \mathbf{z}. (\phi \wedge \phi')) =_{\text{def}} \exists \mathbf{z}. (\text{emr}(\phi) \wedge \phi')$$

723 The boolean rewriting may indeed reduce the overapproximation coming with abstract
724 interpretation of formulas over the booleans, as show by the following proposition.

725 **Proposition 50.** $h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}^+}(\psi) \subseteq \text{sol}^{\mathbb{B}}(\text{br}(\psi)) \subseteq \text{sol}^{\mathbb{B}}(\psi)$.

Proof Let ϕ be a linear equation system, $\mathbf{z} \in \mathcal{V}^*$, $\phi' \in \mathcal{F}_{\Sigma_{\text{bool}}}$ and $\psi =_{\text{def}} \exists \mathbf{z}. \phi \wedge \phi'$. Since ϕ is \mathbb{R}_+ -equivalent to $\text{emr}(\phi)$, it follows that $\text{br}(\psi)$ is \mathbb{R}_+ -equivalent to ψ . Hence, $\text{sol}^{\mathbb{R}^+}(\psi) = \text{sol}^{\mathbb{R}^+}(\text{br}(\psi))$ so that:

$$h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}^+}(\psi) = h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}^+}(\text{br}(\psi))$$

By John's theorem, we have:

$$h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}^+}(\text{br}(\psi)) \subseteq \text{sol}^{\mathbb{B}}(\text{br}(\psi))$$

Furthermore, by $h_{\mathbb{B}}$ -exactness, \mathbb{R}_+ -equivalence, and again John's theorem, we have:

$$\text{sol}^{\mathbb{B}}(\text{emr}(\phi)) = h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}^+}(\text{emr}(\phi)) = h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}^+}(\phi) \subseteq \text{sol}^{\mathbb{B}}(\phi)$$

Therefore, it follows that:

$$\text{sol}^{\mathbb{B}}(\text{br}(\psi)) \subseteq \text{sol}^{\mathbb{B}}(\psi)$$

726 In combination this yields the inclusions of the proposition. \square

727 **Theorem 5 (Main).** For any $h_{\mathbb{B}}$ -mixed system $\psi \in \mathcal{F}_{\Sigma}$ the boolean rewriting $\text{br}(\psi)$ is $h_{\mathbb{B}}$ -exact,
728 \mathbb{R}_+ -equivalent to ψ , and can be computed in at most exponential time.

729 **Proof** Let ψ be a $h_{\mathbb{B}}$ -mixed system $\exists \mathbf{x}. (\phi \wedge \phi')$. where ϕ is a linear equation system
730 and ϕ' a first-order formula that is $h_{\mathbb{B}}$ -exact and $h_{\mathbb{B}}$ -invariant. Based on the elementary
731 modes rewriting Corollary 1, the linear equation system ϕ can be transformed in at
732 most exponential time to the form $\text{emr}(\psi) = \exists \mathbf{z}. \phi''$ where ϕ'' is a quasi-positive strongly-
733 triangular system of linear equations. Such polynomial equation systems are $h_{\mathbb{B}}$ -exact
734 by Theorem 3, and so is ϕ'' . The Invariance Proposition 41 shows that the conjunction
735 $\phi'' \wedge \phi'$ is $h_{\mathbb{B}}$ -exact too, since ϕ' was assumed to be $h_{\mathbb{B}}$ -exact and $h_{\mathbb{B}}$ -invariant. The
736 $h_{\mathbb{B}}$ -exactness is preserved by existential quantification by Proposition 26, so the formula
737 $\text{br}(\psi) = \exists \mathbf{x}. \text{emr}(\phi) \wedge \phi'$ is $h_{\mathbb{B}}$ -exact too. \square

738 **Corollary 3.** The $h_{\mathbb{B}}$ -abstraction of the \mathbb{R}_+ -solution set of a $h_{\mathbb{B}}$ -mixed system ϕ , that is $h_{\mathbb{B}} \circ$
739 $\text{sol}^{\mathbb{R}^+}(\phi)$, can be computed in at most exponential time in the size of the system ϕ .

740 **Proof** Given a $h_{\mathbb{B}}$ -mixed system ϕ , we can apply Theorem 5 to compute in at most
741 exponential time a \mathbb{R}_+ -equivalent formula ϕ'' that is $h_{\mathbb{B}}$ -exact. It is then sufficient to
742 compute $\text{sol}^{\mathbb{B}}(\phi'')$ in exponential time in the size of ϕ . This can be done in the naive
743 manner, that is by evaluating the formula ϕ'' – which may be of exponential size – over
744 all possible boolean variable assignments – of which there may be exponentially many.
745 For each assignment the evaluation can be done in PSPACE and thus in exponential
746 time. The overall time required is thus a product of two exponentials, which remains
747 exponential.

748 The algorithm from the proof Corollary 3 can be improved so that it becomes
749 sufficiently efficient for practical use. For this the two steps with exponential worst case
750 complexity must be made polynomial for the particular instances. First note that the
751 computation of the elementary modes (Corollary 1) is known to be computationally
752 feasible. Various algorithms for this purpose were implemented [15,16,24,25] and applied
753 successfully to problems in systems biology [13]. The second exponential step concerns
754 the enumeration of all boolean variable assignments. This enumeration may be avoided
755 by using constraint programming techniques for computing the solution set $\text{sol}^{\mathbb{B}}(\phi'')$.

756 For those $h_{\mathbb{B}}$ -mixed systems for which both steps can be done in polynomial time, we
 757 can compute the boolean abstraction of the \mathbb{R}_+ -solution set in polynomial time too. The
 758 practical feasibility of this approach was demonstrated recently at an application to
 759 knockout prediction in systems biology [6], where previously only over-approximations
 760 could be computed.

761 9. Computing Sign Abstractions

762 We next show how to compute the sign abstraction $h_{\mathbb{S}} \circ \text{sol}^{\mathbb{R}}(\phi)$ for systems ϕ of
 763 linear Σ_{bool} -equations. In order to apply $h_{\mathbb{B}}$ -exact rewriting, we will decompose the sign
 764 abstraction into the boolean abstraction and functions definable in first-order logic.

765 9.1. Decomposition

We can decompose any real number $r \in \mathbb{R}$ into a pair of two positive numbers
 $\text{dec}(r) \in \mathbb{R}_+^2$ – negative and the positive part – as follows:

$$\text{dec}(r) =_{\text{def}} \begin{cases} (0, r) & \text{if } r \geq 0 \\ (-r, 0) & \text{if } r \leq 0 \end{cases}$$

The image of this surjective function is $\{0\} \times \mathbb{R}_+ \cup (\mathbb{R}_+ \times \{0\})$, so it has an inverse
 $\text{dec}^{-1} : (\{0\} \times \mathbb{R}_+) \cup (\mathbb{R}_+ \times \{0\}) \rightarrow \mathbb{R}$, which satisfies for all pairs (r_1, r_2) in the domain:

$$\text{dec}^{-1}(r_1, r_2) = r_2 -^{\mathbb{R}} r_1$$

766 Furthermore, recall that $h_{\mathbb{B}}^2 : \mathbb{R}_+^2 \rightarrow \mathbb{B}^2$ satisfies $h_{\mathbb{B}}^2(r_1, r_2) = (h_{\mathbb{B}}(r_1), h_{\mathbb{B}}(r_2))$.

767 **Lemma 51 Decomposition.** $h_{\mathbb{S}} = \text{dec}^{-1} \circ h_{\mathbb{B}}^2 \circ \text{dec}$

768 **Proof** If r is negative then $\text{dec}^{-1}(h_{\mathbb{B}}^2(\text{dec}(r))) = \text{dec}^{-1}(h_{\mathbb{B}}^2((-r, 0))) = \text{dec}^{-1}((h_{\mathbb{B}}(-r), 0))$
 769 $= -h_{\mathbb{B}}(-r) = h_{\mathbb{S}}(r)$. Otherwise if r is positive then $\text{dec}^{-1}(h_{\mathbb{B}}^2(\text{dec}(r))) = \text{dec}^{-1}(h_{\mathbb{B}}^2((0, r)))$
 770 $= \text{dec}^{-1}((0, h_{\mathbb{B}}(r))) = h_{\mathbb{B}}(r) = h_{\mathbb{S}}(r)$. \square

771 9.2. Positivity

772 We will show in a first step that first-order formulas over the reals can be rewritten,
 773 such that interpretation over the positive reals is enough.

774 We call a formula $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}$ flat if all equations contained in ϕ have the form
 775 $x \stackrel{\circ}{=} x_1 + x_2$, $x \stackrel{\circ}{=} x_1 * x_2$, $x \stackrel{\circ}{=} 0$, or $x \stackrel{\circ}{=} 1$ for some variables x, x_1, x_2 . Note that
 776 any formula $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}$ can be converted to an equivalent flat formula in linear time
 777 by introducing fresh existentially quantified variables, so that we can assume flatness
 778 without loss of generality.

We fix two generators of fresh variable $v_{\ominus}, v_{\oplus} : \mathcal{V} \rightarrow \mathcal{V}$. For any $x \in \mathcal{V}$, the intention
 is that $v_{\oplus}(x)$ stands for the positive part of x and $v_{\ominus}(x)$ for its negative part. We will
 preserve the invariants $x = v_{\oplus}(x) - v_{\ominus}(x)$ and $v_{\oplus}(x) * v_{\ominus}(x) = 0$. Furthermore, we
 define $v : \mathcal{V} \rightarrow \mathcal{V}^2$ such that for all $x \in \mathcal{V}$:

$$v(x) =_{\text{def}} (v_{\ominus}(x), v_{\oplus}(x))$$

For any flat formula $\phi \in \mathcal{F}_{\Sigma}(V)$ we define a formula $\text{dec}_v(\phi) \in \mathcal{F}_{\Sigma}(v_{\ominus}(V) \cup v_{\oplus}(V))$
 with the variables $v_{\ominus}(x)$ and $v_{\oplus}(x)$ instead of x for all $x \in V$. Otherwise the formula
 $\widetilde{\text{dec}}_v(\phi)$ has the same meaning as over the reals than ϕ .

$$\widetilde{\text{dec}}_v(\phi) = \text{dec}_v(\phi) \wedge \bigwedge_{x \in V} v_{\oplus}(x) * v_{\ominus}(x) \stackrel{\circ}{=} 0$$

where

$$\begin{aligned}
\text{dec}_v(x \overset{\circ}{=} x_1 + x_2) &= & \text{dec}_v(x \overset{\circ}{=} x_1 * x_2) &= \\
v_{\oplus}(x) + v_{\ominus}(x_1) + v_{\ominus}(x_2) \overset{\circ}{=} & & v_{\oplus}(x) + v_{\oplus}(x_1) * v_{\ominus}(x_2) + v_{\ominus}(x_1) * v_{\oplus}(x_2) \overset{\circ}{=} \\
v_{\ominus}(x) + v_{\oplus}(x_1) + v_{\oplus}(x_2) & & v_{\ominus}(x) + v_{\oplus}(x_1) * v_{\oplus}(x_2) + v_{\ominus}(x_1) * v_{\ominus}(x_2) \\
\text{dec}_v(x \overset{\circ}{=} 0) = v_{\oplus}(x) \overset{\circ}{=} v_{\ominus}(x) & & \text{dec}_v(x \overset{\circ}{=} 1) = v_{\oplus}(x) \overset{\circ}{=} v_{\ominus}(x) + 1 \\
\text{dec}_v(\exists x.\phi) = \exists v_{\ominus}(x).\exists v_{\oplus}(x). & & \text{dec}_v(\phi \wedge \phi') = \text{dec}_v(\phi) \wedge \text{dec}_v(\phi') \\
v_{\oplus}(x) * v_{\ominus}(x) \overset{\circ}{=} 0 \wedge \text{dec}_v(\phi) & & \text{dec}_v(\neg\phi) = \neg\text{dec}_v(\phi)
\end{aligned}$$

779 Note that the definition in the case of addition, the definition relies on that subtraction
780 $-^{\mathbb{R}}$ in the structure of reals is the inverse of addition $+^{\mathbb{R}}$. The expressions that are to be
781 subtracted on one side of the equation are added to the other side instead. This is also
782 used in the case of multiplication, in combination with the distributivity law for addition
783 $+^{\mathbb{R}}$ and multiplication $*^{\mathbb{R}}$. Furthermore, $\widetilde{\text{dec}}_v(\phi)$ belongs to $\mathcal{F}_{\Sigma_{\text{bool}}}(v_{\ominus}(V) \cup v_{\oplus}(V))$ and
784 can be computed in linear time from ϕ .

Proposition 52 Positivity. For any flat formula $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}(V)$:

$$\text{dec} \circ \text{sol}_V^{\mathbb{R}}(\phi) = \{\sigma^2 \circ v_{|V} \mid \sigma \in \text{sol}^{\mathbb{R}+}(\widetilde{\text{dec}}_v(\phi))\}$$

785 **Proof** By induction on the structure of ϕ . In the first case of reals, can use that $-^{\mathbb{R}}$ is the
786 inverse of $+^{\mathbb{R}}$ and that the distributivity laws holds for $+^{\mathbb{R}}$ and $*^{\mathbb{R}}$. \square

787 **Lemma 53.** For any flat linear equation system ϕ the formula $\widetilde{\text{dec}}_v(\phi)$ is a $h_{\mathbb{B}}$ -mixed system.

788 **Proof** If ϕ is a flat linear system, then $\text{dec}_v(\phi)$ is a linear system, so that $\widetilde{\text{dec}}_v(\phi)$ is a
789 $h_{\mathbb{B}}$ -mixed system.

790 9.3. Computing Sign Abstractions

791 We now have developed all the prerequisite for computing the sign abstraction of
792 linear equation systems by using $h_{\mathbb{B}}$ -exact boolean rewriting of $h_{\mathbb{B}}$ -mixed systems.

Theorem 6. For any linear equation system $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}(V)$ the formula $\text{br}(\widetilde{\text{dec}}_v(\phi))$ can be computed in at most exponential time and satisfies:

$$h_{\mathbb{S}} \circ \text{sol}_V^{\mathbb{R}}(\phi) = \{[y/\tau(v_{\oplus}(y)) -^{\mathbb{R}} \tau(v_{\ominus}(y)) \mid y \in V] \mid \tau \in \text{sol}^{\mathbb{B}}(\text{br}(\widetilde{\text{dec}}_v(\phi)))\}$$

Proof Let $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}(V)$ be a system of linear equations. Without loss of generality, we can assume that ϕ is flat. Let: $\tilde{\phi} =_{\text{def}} \widetilde{\text{dec}}_v(\phi)$. The formula $\tilde{\phi}$ is a $h_{\mathbb{B}}$ -mixed system by Lemma 53 with $\text{fv}(\tilde{\phi}) = v_{\ominus}(V) \cup v_{\oplus}(V)$ so that we can apply the Main Theorem 5 to it. It shows that boolean rewriting $\text{br}(\tilde{\phi})$ is an \mathbb{R}_+ -equivalent formula in $\mathcal{F}_{\Sigma}(v_{\oplus}(V) \cup v_{\ominus}(V))$ that is $h_{\mathbb{B}}$ -exact and can be computed in at most exponential time. We can now conclude as follows:

$$\begin{aligned}
& h_{\mathbb{S}} \circ \text{sol}_V^{\mathbb{R}}(\phi) \\
&= \text{dec}^{-1} \circ h_{\mathbb{B}}^2 \circ \text{dec} \circ \text{sol}_V^{\mathbb{R}}(\phi) && \text{Decomposition Lemma 51} \\
&= \text{dec}^{-1} \circ h_{\mathbb{B}}^2 \circ \{\sigma^2 \circ v_{|V} \mid \sigma \in \text{sol}^{\mathbb{R}+}(\tilde{\phi})\} && \text{Positivity Proposition 52} \\
&= \text{dec}^{-1} \circ h_{\mathbb{B}}^2 \circ \{\sigma^2 \circ v_{|V} \mid \sigma \in \text{sol}^{\mathbb{R}+}(\text{br}(\tilde{\phi}))\} && \mathbb{R}_+\text{-equivalence of } \tilde{\phi} \text{ and } \text{br}(\tilde{\phi}) \\
&= \{\text{dec}^{-1} \circ \tau^2 \circ v_{|V} \mid \tau \in \text{sol}^{\mathbb{B}}(\text{br}(\tilde{\phi}))\} && h_{\mathbb{B}}\text{-exactness of } \text{br}(\tilde{\phi}) \\
&= \{[y/\tau(v_{\oplus}(y)) -^{\mathbb{R}} \tau(v_{\ominus}(y)) \mid y \in V] \mid \tau \in \text{sol}^{\mathbb{B}}(\text{br}(\tilde{\phi}))\} && \text{definition of } \text{dec}^{-1}
\end{aligned}$$

793

794 The sign abstraction of a system ϕ of Σ_{bool} -equations with free variables in $V = \text{fv}(\phi)$
795 can thus be computed by first computing the $h_{\mathbb{B}}$ -exact formula $\text{br}(\tilde{\phi}) \in \mathcal{F}_{\Sigma}(v_{\oplus}(V) \cup$

```

def I(a: float, s: float):
    if a < 0: raise ValueError('This should never happen')
    if s > a:
        return 0
    else:
        return s * f(a) + I(a - s, s)

```

Figure 6. Python function approximating the integral $\int_0^a f(x)dx$ for a given function $f : \mathbb{R} \rightarrow \mathbb{R}$.

796 $v_{\ominus}(V)$ from Theorem 6 by applying the Positivity Proposition 52 and the Main Theorem
797 5, then computing $sol^{\mathbb{B}}(br(\tilde{\phi}))$ by finite domain constraint programming, and finally
798 inferring $h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\phi)$ thereof based on the equation of Theorem 6.

799 **Corollary 4.** *The sign abstraction $h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\phi)$ can be computed in at most single exponential*
800 *time in the size of ϕ .*

801 **Proof** The formula $br(\tilde{\phi})$ is of exponential size but contains only twice as many variables
802 than ϕ . Let $n = |fv(\phi)|$. We can compute $h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\phi)$ by testing 6^{2n} variable
803 assignments for membership to $sol^{\mathbb{R}}(br(\tilde{\phi}))$. Each such test is linear in the size of $br(\tilde{\phi})$
804 and thus in $O(2^m)$ where m is the size of ϕ . So the overall time is in $O(6^{2n}2^m)$ and since
805 $n \leq m$ in $O(6^{3m})$.

806 We finally show that the same algorithm as for computing the sign abstraction for
807 linear equation systems can be lifted to a richer class of formulas to obtain another and
808 possibly more precise overapproximation of the sign abstraction than John's.

Proposition 54. *Let $\psi = \exists z. \phi \wedge \phi'$ in $\mathcal{F}_{\Sigma_{bool}}(V)$ for some linear equation system ϕ and
formula $\phi' \in \mathcal{F}_{\Sigma_{bool}}$. The formula $br(\widetilde{dec}_v(\psi))$ then yields an overapproximation of the sign
abstraction of ϕ :*

$$h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\psi) \subseteq \{[y/\tau(v_{\oplus}(y)) -^{\mathbb{R}} \tau(v_{\ominus}(y)) \mid y \in V] \mid \tau \in sol^{\mathbb{B}}(br(\widetilde{dec}_v(\psi)))\}$$

809 **Proof** Along the lines of the proof of Theorem 6 except that $br(\widetilde{dec}_v(\psi))$ is not $h_{\mathbb{B}}$ -exact.
810 Therefore, the equality where the $h_{\mathbb{B}}$ -exactness was used must be weakened to an
811 inclusion.

812 10. Application to Program Analysis

813 We illustrate our results by applying the sign abstraction for program analysis
814 based on abstract interpretation. We consider the Python implementation in Fig. 6 of
815 the function $I : \mathbb{R}^2 \rightarrow \mathbb{R}$. A call $I(a, s)$ supposedly computes the approximation of
816 the integral $\int_0^a f(x)dx$ with step width s for some total function $f : \mathbb{R} \rightarrow \mathbb{R}$. Abstract
817 interpretation allows us to find out the conditions that must hold on the input parameters
818 for $I((a : float, s : float))$ to work properly, and in particular to avoid exception throwing.
819

We can first interpret numeric programs abstractly as a formula of first-order logic
with signature Σ_{arith} . We illustrate this in an ad hoc manner on the integral example I:

$$\begin{aligned}
& \exists ret_f \exists ret_I \exists result. \\
& (a < 0 \iff raise_exception \overset{\circ}{=} 1) \wedge \\
\phi_I =_{\text{def}} & ((s > a \wedge do_recursion \overset{\circ}{=} 0 \wedge result \overset{\circ}{=} 0) \vee \\
& (\neg(s > a) \wedge do_recursion \overset{\circ}{=} 1 \wedge a_{rec} \overset{\circ}{=} a - s \wedge s_{rec} \overset{\circ}{=} s \wedge \\
& result \overset{\circ}{=} s \cdot ret_f + ret_I))
\end{aligned}$$

820 The variables a and s are the formal parameters in the definition of $I(a : float, s :$
821 $float)$. The others are fresh variables introduced to handle exceptions or function

#	<i>raise_exception</i>	<i>do_recursion</i>	<i>a</i>	<i>s</i>	<i>a_{rec}</i>	<i>s_{rec}</i>
1.	0	0	0	1	-1	1
2.	0	0	1	1	0	1
3.	0	0	1	1	-1	1
4.	0	0	1	1	1	1
5.	0	1	0	0	0	0
6.	0	1	1	0	1	0
7.	0	1	0	-1	1	-1
8.	0	1	1	1	0	1
9.	0	1	1	-1	1	-1
10.	0	1	1	1	-1	1

#	<i>raise_exception</i>	<i>do_recursion</i>	<i>a</i>	<i>s</i>	<i>a_{rec}</i>	<i>s_{rec}</i>
11.	0	1	1	1	1	1
12.	1	0	-1	0	-1	0
13.	1	0	-1	-1	0	-1
14.	1	0	-1	-1	-1	-1
15.	1	0	-1	-1	1	-1
16.	1	0	-1	1	-1	1
17.	1	1	-1	-1	0	-1
18.	1	1	-1	-1	-1	-1
19.	1	1	-1	-1	1	-1

Table 1: The set of abstract solutions in $sol^{\mathbb{S}}(\phi_I)$. The 6 solutions with gray background color are unjustified since outside $h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\phi_I)$.

calls: the boolean flag *raise_exception* represents exception throwing, the boolean flag *do_recursion* has a true value only when a recursive call is made to I with actual parameters represented by the variables *a_{rec}*, *s_{rec}* and return value represented by *ret_I*, while *ret_f* is the variable for the return value of the call to the function *f*. The final return value of I is represented by the variable *result*. In what follows, we are not interested in the signs of the last three variables, so we quantify them existentially.

The sign behaviour of function I is given by the formula's sign abstraction $h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\phi_I)$. Given that ϕ_I is not $h_{\mathbb{B}}$ -mixed system, we cannot apply the algorithm from Theorem 6 directly to compute this sign abstraction. Nevertheless, it will be beneficial as we will illustrate below.

By John's theorem, the sign abstraction $h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\phi_I)$ can be overapproximated by the abstract interpretation $sol^{\mathbb{S}}(\phi_I)$. Since \mathbb{S} is a finite structure, this abstract interpretation can be computed by finite domain constraint programming. For this, we implemented a solver for first-order formulas over the structure \mathbb{S} with Minizinc [17]. When applied to ϕ_I it returns the set of abstract solutions $sol^{\mathbb{S}}(\phi_I)$ given in Table 1. This set contains the 6 unjustified abstract solutions 2, 4, 10, 13, 15, 18 outside $h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\phi_I)$. In the table they are distinguished by gray background color. We also note that the last three solutions 17, 18, 19 could be ruled out when using a more precise abstract program interpretation, taking into account that no recursive call is possible when an exception is thrown.

The sets of abstract solutions provide information on possible sign of values of the parameters in a call $I(a : float, s : float)$. For example, solution 1 in Table 1 states that when called with values of signs $[a/0, s/1]$ the function I will not raise an exceptions nor make a recursive call. Solution 8 states that when called with values of signs $[a/1, s/1]$ function I may go into recursion with signs $[a_{rec}/0, s_{rec}/1]$ without raising an exception.

Any set of abstract solutions defines an abstract call graph. The abstract call graphs of $sol^{\mathbb{S}}(\phi_I)$ and $h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\phi_I)$ from Table 1 are given in Fig. 7. Solution 1 in Table 1 implies a solid edge from the node $I^{\mathbb{S}}(1, 1)$ to the node $I^{\mathbb{S}}(0, 1)$. The edge is solid since solution 1 is justified. Edges induced by unjustified solutions are dashed. The unjustified solution 10 for instance induces the dashed edge from $I^{\mathbb{S}}(1, 1)$ to $I^{\mathbb{S}}(1, -1)$. It should be noticed that solutions with *do_recursion* = 0 and *raise_exception* = 0 do not induce any edge. Instead, they show that the computation may stop, producing final nodes that are surrounded by a double circle. The final nodes are $I^{\mathbb{S}}(1, 1)$ and $I^{\mathbb{S}}(0, 1)$. Note that for all non-final nodes, either an exception is raised or the computation loops endlessly. Solutions with *raise_exception* = 1 induce an edge to the EXCEPT node.

Given that only 2 unjustified solutions with *do_recursion* = 0 and *raise_exception* = 0 (10 and 18), there are only 2 dashed edges in the graph. Furthermore, the edges induced by the last three solutions 17, 18, 19 are drawn in blue, since these could be removed with a more precise abstract program interpretation than ϕ_I .

The sign analysis without the unjustified dashed edges yields the following result: the program in state $I^{\mathbb{S}}(1, 1)$, where $a > 0$ and $s > 0$ may either terminate, loop indefinitely, or go to state $I^{\mathbb{S}}(0, 1)$ and terminate there immediately. With the unjustified dashed edges, however, it wrongly seems possible that the program may also raise an

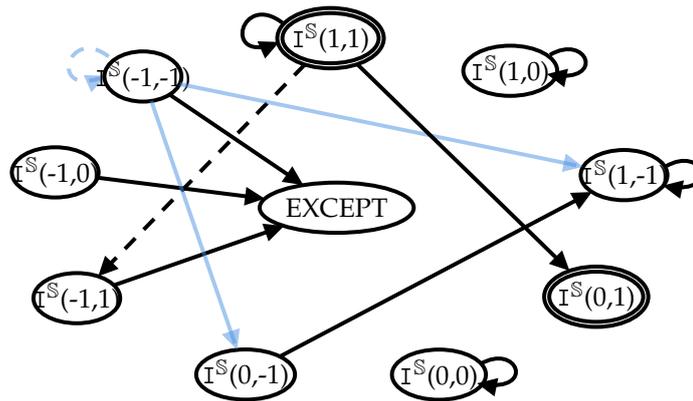


Figure 7. Sign call graph of the function I in Fig. 6 created from the sets of abstract solutions in Table 1. The solid lines correspond to abstract solutions in $h_{\mathbb{S}} \circ \text{sol}^{\text{IR}}(\phi_I)$ while dashed lines correspond to unjustified abstract solutions in $\text{sol}^{\mathbb{S}}(\phi_I)$. For example, $I^{\mathbb{S}}(1, -1)$ represents the assignment $[a/1, s/-1]$, that is the signs of a and s in calls $I(a : \text{float}, s : \text{float})$ where $a > 0$ and $s < 0$. Light blue edges may be removed by improving ϕ_I so that solutions 17, 18, 19 become impossible. The computation may terminate without raising an exception in the nodes surrounded by a double circle.

865 exception by passing through $I^{\mathbb{S}}(-1, 1)$. This overapproximation would be particularly
 866 unfortunate since state $I^{\mathbb{S}}(1, 1)$ is the only useful state to call I .

We next show how to remove the unjustified solutions by applying the overapproximation algorithm for the sign abstraction from Proposition 54, that lifts the algorithm for exact sign abstraction from Theorem 6 to a richer class of formulas. The idea is to split the formula ϕ_I into its linear part and the rest. Before doing so, we preprocess the inequation $s > a$: We introduce a fresh variable *signvar*, add the equation $s - a \stackrel{\circ}{=} \text{signvar}$, and rewrite $s > a$ to $\text{signvar} > 0$. The linear part of ϕ_I then becomes:

$$s - a \stackrel{\circ}{=} \text{signvar} \wedge a_{\text{rec}} \stackrel{\circ}{=} a - s \wedge s_{\text{rec}} \stackrel{\circ}{=} s$$

We can then rewrite the linear part into the signature Σ_{bool} by moving the negative parts positively onto the other side. This yields the following linear equation system:

$$s \stackrel{\circ}{=} \text{signvar} + a \wedge a_{\text{rec}} + s \stackrel{\circ}{=} a \wedge s_{\text{rec}} \stackrel{\circ}{=} s$$

The remainder of ϕ_I can be rewritten as follows:

$$\begin{aligned} & ((a < 0 \wedge \text{raise_exception} > 0) \vee (a \geq 0 \wedge \text{raise_exception} \stackrel{\circ}{=} 0)) \\ & \wedge ((\text{signvar} > 0 \wedge \text{do_recursion} \stackrel{\circ}{=} 0 \wedge \text{result} \stackrel{\circ}{=} 0) \vee \\ & (\text{signvar} \leq 0 \wedge \text{do_recursion} > 0 \wedge \text{result} \stackrel{\circ}{=} s * \text{ret}_{\text{f}} + \text{ret}_{\text{I}})) \end{aligned}$$

867 It is not clear whether the conjunction of both parts is a $h_{\mathbb{B}}$ -mixed system, since it is not
 868 clear how to show the $h_{\mathbb{B}}$ -invariance of the equation $\text{result} \stackrel{\circ}{=} s * \text{ret}_{\text{f}} + \text{ret}_{\text{I}}$. Still we can
 869 apply the overapproximation algorithm of the sign abstraction from Proposition 54. It
 870 indeed improves on John's approximation, ruling out both unjustified solutions. The
 871 details are worked out in Appendix 11.

872 In the general case, linear equation systems are not enough, in which case our
 873 algorithm from Theorem 6 for computing sign abstractions cannot be applied. But
 874 then we can still apply the overapproximation algorithm from Proposition 54 which
 875 rewrites a linear part of the formula exactly. As illustrated by the present example, this
 876 overapproximation is often way more precise than John's.

877 **11. Example for the Overapproximation of the Sign Abstraction**

We reconsider conjunction of the linear part obtained and the rest of ϕ_I , that is $\phi_I^{lin} \wedge \phi_I^{rest}$ where:

$$\phi_I^{lin} =_{\text{def}} \begin{cases} s \overset{\circ}{=} \text{signvar} + a \\ \wedge a_{rec} + s \overset{\circ}{=} a \\ \wedge s_{rec} \overset{\circ}{=} s \end{cases}$$

$$\phi_I^{rest} =_{\text{def}} \begin{cases} (a < 0 \wedge \text{raise_exception} > 0) \\ \vee (a \geq 0 \wedge \text{raise_exception} \overset{\circ}{=} 0) \\ \wedge ((\text{signvar} > 0 \wedge \text{do_recursion} \overset{\circ}{=} 0 \wedge \text{result} \overset{\circ}{=} 0) \\ \vee (\text{signvar} \leq 0 \wedge \text{do_recursion} > 0 \wedge \text{result} \overset{\circ}{=} s * \text{ret}_f + \text{ret}_I)) \end{cases}$$

The decomposition of the linear subsystem $\text{dec}_v(\phi_I^{lin})$ for interpretation over \mathbb{B} as defined in Section 9 is obtained by splitting each variable x into two fresh variables $v_{\oplus}(x)$ and $v_{\ominus}(x)$ representing its positive and negative part:

$$\text{dec}_v(\phi_I^{lin}) = \begin{cases} v_{\oplus}(s) + v_{\ominus}(a) + v_{\ominus}(\text{signvar}) \overset{\circ}{=} v_{\ominus}(s) + v_{\oplus}(a) + v_{\oplus}(\text{signvar}) \\ \wedge v_{\oplus}(a_{rec}) + v_{\ominus}(a) + v_{\oplus}(s) \overset{\circ}{=} v_{\ominus}(a_{rec}) + v_{\oplus}(a) + v_{\ominus}(s) \\ \wedge v_{\oplus}(s_{rec}) + v_{\ominus}(s) \overset{\circ}{=} v_{\ominus}(s_{rec}) + v_{\oplus}(s) \end{cases}$$

The additional constraints on the decomposition variables are:

$$\begin{aligned} & v_{\oplus}(s) * v_{\ominus}(s) \overset{\circ}{=} 0 \\ \wedge & v_{\oplus}(a) * v_{\ominus}(a) \overset{\circ}{=} 0 \\ \wedge & v_{\oplus}(\text{signvar}) * v_{\ominus}(\text{signvar}) \overset{\circ}{=} 0 \\ \wedge & v_{\oplus}(a_{rec}) * v_{\ominus}(a_{rec}) \overset{\circ}{=} 0 \\ \wedge & v_{\oplus}(s_{rec}) * v_{\ominus}(s_{rec}) \overset{\circ}{=} 0 \\ \wedge & v_{\oplus}(\text{result}) * v_{\ominus}(\text{result}) \overset{\circ}{=} 0 \\ \wedge & v_{\oplus}(\text{ret}_I) * v_{\ominus}(\text{ret}_I) \overset{\circ}{=} 0 \\ \wedge & v_{\oplus}(\text{ret}_f) * v_{\ominus}(\text{ret}_f) \overset{\circ}{=} 0 \end{aligned}$$

878 The elementary mode rewriting $\text{emr}(\text{dec}_v(\phi_I^{lin}))$ is the following \mathbb{R}_+ -equivalent
879 $h_{\mathbb{B}}$ -exact Σ_{bool} -formula obtained via Corollary 1:

$$\begin{aligned} & \exists x_0 \dots \exists x_{10}. \\ \wedge & v_{\ominus}(a) \overset{\circ}{=} x_{10} + x_8 + x_9 \\ \wedge & v_{\oplus}(a) \overset{\circ}{=} x_{10} + x_6 + x_7 \\ \wedge & v_{\ominus}(a_{rec}) \overset{\circ}{=} x_4 + x_5 + x_9 \\ \wedge & v_{\oplus}(a_{rec}) \overset{\circ}{=} x_3 + x_5 + x_7 \\ \wedge & v_{\ominus}(\text{signvar}) \overset{\circ}{=} x_2 + x_3 + x_7 \\ \wedge & v_{\oplus}(\text{signvar}) \overset{\circ}{=} x_2 + x_4 + x_9 \\ \wedge & v_{\ominus}(s) \overset{\circ}{=} x_1 + x_3 + x_8 \\ \wedge & v_{\oplus}(s) \overset{\circ}{=} x_1 + x_4 + x_6 \\ \wedge & v_{\ominus}(s_{rec}) \overset{\circ}{=} x_0 + x_3 + x_8 \\ \wedge & v_{\oplus}(s_{rec}) \overset{\circ}{=} x_0 + x_4 + x_6 \end{aligned}$$

880 The nonlinear remainder also needs to be rewritten with the decomposition vari-
881 ables for interpretation over \mathbb{B} . The formula below is $\text{dec}_v(\phi_I^{lin})$ except that we simplified
882 the rewriting of inequations a bit.

$$\begin{aligned}
& (\neg v_{\ominus}(a) \stackrel{\circ}{=} 0 \wedge \neg v_{\oplus}(\text{raise_exception}) \stackrel{\circ}{=} 0) \\
& \vee (v_{\ominus}(a) \stackrel{\circ}{=} 0 \wedge v_{\ominus}(\text{raise_exception}) \stackrel{\circ}{=} 0 \wedge v_{\oplus}(\text{raise_exception}) \stackrel{\circ}{=} 0) \\
\wedge & (\neg v_{\oplus}(\text{signvar}) \stackrel{\circ}{=} 0 \wedge v_{\ominus}(\text{do_recursion}) \stackrel{\circ}{=} 0 \wedge v_{\oplus}(\text{do_recursion}) \stackrel{\circ}{=} 0 \\
& \wedge v_{\ominus}(\text{result}) \stackrel{\circ}{=} 0 \wedge v_{\oplus}(\text{result}) \stackrel{\circ}{=} 0) \\
\vee & (v_{\oplus}(\text{signvar}) \stackrel{\circ}{=} 0 \wedge \neg v_{\oplus}(\text{do_recursion}) \stackrel{\circ}{=} 0 \\
& \wedge v_{\oplus}(\text{result}) + v_{\ominus}(s) * v_{\oplus}(\text{ret}_{\mathbf{f}}) + v_{\oplus}(s) * v_{\ominus}(\text{ret}_{\mathbf{f}}) + v_{\ominus}(\text{ret}_{\mathbf{I}}) \\
& \stackrel{\circ}{=} v_{\ominus}(\text{result}) + v_{\oplus}(s) * v_{\oplus}(\text{ret}_{\mathbf{f}}) + v_{\ominus}(s) * v_{\ominus}(\text{ret}_{\mathbf{f}}) + v_{\oplus}(\text{ret}_{\mathbf{I}}))
\end{aligned}$$

For any solution τ of the conjunction of the above three blocks of formulas over the algebra of booleans \mathbb{B} we then obtain an assignment $\sigma \in h_{\mathbb{S}} \circ \text{sol}^{\mathbb{R}}(\phi_{\mathbf{I}})$ according to Theorem 6:

$$\begin{aligned}
\sigma(s) &= \tau(v_{\oplus}(s)) -^{\mathbb{R}} \tau(v_{\ominus}(s)) \\
\sigma(a) &= \tau(v_{\oplus}(a)) -^{\mathbb{R}} \tau(v_{\ominus}(a)) \\
\sigma(\text{signvar}) &= \tau(v_{\oplus}(\text{signvar})) -^{\mathbb{R}} \tau(v_{\ominus}(\text{signvar})) \\
\sigma(a_{\text{rec}}) &= \tau(v_{\oplus}(a_{\text{rec}})) -^{\mathbb{R}} \tau(v_{\ominus}(a_{\text{rec}})) \\
\sigma(s_{\text{rec}}) &= \tau(v_{\oplus}(s_{\text{rec}})) -^{\mathbb{R}} \tau(v_{\ominus}(s_{\text{rec}})) \\
\sigma(\text{result}) &= \tau(v_{\oplus}(\text{result})) -^{\mathbb{R}} \tau(v_{\ominus}(\text{result})) \\
\sigma(\text{ret}_{\mathbf{f}}) &= \tau(v_{\oplus}(\text{ret}_{\mathbf{f}})) -^{\mathbb{R}} \tau(v_{\ominus}(\text{ret}_{\mathbf{f}})) \\
\sigma(\text{ret}_{\mathbf{I}}) &= \tau(v_{\oplus}(\text{ret}_{\mathbf{I}})) -^{\mathbb{R}} \tau(v_{\ominus}(\text{ret}_{\mathbf{I}}))
\end{aligned}$$

883

884 12. Conclusion and Future Work

885 We have shown that any $h_{\mathbb{B}}$ -mixed system can be rewritten into an $h_{\mathbb{B}}$ -exact formula,
886 by computing the elementary modes of the linear subsystem. In previous work $h_{\mathbb{B}}$ -exact
887 rewriting $h_{\mathbb{B}}$ -mixed systems was applied to compute difference abstractions exactly. In
888 the present paper, we have show that $h_{\mathbb{B}}$ -exact rewriting can also be used to compute
889 sign-abstractions exactly.

890 We have illustrated the usefulness of the computation of sign abstraction for linear
891 formulas for the sign analysis of function programs. Using John's overapproximation is
892 often not good enough for such applications, since the relationships between the signs
893 of different variables are quickly lost. We have seen that elementary mode rewriting
894 yields better a better approximation of the sign abstraction even for nonlinear equation
895 systems, that may preserve these relationships.

896 The time for computing abstractions exactly strongly depends on the time needed
897 to compute the elementary modes. Some experiments were reported in [6] in the case of
898 the difference abstraction. There, one has to compute the elementary modes for a linear
899 equation system that contains two copies of the linear equation system given with the
900 input. The copying doubles the size and may increase the time for the computation of
901 the elementary modes seriously. In the application of difference abstraction to change
902 prediction of reaction networks, we observed cases where John's overapproximation
903 of the difference abstraction could be computed in circa 10 minutes, while the exact
904 computation required circa 10 hours.

905 In the future, it would we of interest to find heuristics for approximating abstrac-
906 tions of linear equation systems that reduce the computation time of the exact algorithm
907 while improving John's overapproximation in precision. In the case of difference abstrac-
908 tions, the minimal support heuristics was proposed for this purpose [6]. In the example
909 mentioned above, this heuristics could be computed in circa 10 minutes, like John's
910 overapproximation, while yielding the exact result. In general, however, the minimal
911 support heuristics is not exact.

912 Another interesting question for future work is how to compute more quantitative
913 abstractions exactly, as for instance with intervals. In this case however the structure of

914 abstract values is infinite, therefore finite domain constraint programming is no longer
 915 sufficient to compute the set of abstract solutions.

916 Appendix A

The system of linear Σ_{bool} -equations $\text{dec}_v(\phi_I^{lin})$ corresponds to the following linear integer matrix equation:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} v_{\ominus}(a) \\ v_{\oplus}(a) \\ v_{\ominus}(a_{rec}) \\ v_{\oplus}(a_{rec}) \\ v_{\ominus}(signvar) \\ v_{\oplus}(signvar) \\ v_{\ominus}(s) \\ v_{\oplus}(s) \\ v_{\ominus}(s_{rec}) \\ v_{\oplus}(s_{rec}) \end{pmatrix} \stackrel{=}{=} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The elementary mode rewriting $\text{emr}(\text{dec}_v(\phi_I^{lin}))$ corresponds to the linear integer matrix equation :

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_{10} \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix} \stackrel{=}{=} \begin{pmatrix} v_{\ominus}(a) \\ v_{\oplus}(a) \\ v_{\ominus}(a_{rec}) \\ v_{\oplus}(a_{rec}) \\ v_{\ominus}(signvar) \\ v_{\oplus}(signvar) \\ v_{\ominus}(s) \\ v_{\oplus}(s) \\ v_{\ominus}(s_{rec}) \\ v_{\oplus}(s_{rec}) \end{pmatrix}$$

917

918 References

- 919 1. Cousot, P.; Cousot, R. Systematic Design of Program Analysis Frameworks. Conference
 920 Record of the Sixth Annual ACM Symposium on Principles of Programming Languages, San
 921 Antonio, Texas, USA, January 1979; Aho, A.V.; Zilles, S.N.; Rosen, B.K., Eds. ACM Press,
 922 1979, pp. 269–282. doi:10.1145/567752.567778.
- 923 2. Paulevé, L.; Sené, S. Non-Deterministic Updates of Boolean Networks. 27th IFIP WG 1.5
 924 International Workshop on Cellular Automata and Discrete Complex Systems, AUTOMATA
 925 2021, July 12-14, 2021, Aix-Marseille University, France; Castillo-Ramirez, A.; Guillon, P.;
 926 Perrot, K., Eds. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, Vol. 90, OASlcs, pp.
 927 10:1–10:16. doi:10.4230/OASlcs.AUTOMATA.2021.10.
- 928 3. Paulevé, L. Most Permissive Reaction Networks. Preprint available at [https://loicpauleve.
 929 name/md/ak8WJ5d2TqKpmJBtP_8BaQ#](https://loicpauleve.name/md/ak8WJ5d2TqKpmJBtP_8BaQ#).
- 930 4. Cousot, P.; Halbwachs, N. Automatic Discovery of Linear Restraints Among Variables of
 931 a Program. Conference Record of the Fifth Annual ACM Symposium on Principles
 932 of Programming Languages, Tucson, Arizona, USA, January 1978; Aho, A.V.; Zilles, S.N.;
 933 Szymanski, T.G., Eds. ACM Press, 1978, pp. 84–96. doi:10.1145/512760.512770.
- 934 5. Granger, P. Static Analysis of Linear Congruence Equalities among Variables of a Program.
 935 TAPSOFT'91: Proceedings of the International Joint Conference on Theory and Practice of
 936 Software Development, Brighton, UK, April 8-12, 1991, Volume 1: Colloquium on Trees in
 937 Algebra and Programming (CAAP'91); Abramsky, S.; Maibaum, T.S.E., Eds. Springer, 1991,
 938 Vol. 493, Lecture Notes in Computer Science, pp. 169–192. doi:10.1007/3-540-53982-4_10.

- 939 6. Allart, E.; Niehren, J.; Versari, C. Computing Difference Abstractions of Linear
940 Equation Systems. *Theoretical Computer Science* **2021**. Journal extension of [26], doi:
941 <https://doi.org/10.1016/j.tcs.2021.06.030>.
- 942 7. Niehren, J.; Versari, C.; John, M.; Coutte, F.; Jacques, P. Predicting changes of reac-
943 tion networks with partial kinetic information. *Biosyst.* **2016**, *149*, 113–124. doi:
944 [10.1016/j.biosystems.2016.09.003](https://doi.org/10.1016/j.biosystems.2016.09.003).
- 945 8. John, M.; Nebut, M.; Niehren, J. Knockout Prediction for Reaction Networks with Partial
946 Kinetic Information. Verification, Model Checking, and Abstract Interpretation, 14th Interna-
947 tional Conference, VMCAI 2013, Rome, Italy, January 20–22, 2013. Proceedings; Giacobazzi,
948 R.; Berdine, J.; Mastroeni, I., Eds. Springer, 2013, Vol. 7737, *Lecture Notes in Computer Science*,
949 pp. 355–374. doi:[10.1007/978-3-642-35873-9_22](https://doi.org/10.1007/978-3-642-35873-9_22).
- 950 9. Giacobazzi, R.; Ranzato, F.; Scozzari, F. Making abstract interpretations complete. *J. ACM*
951 **2000**, *47*, 361–416. doi:[10.1145/333979.333989](https://doi.org/10.1145/333979.333989).
- 952 10. Nethercote, N.; Stuckey, P.J.; Becket, R.; Brand, S.; Duck, G.J.; Tack, G. MiniZinc: Towards a
953 Standard CP Modelling Language. Principles and Practice of Constraint Programming - CP
954 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23–27, 2007,
955 Proceedings; Bessiere, C., Ed. Springer, 2007, Vol. 4741, *Lecture Notes in Computer Science*, pp.
956 529–543. doi:[10.1007/978-3-540-74970-7_38](https://doi.org/10.1007/978-3-540-74970-7_38).
- 957 11. Motzkin, T.; Raiffa, H.; Thompson, G.; Thrall, R. The double description method. Contri-
958 butions to theory of games; Kuhn, H.; A.W.Tucker, Eds. Princeton University Press, 1953,
959 Vol. 2.
- 960 12. Fukuda, K.; Prodon, A. Double Description Method Revisited. Combinatorics and Computer
961 Science, 8th Franco-Japanese and 4th Franco-Chinese Conference, Brest, France, July 3–5,
962 1995, Selected Papers; Deza, M.; Euler, R.; Manoussakis, Y., Eds. Springer, 1995, Vol. 1120,
963 *Lecture Notes in Computer Science*, pp. 91–111. doi:[10.1007/3-540-61576-8_77](https://doi.org/10.1007/3-540-61576-8_77).
- 964 13. Gagneur, J.; Klamt, S. Computation of elementary modes: a unifying framework and the
965 new binary approach. *BMC Bioinform.* **2004**, *5*, 175. doi:[10.1186/1471-2105-5-175](https://doi.org/10.1186/1471-2105-5-175).
- 966 14. Zanghellini, D.; Ruckerbauer, D.E.; Hanscho, M.; Jungreuthmayer, C. Elementary flux
967 modes in a nutshell: Properties, calculation and applications. *Biotechnology Journal* **2013**,
968 *8*(9), 1009–1016. doi:[10.1002/biot.201200269](https://doi.org/10.1002/biot.201200269).
- 969 15. Bagnara, R.; Hill, P.M.; Zaffanella, E. The Parma Polyhedra Library: Toward a complete set
970 of numerical abstractions for the analysis and verification of hardware and software systems.
971 *Sci. Comput. Program.* **2008**, *72*, 3–21. doi:[10.1016/j.scico.2007.08.001](https://doi.org/10.1016/j.scico.2007.08.001).
- 972 16. Fukuda, K. cddlib – An efficient implementation of the Double Description Method, 2018.
973 Available at <https://github.com/cddlib/cddlib>.
- 974 17. Rendl, A.; Guns, T.; Stuckey, P.J.; Tack, G. MiniSearch: A Solver-Independent Meta-Search
975 Language for MiniZinc. Principles and Practice of Constraint Programming - 21st Inter-
976 national Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings;
977 Pesant, G., Ed. Springer, 2015, Vol. 9255, *Lecture Notes in Computer Science*, pp. 376–392. doi:
978 [10.1007/978-3-319-23219-5_27](https://doi.org/10.1007/978-3-319-23219-5_27).
- 979 18. Allart, E.; Niehren, J.; Versari, C. Reaction Networks to Boolean Networks. Preprint available
980 at <https://hal.archives-ouvertes.fr/hal-02279942>.
- 981 19. Dines, L.L. On Positive Solutions of a System of Linear Equations. *Annals of Mathematics*
982 **1926**, *28*, 386–392.
- 983 20. Miné, A. A Few Graph-Based Relational Numerical Abstract Domains. Static Analysis, 9th
984 International Symposium, SAS 2002, Madrid, Spain, September 17–20, 2002, Proceedings;
985 Hermenegildo, M.V.; Puebla, G., Eds. Springer, 2002, Vol. 2477, *Lecture Notes in Computer*
986 *Science*, pp. 117–132. doi:[10.1007/3-540-45789-5_11](https://doi.org/10.1007/3-540-45789-5_11).
- 987 21. Cousot, P.; Cousot, R. Static determination of dynamic properties of programs. Proceedings
988 of the Second International Symposium on Programming. Dunod, Paris, France, 1976, pp.
989 106–130.
- 990 22. Granger, P. Static analysis of arithmetical congruences. *International Journal of Com-*
991 *puter Mathematics* **1989**, *30*, 165–190, [<https://doi.org/10.1080/00207168908803778>]. doi:
992 [10.1080/00207168908803778](https://doi.org/10.1080/00207168908803778).
- 993 23. Karr, M. Affine relationships among variables of a program. *Acta Informatica* **1976**, *6*, 133–151.
- 994 24. Klamt, S.; Stelling, J.; Ginkel, M.; Gilles, E.D. FluxAnalyzer: exploring structure,
995 pathways, and flux distributions in metabolic networks on interactive flux maps.
996 *Bioinformatics* **2003**, *19*, 261–269, [[https://academic.oup.com/bioinformatics/article-](https://academic.oup.com/bioinformatics/article-pdf/19/2/261/1059937/190261.pdf)
997 [pdf/19/2/261/1059937/190261.pdf](https://academic.oup.com/bioinformatics/article-pdf/19/2/261/1059937/190261.pdf)]. doi:[10.1093/bioinformatics/19.2.261](https://doi.org/10.1093/bioinformatics/19.2.261).

-
- 998 25. Avis, D.; Jordan, C. mplrs: A scalable parallel vertex/facet enumeration code. *Math. Program.*
999 *Comput.* **2018**, *10*, 267–302. doi:10.1007/s12532-017-0129-y.
- 1000 26. Allart, E.; Versari, C.; Niehren, J. Computing Difference Abstractions of Metabolic Networks
1001 Under Kinetic Constraints. CMSB 2019 - 17th International Conference on Computational
1002 Methods in Systems Biology; Luca Bortolussi and Guido Sanguinetti, Springer: Trieste, Italy,
1003 2019; Vol. 11773, *Lecture Notes in Computer Science*, pp. 266–285. doi:10.1007/978-3-030-31304-
1004 3_14.

Conflicts of Interest: The authors declare no conflict of interest.