



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. hal-03384058v1

HAL Id: hal-03384058

<https://inria.hal.science/hal-03384058v1>

Submitted on 18 Oct 2021 (v1), last revised 20 Oct 2021 (v2)

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 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 $x + y \stackrel{\circ}{=} x + z$. Here we write $\stackrel{\circ}{=}$ 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 $y \stackrel{\circ}{=} z$, 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 $y \stackrel{\circ}{=} z$. 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
 71 of the cone $\text{sol}^{\mathbb{R}_+}(\phi_0)$. Up to normalization, there are only finitely many extreme rays,
 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 (Theorem ??), we can rewrite any (homogeneous)
 76 linear equation system into quasi-positive and strongly-triangular linear equation system

77 that is equivalent over \mathbb{R}_+ (Corollary 21), that can be computed in at most exponential
 78 time. As we will prove, such systems are always $h_{\mathbb{B}}$ -exact (Theorem 3). Hence, any
 79 system of linear equations can be converted in at most exponential time to some \mathbb{R}_+ -
 80 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. For this we used a library from computational
 116 geometry [15] for computing elementary modes. We also used finite domain constraint
 117 programming with Minizinc [16] for computing the set of boolean solutions over logical
 118 formulas. Some successful experiments are mentioned in the related work subsection
 119 below. We did not yet implemented the algorithm for computing sign abstractions, nor
 120 its application to program analysis though.

121 Related Work

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

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

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

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

141 *Abstracting Reaction Networks to Boolean Networks.* Independently, the authors proposed an
 142 abstraction of chemical reaction networks to boolean networks in [17], whose precision
 143 can be improved by using the $h_{\mathbb{B}}$ -exact rewriting of $h_{\mathbb{B}}$ -mixed equation systems.

144 *Alternative Algorithm for Computing Sign Abstractions.* An alternative algorithm for com-
 145 puting the sign abstraction of linear equation systems (and thus also the boolean ab-
 146 straction) can be obtained by John's overapproximation theorem [6]. It shows that it
 147 is sufficient to generate the finitely many abstract solutions in $\tau \in \text{sol}^{\mathbb{S}}(\phi)$ and then to
 148 check for each of them whether there exists a concrete solution σ such that $\tau = h_{\mathbb{S}} \circ \sigma$. In
 149 order to perform this latter test, note that $h_{\Delta}(x) \stackrel{\circ}{=} 1$ is equivalent to the strict inequation
 150 $x > 0$ 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
 151 by the strict inequation $x < 0$. Whether there exists a concrete solution $\sigma \in \text{sol}^{\mathbb{R}}(\phi)$
 152 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
 153 \mathbb{R} , where $\text{fv}(\phi)$ is the set of free variables of ϕ . The satisfiability of systems of strict
 154 linear inequations and homogeneous linear equations without constant terms over \mathbb{R}
 155 are known to be decidable since at least 1926 [18]. But still one has to generate the set
 156 of all abstract solutions $\text{sol}^{\mathbb{S}}(\phi)$. The new algorithm presented above avoids generating
 157 this set.

158 *Abstract Program Interpretation over Numerical Domains.* In abstract interpretation [19],
 159 non relational domains permit to approximate the set of values of program variables
 160 while ignoring the relationship to the values of others. Well-known non-relational
 161 numerical domains include the interval domain [20] describing invariants of the form
 162 $\bigwedge_{i=1}^m x_i \in [r_i, r'_i]$ with reals $r_i \leq r'_i$ and the constant propagation domain for invariants of
 163 the form $\bigwedge_{i=1}^m x_i \stackrel{\circ}{=} r_i$ [21].

164 Abstract interpretation of relational domains may yield better approximations
 165 that with non relational domains, since relationships between the values of different
 166 variables can be taken into account. Well-known relational numerical domains include
 167 the polyhedral domain [4]. A polyhedron is the solution set of systems of inhomogeneous
 168 linear inequations of the form $n_1x_1 + \dots + n_mx_m \leq r$. Alternatively, the linear equality
 169 domain [22] was considered. These are defined by system of inhomogeneous linear
 170 equations $n_1x_1 + \dots + n_mx_n \stackrel{\circ}{=} r$.

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

176 Outline

177 In Section 2 we recall preliminaries on homomorphisms between Σ -structures. In
 178 Section 3 the first-order logic is recalled. John's theorem and its relation to the soundness

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

188 2. Homomorphisms on Σ -Structures

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

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

195 2.1. Σ -Algebras

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

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

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

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

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

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

218 2.2. Σ -Structures

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

222 **Definition 4.** A Σ -structure $\Delta = (dom(\Delta), .^\Delta)$ consists of a set $dom(\Delta)$ and an interpretation
 223 $.^\Delta$ such that $c^\Delta \in dom(\Delta)$ for all $c \in C$ and $f^\Delta \subseteq dom(\Delta)^{n+1}$ for all $f \in F^{(n)}$.

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} .

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

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

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

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

238 **Example 6.** The set of signs $\{-1, 0, 1\} \subseteq \mathbb{R}$ can be turned into a Σ_{arith} -structure $\mathbb{S} =$
 239 $(\{-1, 0, 1\}, \mathbb{S})$ with the interpretation $+^{\mathbb{S}}, -^{\mathbb{S}}, *^{\mathbb{S}}$ and $/^{\mathbb{S}}$ given in Fig. 1. The constants
 240 are interpreted by themselves $0^{\mathbb{S}} = 0$ and $1^{\mathbb{S}} = 1$. Note that all $+^{\mathbb{S}}$ contains $(-1, 1, -1)$,
 241 $(-1, 1, 1)$ and $(-1, 1, 0)$ meaning that the sum of a strictly negative and a strictly positive real
 242 has a sign in $-1 +^{\mathbb{S}} 1$, so it may either be strictly positive, strictly negative, or zero. So \mathbb{S} is a
 243 Σ_{arith} -structure and even when restricting the signature to Σ_{bool} it remains a Σ_{bool} -structure
 244 that is not a Σ_{bool} -algebra.

245 2.3. Homomorphisms

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

248 **Definition 7.** A homomorphism between two Σ -structures S and Δ is a function $h : \text{dom}(S) \rightarrow$
 249 $\text{dom}(\Delta)$ such that for $c \in C$, $n \in \mathbb{N}$, $f \in F^{(n)}$, and $s_1, \dots, s_{n+1} \in \text{dom}(S)$:

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

252 We can convert any $n + 1$ -ary relation to a n -ary set valued functions. In this
 253 way any n -function is converted to a n -ary set valued n -functions. In other words,
 254 functions of type $D^n \rightarrow D$ are converted to functions of type $D^n \rightarrow 2^D$ where $D =$
 255 $\text{dom}(\Delta)$. In set-valued notation, the second condition on homomorphism can then be

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

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

256 rewritten equivalently as $h(f^S(s_1, \dots, s_n)) \subseteq f^\Delta(h(s_1), \dots, h(s_n))$. A homomorphism
 257 for Σ -algebras thus satisfies $h(c^S) = c^\Delta$ and for all function symbols $f \in F^{(n)}$ and
 258 $s_1, \dots, s_n \in \text{dom}(S)$ it satisfies $h(f^S(s_1, \dots, s_n)) = f^\Delta(h(s_1), \dots, h(s_n))$.

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

261 **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:

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

262 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
 263 both constants $c \in C$ we have that $h_{\mathbb{B}}(c^{\mathbb{R}_+}) = h_{\mathbb{B}}(c) = c = c^{\mathbb{B}}$.

264 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
 265 strictly negative reals $r < 0$ and $h_{\mathbb{S}}(r) = 1$ for all strictly positive reals $r > 0$.

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

267 **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
 268 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
 269 distinguish cases. If r and r' have the same sign, so $r +^{\mathbb{R}} r'$ has the same sign, so that
 270 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
 271 $h_{\mathbb{S}}(r) +^{\mathbb{S}} h_{\mathbb{S}}(r') = \mathbb{S}$ 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
 272 $/^{\mathbb{S}}$ is similar. For the constants we have $h_{\mathbb{S}}(0^{\mathbb{R}}) = 0^{\mathbb{S}}$ and $h_{\mathbb{S}}(1^{\mathbb{R}}) = 1^{\mathbb{S}}$.

273 3. First-Order Logic

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

276 3.1. Expressions

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

$$280 \quad e_1, \dots, e_n \in \mathcal{E}_{\Sigma} \quad ::= x \mid c \mid f(e_1, \dots, e_n) \quad \text{where } c \in C, n \geq 0, f \in F^{(n)}, x \in \mathcal{V}$$

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

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

288 3.2. Logic Formulas

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

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

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 $\sigma : V \rightarrow \text{dom}(S)$.

$$291 \quad \phi \in \mathcal{F}_{\Sigma} \quad ::= e \stackrel{\circ}{=} e' \mid \exists x. \phi \mid \phi \wedge \phi' \mid \neg \phi \quad \text{where } e, e' \in \mathcal{E}_{\Sigma} \text{ and } x \in \mathcal{V}$$

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

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

299 First-order formulas can be defined for providing the missing logical operators.
300 First, we can define disjunctions $\phi \vee \phi' =_{\text{def}} \neg(\neg \phi \wedge \neg \phi')$ and implications $\phi \rightarrow \phi' =_{\text{def}}$
301 $\neg \phi \vee \phi'$, and second universally quantified formulas $\forall x. \phi =_{\text{def}} \neg \exists x. \neg \phi$. Note that these
302 formulas are not negation-free (and thus John's theorem cannot be applied to them).

303 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
304 of $\phi_1 \wedge \dots \wedge \phi_n$. In the case $n = 0$ the conjunction is $true$. Fourth, for any vector of
305 variables $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_n) \in \mathcal{V}^n$ we will write $\exists \mathbf{y}. \phi$ instead of $\exists \mathbf{y}_1 \dots \exists \mathbf{y}_n. \phi$.

306 For any $V \subseteq \mathcal{V}$, the semantics of first-order formulas $\phi \in \mathcal{F}_{\Sigma}(V)$ for a Σ -structure S
307 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.
308 Note that the equality symbol $\stackrel{\circ}{=}$ is interpreted as nondisjointness, i.e., an equation $e \stackrel{\circ}{=} e'$
309 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 $\stackrel{\circ}{=}$
310 is indeed interpreted as equality of singletons. In the case of more general Σ -structures,
311 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:

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

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

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

316 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 $y \stackrel{\circ}{=} h_{\mathbb{S}}(x)$ in $\mathcal{F}_{\Sigma_{\text{bool}}}$
over \mathbb{R} with two variables $x, y \in \mathcal{V}$:

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

where:

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

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

320 3.4. Semantic Properties of Free and Bound Variables

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

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

326 **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
327 to V and any variable assignment $\sigma : \text{fv}(e) \cup V \rightarrow \text{dom}(S)$ by induction on the structure
328 of expressions. From this we can prove for all formulas $\phi \in \mathcal{F}_\Sigma$ such that $\text{fv}(\phi)$ is disjoint
329 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
330 of Σ -formulas. This implies the lemma.

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

333 **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:

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

334

335 4. Abstract Interpretation

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

344 4.1. John's Overapproximation for Σ -Abstractions

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

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

349 The boolean abstraction $h_{\mathbb{B}}$ is a Σ_{bool} -abstraction by Lemma 8. The sign abstraction
350 $h_{\mathbb{S}}$ 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 $\phi \in \mathcal{F}_\Sigma$:*

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

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

354 We only give a brief sketch of the full proof which can be found in [6]. Let $V = \text{fv}(\phi)$
355 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
356 induction on the structure of e . It then follows for any negation-free formula $\phi \in \mathcal{F}_\Sigma(V)$
357 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
358 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
359 required.

360 4.2. Exactness of Σ -Formulas for Σ -Abstractions

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

363 **Definition 13 h -Exactness.** *Let $h : S \rightarrow \Delta$ be a Σ -abstraction and $\phi \in \mathcal{F}_\Sigma(V)$ a formula. We
364 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
365 with respect to $\text{fv}(\phi)$.*

366 For instance, the linear equation system ϕ equal to $x + y \stackrel{\circ}{=} x + z$ is neither $h_{\mathbb{B}}$ -exact
367 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
368 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
369 $\tau(y) \neq \tau(z)$. The same assignment also belongs to $\text{sol}^{\mathbb{S}}(\phi)$ but not to $h_{\mathbb{S}} \circ \text{sol}^{\mathbb{R}}(\phi)$ since
370 $\tau(y) \neq \tau(z)$.

371 4.3. Soundness and Completeness of Abstract Interpretation

372 John's theorem is related to the soundness of abstract interpretation and the notion
373 of exactness to its completeness. In order to state the precise relationship, we need to
374 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 :

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

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 $R \subseteq C$:

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

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

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

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

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

4.02 4.4. Galois Connection

4.03 We finally introduce the concretization operation that corresponds to the abstraction
 4.04 of the solution set of a logic formula with respect to a Σ -abstraction, and show that the
 4.05 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 $T \in A$:

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

406 **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 $R \subseteq h \circ T$.
 407 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
 408 $h \circ R \subseteq T$.

409 5. Equation Systems, Positivity, and Triangularity

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

413 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:

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

414 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
 415 called the *coefficients*, and all $m_{j,k} \neq 0$ are natural numbers called the *exponents*. The
 416 products $\prod_{k=1}^{i_j} x_{j,k}^{m_{j,k}}$ are called the *monomials* of the polynomial.

417 **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
 418 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
 419 all its monomials are variables, i.e. $i_j = 1$ and $m^{j,1} = \dots = m^{j,i_j} = 1$ for all $1 \leq j \leq l$.

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

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

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

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

430 negatively to the left and factorizing the expressions for the different occurrences of the
431 same variable.

432 5.2. Positivity and Triangularity

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

435 **Definition 17.** A Σ_{bool} -equation is called positive if it has the form $e \stackrel{\circ}{=} 0$ and quasi-positive if
436 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
437 positive respectively quasi-positive if all its equations are.

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

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

445 **Definition 18.** We call a quasi-positive system of Σ_{bool} -equations triangular if it has the form
446 $\bigwedge_{i=1}^n e_i \stackrel{\circ}{=} n_i y_i$ such that the variables y_l are l -fresh for all $1 \leq l \leq n$, i.e., $y_l \notin \text{fv}(\bigwedge_{i=1}^{l-1} e_i \stackrel{\circ}{=} e'_i)$
447 and if $n_l \neq 0$ then $y_l \notin \text{fv}(e_l)$. We call the quasi-positive polynomial system strongly-triangular
448 if it is triangular and satisfies $n_l \neq 0$ for all $1 \leq l \leq n$.

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

455 5.3. Linear Equation Systems and Elementary Modes

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

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

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

463 **Definition 20 Elementary Modes.** An elementary mode of an integer matrix $A \in \mathbb{Z}^{n,m}$ is
464 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
465 $\sigma = [\mathbf{y}/\mathbf{n}]$ is a solution in $\text{sol}^{\mathbb{R}_+}(Ay \stackrel{\circ}{=} \mathbf{0})$ such that:

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

469 The elementary modes of a matrix A are the extreme directions of the polyhedral
470 cone $\text{sol}^{\mathbb{R}_+}(Ay \stackrel{\circ}{=} \mathbf{0})$. This implies that any solution of the linear system can be expressed
471 as a weighted sum of its elementary modes, where all the weights are non negative. Due
472 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.

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

477 We note that Theorem 2 can be lifted to matrices of rational numbers \mathbb{Q} , since
 478 any rational matrix equation $A\mathbf{x} \stackrel{\circ}{=} \mathbf{0}$ can be rewritten to a integer matrix equation
 479 $A'\mathbf{x} \stackrel{\circ}{=} \mathbf{0}$ with the same \mathbb{R}_+ -solution set (by multiplying with the natural numbers in the
 480 denominators of the rational numbers). The freely available cddlib tool in the rational
 481 mode inputs any matrix $A \in \mathbb{Q}^{n,m}$, and outputs the list of (integer) elementary modes of
 482 A ¹. From this list, we can construct the matrix E for A by aligning the elementary modes
 483 of A as the columns of E .

484 **Corollary 21 Elementary Mode Rewriting.** For any system of linear equations $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}$
 485 one can compute in at most exponential time an \mathbb{R}_+ -equivalent formula $\text{emr}(\phi) \in \mathcal{F}_{\Sigma_{\text{bool}}}$ that
 486 has the form $\exists \mathbf{x}. \phi'$ where ϕ' is quasi-positive and strongly-triangular system of equations.

487 **Proof** Any system of linear equations $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}$ can be converted into an \mathbb{R}_+ -equivalent
 488 integer matrix equation $A\mathbf{y} \stackrel{\circ}{=} \mathbf{0}$ where \mathbf{y} is a vector that contains all variables in $\text{fv}(\phi)$
 489 exactly once. Let E be a matrix of elementary modes of A from Theorem 2. The theorem
 490 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
 491 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}
 492 are pairwise distinct, and the variables in \mathbf{x} are chosen freshly, it follows that ϕ' is both
 493 quasi-positive and strongly-triangular. \square

494 We have implemented the elementary mode rewriting in Python based on the
 495 cddlib tool [?]. An example input is the system of linear Σ_{bool} -equations ϕ_0 given in Fig.
 496 4. The corresponding integer matrix equation system is given there too. The elementary
 497 modes of the matrix of this system are the vectors $(1, 0, 1, 1)$ and $(1, 1, 0, 0)$. When putting

¹ Out of the box, the interface of the cddlib tool is more general: it applies to matrix inequations over the reals, rather than to matrix equations over the positive reals. So the cddlib tool permits to compute the normalized extreme directions of polyedral cones $\text{sol}^{\mathbb{R}}(B\mathbf{x} \geq 0)$ for any rational matrix inequation B over the reals. If we want to compute the elementary modes of integer matrix equations A , that is the normalized extreme directions of polyhedral cones matrix equations over the positive reals $\text{sol}^{\mathbb{R}_+}(A\mathbf{x} \stackrel{\circ}{=} 0)$, then we can chose $B = \begin{pmatrix} A \\ -A \\ Id \end{pmatrix}$ where Id is the identity matrix with as many columns than A , since $\text{sol}^{\mathbb{R}}(B\mathbf{x} \geq 0) = \text{sol}^{\mathbb{R}}(A\mathbf{x} \geq 0 \wedge A\mathbf{x} \leq 0 \wedge \mathbf{x} \geq 0) = \text{sol}^{\mathbb{R}_+}(A\mathbf{x} \stackrel{\circ}{=} 0)$.

498 these vectors in the columns of a new matrix, our tool returns the elementary mode
499 rewriting $emr(\phi_0)$ in Fig. 5.

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

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

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

508 **Lemma 22 Variable extension preserves exactness.** *Let $h : S \rightarrow \Delta$ be a Σ -abstraction that*
509 *is surjective and $\phi \in \mathcal{F}_{\Sigma}(V)$ a formula. Then the h -exactness of ϕ implies the h -exactness of ϕ*
510 *with respect to V .*

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

513 **Claim 23.** *For all $\sigma : V \rightarrow \Delta$: $\sigma \in h \circ sol^S(\phi)$ iff $\sigma_{|fv(\phi)} \in h \circ sol^S(\phi)$.*

514 For the one direction let $\sigma \in h \circ sol_V^S(\phi)$. Then there exists $\sigma \in sol_V^S(\phi)$ such that
515 $\sigma = h \circ \sigma$. Since $V \supseteq fv(\phi)$ it follows that $\sigma_{|fv(\phi)} \in sol^S(\phi)$. Furthermore $\sigma_{|fv(\phi)} =$
516 $h \circ \sigma_{|fv(\phi)}$ and thus $\sigma_{|fv(\phi)} \in h \circ sol^S(\phi)$.

517 For the other direction let $\sigma_{|fv(\phi)} \in h \circ sol^S(\phi)$. Then there exists $\sigma \in sol^S(\phi)$ such
518 that $\sigma_{|fv(\phi)} = h \circ \sigma$. For any $y \in V \setminus fv(\phi)$ let $s_y \in dom(S)$ be such that $h(s_y) = \sigma(y)$.
519 Such values exists since h is surjective. Now define $\sigma' = \sigma[y/s_y \mid y \in V \setminus fv(\phi)]$. Since
520 $V \supseteq fv(\phi)$ it follows that $\sigma' \in sol_V^S(\phi)$. Furthermore, $\sigma = h \circ \sigma'$, so $\sigma \in h \circ sol_V^S(\phi)$. \square

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

Lemma 24 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:

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

523

524 **Proposition 25.** *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 22 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:

$$\begin{aligned}
h \circ \text{sol}^S(\phi_1 \vee \phi_2) &= h \circ (\text{sol}_V^S(\phi_1) \cup \text{sol}_V^S(\phi_2)) \\
&= h \circ \text{sol}_V^S(\phi_1) \cup h \circ \text{sol}_V^S(\phi_2) && \text{by Lemma 24} \\
&= \text{sol}_V^\Delta(\phi_1) \cup \text{sol}_V^\Delta(\phi_2) && \text{by } h\text{-exactness of } \phi_1 \text{ and } \phi_2 \text{ wrt. } V \\
&= \text{sol}^\Delta(\phi_1 \vee \phi_2)
\end{aligned}$$

525

Lemma 26 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 $\sigma : V \rightarrow \text{dom}(S)$ 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 27 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 $V = \text{fv}(\phi)$. Since h is assumed to be surjective, Lemma 22 implies that ϕ is h -exact with respect to $V \cup \{x\}$ (independently of whether x occurs freely in ϕ or not). Hence:

$$\begin{aligned}
h(\text{sol}^S(\exists x.\phi)) &= h(\pi_x(\text{sol}^S(\phi))) && \text{by Lemma 11} \\
&= \pi_x(h(\text{sol}^S(\phi))) && \text{by Lemma 26} \\
&= \pi_x(\text{sol}^\Delta(\phi)) && \text{since } \phi \text{ is } h\text{-exact} \\
&= \text{sol}^\Delta(\exists x.\phi) && \text{by Lemma 11}
\end{aligned}$$

532

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 29).

Lemma 28 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 29. We say that a Σ_{bool} -structure S has unique division, if it satisfies the first-order formula $\forall x.\exists=^1 y. ny \overset{\circ}{=} x$ for all nonzero natural numbers $n \in \mathbb{N}$.

Clearly, the Σ_{bool} -structures \mathbb{R}_+ , \mathbb{B} , and \mathbb{S} have unique division. Note however, that \mathbb{S} is not a Σ_{bool} -algebra, so that the following two Propositions 31 and 34 cannot be applied to \mathbb{S} instead of \mathbb{B} .

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

Lemma 30. Let $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}$ be a formula and S a Σ_{bool} -algebra with unique division. For nonzero natural number n , variable $y \notin \text{fv}(\phi)$, and expression $e \in \mathcal{E}_{\Sigma}(\text{fv}(\phi))$:

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

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

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

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

Proposition 31. Let $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}(V)$ a formula, $n \neq 0$ a natural number, $e \in \mathcal{E}_{\Sigma_{\text{bool}}}(V)$ an expression, $y \notin V$, and $h : S \rightarrow \Delta$ a Σ_{bool} -abstraction between Σ_{bool} -algebras with unique division. Under these conditions, if ϕ is h -exact then $\phi \wedge e \stackrel{\circ}{=} ny$ is h -exact.

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

Claim 32. For any $\sigma : V \rightarrow \mathbb{R}_+$: $h(\text{ele}(\llbracket e \rrbracket^{\sigma, S})) = \text{ele}(\llbracket e \rrbracket^{h \circ \sigma, \Delta})$.

This can be seen as follows. For any $\sigma : V \rightarrow S$ Theorem 1 on homomorphism 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 $\llbracket e \rrbracket^{h \circ \sigma, \Delta}$ are both singletons by Lemma 28, so that $h(\text{ele}(\llbracket e \rrbracket^{\sigma, S})) = \text{ele}(\llbracket e \rrbracket^{h \circ \sigma, \Delta})$.

Claim 33. For any $s \in \text{dom}(S)$ and $n \neq 0$ a natural number: $h\left(\frac{s}{n}\right) = \frac{h(s)}{n}$.

Since S is assumed to have unique division $s' = \frac{s}{n}$ is well-defined as the unique element of $\text{dom}(S)$ such that $\underbrace{s' +^S \dots +^S s'}_n = s$. Hence, $h\left(\underbrace{s' +^S \dots +^S s'}_n\right) = h(s)$ and since h is a homomorphism, it follows that $\underbrace{h(s') +^\Delta \dots +^\Delta h(s')}_n = h(s)$. Since Δ is 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 $\text{fv}(e) \subseteq V$. We have to show that $\phi \wedge ny \stackrel{\circ}{=} e$ is h -exact too:

$$\begin{aligned} h \circ \text{sol}^S(\phi \wedge e \stackrel{\circ}{=} ny) &= h \circ \left\{ \sigma \left[y / \frac{\text{ele}(\llbracket e \rrbracket^{\sigma, S})}{n} \right] \mid \sigma \in \text{sol}^S(\phi) \right\} && \text{by Lemma 30} \\ &= \left\{ (h \circ \sigma) \left[y / h\left(\frac{\text{ele}(\llbracket e \rrbracket^{\sigma, S})}{n}\right) \right] \mid \sigma \in \text{sol}^S(\phi) \right\} && \text{elementary} \\ &= \left\{ \sigma \left[y / h\left(\frac{\text{ele}(\llbracket e \rrbracket^{\sigma, S})}{n}\right) \right] \mid \sigma \in \text{sol}^\Delta(\phi) \right\} && h\text{-exactness of } \phi \\ &= \left\{ \sigma \left[y / \frac{h(\text{ele}(\llbracket e \rrbracket^{\sigma, S}))}{n} \right] \mid \sigma \in \text{sol}^\Delta(\phi) \right\} && \text{by Claim 33} \\ &= \left\{ \sigma \left[y / \frac{\text{ele}(\llbracket e \rrbracket^{h \circ \sigma, \Delta})}{n} \right] \mid \sigma \in \text{sol}^\Delta(\phi) \right\} && \text{by Claim 32} \\ &= \text{sol}^\Delta(\phi \wedge e \stackrel{\circ}{=} ny) && \text{by Lemma 30} \quad \square \end{aligned}$$

Proposition 34. Let $h : S \rightarrow \Delta$ be a Σ_{bool} -abstraction between algebras with unique division. Then any strongly-triangular system of Σ_{bool} -equations is h -exact.

Proof Any strongly-triangular system of equations has the form $\bigwedge_{i=1}^n e_i \stackrel{\circ}{=} n_i y_i$ where n 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 .

579 In the case $n = 0$, the conjunction is equal to *true* which is h -exact since $h(\text{sol}^S(\text{true})) =$
 580 $\text{sol}^\Delta(\text{true})$. 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
 581 h -exact. Since $n_i \neq 0$ it follows from Proposition 31 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
 582 h -exact. \square

583 We notice that Proposition 34 remains true for triangular systems that are not
 584 strongly-triangular. This will follow from results that we can only present in the next
 585 section (Theorem 4 and Proposition 43), since they require an additional argument.

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

587 **Proof** The Σ_{bool} -algebras \mathbb{R}_+ and \mathbb{B} have unique division, so we can apply Proposition
 588 34 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:

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

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

592 **Corollary 35 $h_{\mathbb{B}}$ -exact rewriting of linear equation systems.** *For any linear Σ_{bool} -equations*
 593 *ϕ the elementary mode rewriting $\text{emr}(\phi) \in \mathcal{F}_{\Sigma_{\text{bool}}}$ is \mathbb{R}_+ -equivalent, $h_{\mathbb{B}}$ -exact, and can be*
 594 *computed in at most exponential time from ϕ .*

595 **Proof** The elementary modes rewriting Corollary 21 shows that any linear Σ_{bool} -equation
 596 system ϕ is \mathbb{R}_+ -equivalent a formula $\text{emr}(\phi)$ of the form $\exists \mathbf{z}.\phi'$ such that ϕ' is a quasi-
 597 positive strongly-triangular linear equation system. Theorem 3 shows that any quasi-
 598 positive strongly-triangular linear equation system is $h_{\mathbb{B}}$ -exact, so is ϕ' . Existential
 599 quantification preserves $h_{\mathbb{B}}$ -exactness by Proposition 27, so $\text{emr}(\phi)$ is $h_{\mathbb{B}}$ -exact too. \square

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

603 Our objective to find an algorithm for computing the sign abstraction of a system of
 604 linear Σ_{bool} -equations remains open. We will finally approach it in Section 9. While the
 605 idea is to use the $h_{\mathbb{B}}$ -exact rewriting algorithm, we first need to generalize it from linear
 606 systems to mixed systems. This will be done in Section 8. The generalization will rely on
 607 the notion of $h_{\mathbb{B}}$ -invariance that we discuss next in Section 7.

608 7. Invariance

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

Definition 36 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:*

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

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

613 The relevance of the notion of invariance for exactness of conjunctions – that we
 614 will formalize in Proposition 43 – is due to the the following lemma:

615 **Lemma 37.** *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:

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

616 For the other inclusion, we can assume without loss of generality that R_1 is h -invariant.
617 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 $\tau = h \circ \sigma_1 =$
618 $h \circ \sigma_2$. 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,
619 $\tau \in h \circ (R_1 \cap R_2)$.

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

624 **Lemma 38 Algebraic characterization.** *Let $h : S \rightarrow \Delta$ be a Σ -abstraction. A subset R of*
625 *concrete variable assignment $V \rightarrow \text{dom}(S)$ is h -invariant for h iff $h \circ (h \circ R) \subseteq R$.*

626 **Proof “ \Rightarrow ”.** Let R be h -invariant and $\sigma \in h \circ (h \circ R)$. Then there exists $\sigma' \in R$ such that
627 $h \circ \sigma = h \circ \sigma'$. The h -invariance of R thus implies that $\sigma \in R$.

628 “ \Leftarrow ”. 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
629 $\sigma \in R$. We have to show that $\sigma' \in R$. From $h \circ \sigma = h \circ \sigma'$ and $\sigma \in R$ it follows that
630 $\sigma' \in h \circ (h \circ R)$ and thus $\sigma' \in R$ as required.

631 **Lemma 39 Variable extension preserves invariance.** *Let h be a surjective abstraction and*
632 *R a subset of functions of type $V' \rightarrow \text{dom}(S)$ and V a subset of variables disjoint from V' . If R*
633 *is h -invariant then $\text{ext}_V^S(R)$ is h -invariant too.*

634 **Proof** This will follow straightforwardly from the characterization of h -invariance in
635 Lemma 38 and the following two claims:

636 **Claim 40.** *If h is surjective then $h \circ \text{ext}_V^S(R) = \text{ext}_V^\Delta(h \circ R)$.*

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

639 **Claim 41.** *$h \circ \text{ext}_V^\Delta(R') = \text{ext}_V^S(h \circ R')$ for any subset R' of functions of type $V' \rightarrow \text{dom}(\Delta)$.*

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

640

641 **Lemma 42.** *Let $h : S \rightarrow \Delta$ be a surjective Σ -abstraction, ϕ be a Σ -formula, and $V \supseteq \text{fv}(\phi)$.*
642 *Then the h -invariance of ϕ implies the h -invariance of $\text{sol}_V^S(\phi)$.*

643 **Proof** This follows from the cylindrification Lemma 10 and that extension preserves
644 h -invariance as shown in Lemma 39.

645 **Proposition 43 Exactness is preserved by conjunction when assuming invariance.** *Let*
646 *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*
647 *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 42. We can now show that $\phi_1 \wedge \phi_2$ is h -exact as follows:

$$\begin{aligned}
h \circ \text{sol}^S(\phi_1 \wedge \phi_2) &= h \circ (\text{sol}_V^S(\phi_1) \cap \text{sol}_V^S(\phi_2)) \\
&= h \circ \text{sol}_V^S(\phi_1) \cap h \circ \text{sol}_V^S(\phi_2) && \text{by Lemma 37} \\
&= \text{sol}_V^\Delta(\phi_1) \cap \text{sol}_V^\Delta(\phi_2) && \text{by } h\text{-exactness of } \phi_1 \text{ and } \phi_2 \text{ wrt } V \\
&= \text{sol}^\Delta(\phi_1 \wedge \phi_2)
\end{aligned}$$

648

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

652 **Lemma 44 Concretization γ_h preserves join and meet.** For any Σ -abstraction $h : S \rightarrow \Delta$,
653 any subsets of assignments of type $V \rightarrow \text{dom}(S)$ R_1 and R_2 and V a subset of variables:

- 654 • $h \circ (R_1 \cap R_2) = h \circ R_1 \cap h \circ R_2$.
655 • $h \circ (R_1 \cup R_2) = h \circ R_1 \cup h \circ R_2$.

656 For general Galois connections, concretization is well-known to preserve joins but
657 may not preserve meets. Still, meets are preserved for any Galois connections where the
658 the concrete and abstract domains C and A are powersets as in our setting, so that joins
659 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:

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

660

661 **Lemma 45 Intersection and union preserve invariance.** Let $h : S \rightarrow \Delta$ be a Σ -abstraction.
662 Then the intersection and union of any two h -invariant subsets R_1 and R_2 of variables assign-
663 ments of type $V \rightarrow \text{dom}(S)$ is h -invariant.

664 **Proof** This follows from the algebraic characterization Lemma 38 for invariance, in
665 combination with the algebraic properties of composition and decomposition given in
666 Lemmas 24, 37, and 44.

667 **Lemma 46 Concretization γ_h commutes with projection..** $h \circ \pi_x(R) = \pi_x(h \circ R)$.

668 **Proof** For all $\sigma : V \rightarrow \text{dom}(\Delta)$ we have $h \circ \pi_x(\sigma) = h \circ \sigma|_{V \setminus \{x\}} = (h \circ \sigma)|_{V \setminus \{x\}} =$
669 $\pi_x(h \circ \sigma)$.

670 **Proposition 47 Invariance is preserved by conjunction, disjunction, and quantifica-**
671 **tion.** If h is a surjective abstraction then the class of h -invariant FO-formulas is closed under
672 conjunction, disjunction, and existential quantification.

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

Case of conjunction: Let ϕ_1 and ϕ_2 be h -invariant and $V = fv(\phi_1 \wedge \phi_2)$. By Lemma 42 the sets $sol_V^S(\phi_1)$ and $sol_V^S(\phi_2)$ are both h -invariant, and so by Lemma 45 is their intersection. Hence:

$$\begin{aligned} h \ominus (h \circ sol^S(\phi_1 \wedge \phi_2)) & \\ &= h \ominus (h \circ (sol_V^S(\phi_1) \cap sol_V^S(\phi_2))) \\ &\subseteq sol_V^S(\phi_1) \cap sol_V^S(\phi_2) && \text{by } h\text{-invariance and Lemma 38} \\ &= sol^S(\phi_1 \wedge \phi_2) \end{aligned}$$

674 By Lemma 38 in the other direction, this implies that $\phi_1 \wedge \phi_2$ is h -invariant.

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

Case of existential quantification:

$$\begin{aligned} h \ominus (h \circ sol^S(\exists x.\phi_1)) & \\ &= h \ominus (h \circ \pi_x(sol^S(\phi_1))) && \text{by Lemma 11} \\ &= h \ominus (\pi_x(h \circ sol^S(\phi_1))) && \text{by Lemma 26} \\ &= \pi_x(h \ominus (h \circ sol^S(\phi_1))) && \text{by Lemma 46} \\ &\subseteq \pi_x(sol^S(\phi_1)) && \text{by } h\text{-invariance of } \phi_1 \text{ and Lemma 38} \\ &= sol^S(\exists x.\phi_1) && \text{by Lemma 11} \end{aligned}$$

676 By Lemma 38, this implies that $\exists x.\phi_1$ is h -invariant. \square

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

680 **Proposition 48.** *Let h be a surjective Σ -abstraction. Then the class of h -exact and h -invariant
681 Σ -formulas is closed under conjunction, disjunction and existential quantification.*

682 **Proof** Closure under conjunction follows from Propositions 43 and 47, closure under
683 disjunction from Propositions 25 and 47, and closure under existential quantification by
684 Propositions 27 and 47.

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

687 **Proof** Consider a positive polynomial equation $p \stackrel{\circ}{=} 0$ such that p has no constant term
688 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 $l \geq 0$,
689 and $n_j, i_j, m_{j,k} > 0$.

690 **Claim 49.** *For both algebras $S \in \{\mathbb{B}, \mathbb{R}_+\}$: $sol^S(p \stackrel{\circ}{=} 0) = sol^S(\bigwedge_{j=1}^l \bigvee_{k=1}^{i_j} x_{j,k} \stackrel{\circ}{=} 0)$.*

691 The polynomial has values zero if and only if all its monomials do, that is: $\prod_{k=1}^{i_j} x_{j,k}^{m_{j,k}} =$
692 0 for all $1 \leq j \leq l$. Since constant terms are ruled out, we have $i_j \neq 0$. Furthermore, we
693 assumed for all polynomials that $m_{j,k} \neq 0$. So for all $1 \leq j \leq l$ there must exist $1 \leq k \leq i_j$
694 such that $x_{j,k} = 0$.

695 **Claim 50.** *The equation $x \stackrel{\circ}{=} 0$ is $h_{\mathbb{B}}$ -exact and $h_{\mathbb{B}}$ -invariant.*

696 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 48, it follows from by Claim 50 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 49, 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} :

$$\begin{aligned} h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}_+}(p \stackrel{\circ}{=} 0) &= h_{\mathbb{B}} \circ \text{sol}_V^{\mathbb{R}_+}(\bigwedge_{j=1}^l \bigvee_{k=1}^{i_j} x_{j,k} \stackrel{\circ}{=} 0) && \text{by Claim 49 for } \mathbb{R}_+ \\ &= \text{sol}^{\mathbb{B}}(\bigwedge_{j=1}^l \bigvee_{k=1}^{i_j} x_{j,k} \stackrel{\circ}{=} 0) && \text{by } h_{\mathbb{B}} \text{ exactness} \\ &= \text{sol}^{\mathbb{B}}(p \stackrel{\circ}{=} 0) && \text{by Claim 49 for } \mathbb{B}. \quad \square \end{aligned}$$

697

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

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

701 **Definition 51.** 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
702 system of linear Σ_{bool} -equations and ϕ' a $h_{\mathbb{B}}$ -invariant and $h_{\mathbb{B}}$ -exact first-order formula.

703 Note that linear equation systems $A\mathbf{y} \stackrel{\circ}{=} \mathbf{0}$, with A an integer matrix and \mathbf{y} a
704 sequence of pairwise distinct variables, need not to be $h_{\mathbb{B}}$ -exact, if A is not positive.
705 However, as shown by the elementary mode rewriting Corollary 21 any linear equation
706 systems is \mathbb{R}_+ -equivalent to some quasi-positive strongly-triangular linear system, that
707 is $h_{\mathbb{B}}$ -exact by Theorem 3.

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

We recall from Corollary 21 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:

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

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

714 **Proposition 52.** $h_{\mathbb{B}} \circ \text{sol}^{\mathbb{R}_+}(\psi) \subseteq \text{sol}^{\mathbb{B}}(\text{br}(\psi)) \subseteq \text{sol}^{\mathbb{B}}(\psi)$.

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:

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

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

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

718 **Proof** Let ψ be a $h_{\mathbb{B}}$ -mixed system $\exists \mathbf{x}. (\phi \wedge \phi')$, where ϕ is a linear equation system
 719 and ϕ' a first-order formula that is $h_{\mathbb{B}}$ -exact and $h_{\mathbb{B}}$ -invariant. Based on the elementary
 720 modes rewriting Corollary 21, the linear equation system ϕ can be transformed in at
 721 most exponential time to the form $emr(\psi) = \exists \mathbf{z}. \phi''$ where ϕ'' is a quasi-positive strongly-
 722 triangular system of linear equations. Such polynomial equation systems are $h_{\mathbb{B}}$ -exact
 723 by Theorem 3, and so is ϕ'' . The Invariance Proposition 43 shows that the conjunction
 724 $\phi'' \wedge \phi'$ is $h_{\mathbb{B}}$ -exact too, since ϕ' was assumed to be $h_{\mathbb{B}}$ -exact and $h_{\mathbb{B}}$ -invariant. The
 725 $h_{\mathbb{B}}$ -exactness is preserved by existential quantification by Proposition 27, so the formula
 726 $br(\psi) = \exists \mathbf{x}. emr(\phi) \wedge \phi'$ is $h_{\mathbb{B}}$ -exact too. \square

727 **Corollary 53.** *The $h_{\mathbb{B}}$ -abstraction of the \mathbb{R}_+ -solution set of a $h_{\mathbb{B}}$ -mixed system ϕ , that is*
 728 *$h_{\mathbb{B}} \circ sol^{\mathbb{R}_+}(\phi)$, can be computed in at most exponential time in the size of the system ϕ .*

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

737 The algorithm from the proof Corollary 53 can be improved so that it becomes
 738 sufficiently efficient for practical use. For this the two steps with exponential worst case
 739 complexity must be made polynomial for the particular instances. First note that the
 740 computation of the elementary modes (Corollary 21) is known to be computationally
 741 feasible. Various algorithms for this purpose were implemented [15,23–25] and applied
 742 successfully to problems in systems biology [13]. The second exponential step concerns
 743 the enumeration of all boolean variable assignments. This enumeration may be avoided
 744 by using constraint programming techniques for computing the solution set $sol^{\mathbb{B}}(\phi'')$.
 745 For those $h_{\mathbb{B}}$ -mixed systems for which both steps can be done in polynomial time, we
 746 can compute the boolean abstraction of the \mathbb{R}_+ -solution set in polynomial time too. The
 747 practical feasibility of this approach was demonstrated recently at an application to
 748 knockout prediction in systems biology [6], where previously only over-approximations
 749 could be computed.

750 9. Computing Sign Abstractions

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

754 9.1. Decomposition

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

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

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:

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

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

756 **Lemma 54 Decomposition.** $h_{\mathbb{S}} = \text{dec}^{-1} \circ h_{\mathbb{B}}^2 \circ \text{dec}$

757 **Proof** If r is negative then $\text{dec}^{-1}(h_{\mathbb{B}}^2(\text{dec}(r))) = \text{dec}^{-1}(h_{\mathbb{B}}^2((-r, 0))) = \text{dec}^{-1}((h_{\mathbb{B}}(-r), 0))$
 758 $= -h_{\mathbb{B}}(-r) = h_{\mathbb{S}}(r)$. Otherwise if r is positive then $\text{dec}^{-1}(h_{\mathbb{B}}^2(\text{dec}(r))) = \text{dec}^{-1}(h_{\mathbb{B}}^2((0, r)))$
 759 $= \text{dec}^{-1}((0, h_{\mathbb{B}}(r))) = h_{\mathbb{B}}(r) = h_{\mathbb{S}}(r)$. \square

760 9.2. Positivity

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

763 We call a formula $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}$ flat if all equations contained in ϕ have the form
 764 $x \stackrel{\circ}{=} x_1 + x_2$, $x \stackrel{\circ}{=} x_1 * x_2$, $x \stackrel{\circ}{=} 0$, or $x \stackrel{\circ}{=} 1$ for some variables x, x_1, x_2 . Note that
 765 any formula $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}$ can be converted to an equivalent flat formula in linear time
 766 by introducing fresh existentially quantified variables, so that we can assume flatness
 767 without loss of generality.

We fix two generators of fresh variable $v_{\oplus}, v_{\ominus} : \mathcal{V} \rightarrow \mathcal{V}$. 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

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

768 Note that the definition in the case of addition, the definition relies on that subtraction
 769 $-^{\mathbb{R}}$ in the structure of reals is the inverse of addition $+^{\mathbb{R}}$. The expressions that are to be
 770 subtracted on one side of the equation are added to the other side instead. This is also
 771 used in the case of multiplication, in combination with the distributivity law for addition
 772 $+^{\mathbb{R}}$ 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
 773 can be computed in linear time from ϕ .

Proposition 55 Positivity. For any flat formula $\phi \in \mathcal{F}_{\Sigma_{\text{bool}}}(V)$:

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

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

776 **Lemma 56.** For any flat linear equation system ϕ the formula $\widetilde{\text{dec}}_v(\phi)$ is a $h_{\mathbb{B}}$ -mixed system.

777 **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
778 $h_{\mathbb{B}}$ -mixed system.

779 9.3. Computing Sign Abstractions

780 We now have developed all the prerequisite for computing the sign abstraction of
781 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 56 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:

$$\begin{aligned} h_{\mathbb{S}} \circ \text{sol}_V^{\mathbb{R}}(\phi) &= \text{dec}^{-1} \circ h_{\mathbb{B}}^2 \circ \text{dec} \circ \text{sol}_V^{\mathbb{R}}(\phi) && \text{Decomposition Lemma 54} \\ &= \text{dec}^{-1} \circ h_{\mathbb{B}}^2 \circ \{\sigma^2 \circ \nu_{|V} \mid \sigma \in \text{sol}^{\mathbb{R}_+}(\tilde{\phi})\} && \text{Positivity Proposition 55} \\ &= \text{dec}^{-1} \circ h_{\mathbb{B}}^2 \circ \{\sigma^2 \circ \nu_{|V} \mid \sigma \in \text{sol}^{\mathbb{R}_+}(\text{br}(\tilde{\phi}))\} && \mathbb{R}_+\text{-equivalence of } \tilde{\phi} \text{ and } \text{br}(\tilde{\phi}) \\ &= \{\text{dec}^{-1} \circ \tau^2 \circ \nu_{|V} \mid \tau \in \text{sol}^{\mathbb{B}}(\text{br}(\tilde{\phi}))\} && h_{\mathbb{B}}\text{-exactness of } \text{br}(\tilde{\phi}) \\ &= \{[y/\tau(v_{\oplus}(y)) -^{\mathbb{R}} \tau(v_{\ominus}(y)) \mid y \in V] && \text{definition of } \text{dec}^{-1} \\ &\quad \mid \tau \in \text{sol}^{\mathbb{B}}(\text{br}(\tilde{\phi}))\} && \square \end{aligned}$$

782

783 The sign abstraction of a system ϕ of Σ_{bool} -equations with free variables in $V = \text{fv}(\phi)$
784 can thus be computed by first computing the $h_{\mathbb{B}}$ -exact formula $\text{br}(\tilde{\phi}) \in \mathcal{F}_{\Sigma}(v_{\oplus}(V) \cup$
785 $v_{\ominus}(V))$ from Theorem 6 by applying the Positivity Proposition 55 and the Main Theorem
786 5, then computing $\text{sol}^{\mathbb{B}}(\text{br}(\tilde{\phi}))$ by finite domain constraint programming, and finally
787 inferring $h_{\mathbb{S}} \circ \text{sol}_V^{\mathbb{R}}(\phi)$ thereof based on the equation of Theorem 6.

788 **Corollary 57.** The sign abstraction $h_{\mathbb{S}} \circ \text{sol}_V^{\mathbb{R}}(\phi)$ can be computed in at most single exponential
789 time in the size of ϕ .

790 **Proof** The formula $\text{br}(\tilde{\phi})$ is of exponential size but contains only twice as many variables
791 than ϕ . Let $n = |\text{fv}(\phi)|$. We can compute $h_{\mathbb{S}} \circ \text{sol}_V^{\mathbb{R}}(\phi)$ by testing 6^{2n} variable
792 assignments for membership to $\text{sol}^{\mathbb{B}}(\text{br}(\tilde{\phi}))$. Each such test is linear in the size of $\text{br}(\tilde{\phi})$
793 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
794 $n \leq m$ in $O(6^{3m})$.

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

```

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 $\int_0^a \mathbf{f}(x)dx$ for a given function $\mathbf{f} : \mathbb{R} \rightarrow \mathbb{R}$.

Proposition 58. Let $\psi = \exists z. \phi \wedge \phi'$ in $\mathcal{F}_{\Sigma_{\text{bool}}}(V)$ for some linear equation system ϕ and formula $\phi' \in \mathcal{F}_{\Sigma_{\text{bool}}}$. The formula $\text{br}(\widetilde{\text{dec}}_V(\psi))$ then yields an overapproximation of the sign abstraction of ϕ :

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

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

801 10. Application to Program Analysis

802 We illustrate our results by applying the sign abstraction for program analysis
803 based on abstract interpretation. We consider the Python implementation in Fig. 6 of
804 the function $I : \mathbb{R}^2 \rightarrow \mathbb{R}$. A call $I(a, s)$ supposedly computes the approximation of
805 the integral $\int_0^a \mathbf{f}(x)dx$ with step width s for some total function $\mathbf{f} : \mathbb{R} \rightarrow \mathbb{R}$. Abstract
806 interpretation allows us to find out the conditions that must hold on the input parameters
807 for $I((a : \text{float}, s : \text{float}))$ to work properly, and in particular to avoid exception throwing.
808

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 :

$$\begin{aligned}
& \exists \text{ret}_{\mathbf{f}} \exists \text{ret}_{\mathbf{I}} \exists \text{result}. \\
& (a < 0 \iff \text{raise_exception} \stackrel{\circ}{=} 1) \wedge \\
\phi_{\mathbf{I}} =_{\text{def}} & ((s > a \wedge \text{do_recursion} \stackrel{\circ}{=} 0 \wedge \text{result} \stackrel{\circ}{=} 0) \vee \\
& (\neg(s > a) \wedge \text{do_recursion} \stackrel{\circ}{=} 1 \wedge a_{\text{rec}} \stackrel{\circ}{=} a - s \wedge s_{\text{rec}} \stackrel{\circ}{=} s \wedge \\
& \text{result} \stackrel{\circ}{=} s \cdot \text{ret}_{\mathbf{f}} + \text{ret}_{\mathbf{I}})
\end{aligned}$$

809 The variables a and s are the formal parameters in the definition of $I(a : \text{float}, s :$
810 $\text{float})$. The others are fresh variables introduced to handle exceptions or function
811 calls: the boolean flag raise_exception represents exception throwing, the boolean flag
812 do_recursion has a true value only when a recursive call is made to I with actual param-
813 eters represented by the variables $a_{\text{rec}}, s_{\text{rec}}$ and return value represented by $\text{ret}_{\mathbf{I}}$, while $\text{ret}_{\mathbf{f}}$
814 is the variable for the return value of the call to the function \mathbf{f} . The final return value of
815 I is represented by the variable result . In what follows, we are not interested in the signs
816 of the last three variables, so we quantify them existentially.

817 The sign behaviour of function I is given by the formula's sign abstraction $h_{\mathbb{S}} \circ$
818 $\text{sol}^{\mathbb{R}}(\phi_{\mathbf{I}})$. Given that $\phi_{\mathbf{I}}$ is not $h_{\mathbb{B}}$ -mixed system, we cannot apply the algorithm from
819 Theorem 6 directly to compute this sign abstraction. Nevertheless, it will be beneficial as
820 we will illustrate below.

821 By John's theorem, the sign abstraction $h_{\mathbb{S}} \circ \text{sol}^{\mathbb{R}}(\phi_{\mathbf{I}})$ can be overapproximated
822 by the abstract interpretation $\text{sol}^{\mathbb{S}}(\phi_{\mathbf{I}})$. Since \mathbb{S} is a finite structure, this abstract inter-
823 pretation can be computed by finite domain constraint programming. For this, we
824 implemented a solver for first-order formulas over the structure \mathbb{S} with Minizinc [16].
825 When applied to $\phi_{\mathbf{I}}$ it returns the set of abstract solutions $\text{sol}^{\mathbb{S}}(\phi_{\mathbf{I}})$ given in Table 1. This
826 set contains the 6 unjustified abstract solutions 2, 4, 10, 13, 15, 18 outside $h_{\mathbb{S}} \circ \text{sol}^{\mathbb{R}}(\phi_{\mathbf{I}})$.

#	raise_exception	do_recursion	a	s	a _{rec}	s _{rec}
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

#	raise_exception	do_recursion	a	s	a _{rec}	s _{rec}
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 $h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\phi_I)$.

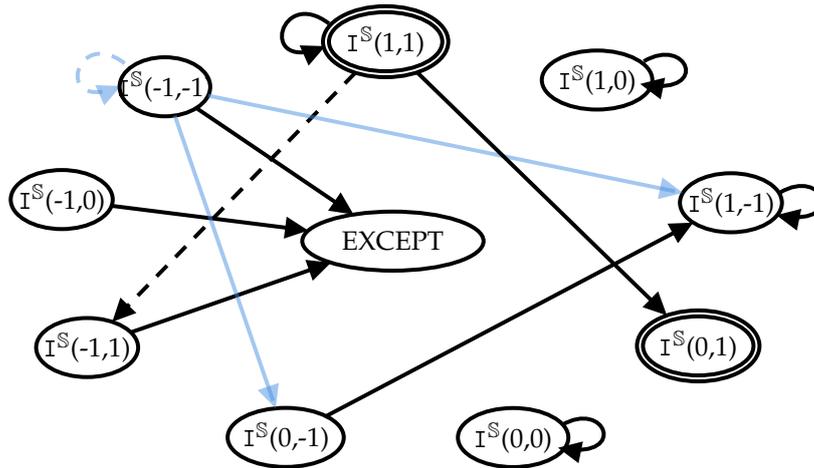


Figure 7. Sign call graph of the function I in Fig. 6 created from the sets of abstract solutions in Table 1. The solid lines correspond to abstract solutions in $h_{\mathbb{S}} \circ sol^{\mathbb{R}}(\phi_I)$ while dashed lines correspond to unjustified abstract solutions in $sol^{\mathbb{S}}(\phi_I)$. For example, $I^{\mathbb{S}}(1, -1)$ represents the assignment $[a/1, s/-1]$, that is the signs of a and s in calls $I(a : float, s : float)$ 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.

827 In the table they are distinguished by gray background color. We also note that the last
828 three solutions 17, 18, 19 could be ruled out when using a more precise abstract program
829 interpretation, taking into account that no recursive call is possible when an exception is
830 thrown.

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

836 Any set of abstract solutions defines an abstract call graph. The abstract call graphs
837 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
838 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
839 solution 1 is justified. Edges induced by unjustified solutions are dashed. The unjustified
840 solution 10 for instance induces the dashed edge from $I^{\mathbb{S}}(1, 1)$ to $I^{\mathbb{S}}(1, -1)$. It should be
841 noticed that solutions with $do_recursion = 0$ and $raise_exception = 0$ do not induce any
842 edge. Instead, they show that the computation may stop, producing final nodes that
843 are surrounded by a double circle. The final nodes are $I^{\mathbb{S}}(1, 1)$ and $I^{\mathbb{S}}(0, 1)$. Note that
844 for all non-final nodes, either an exception is raised or the computation loops endlessly.
845 Solutions with $raise_exception = 1$ induce an edge to the EXCEPT node.

846 Given that only 2 unjustified solutions with $do_recursion = 0$ and $raise_exception = 0$
 847 (10 and 18), there are only 2 dashed edges in the graph. Furthermore, the edges induced
 848 by the last three solutions 17, 18, 19 are drawn in blue, since these could be removed
 849 with a more precise abstract program interpretation than ϕ_I .

850 The sign analysis without the unjustified dashed edges yields the following result:
 851 the program in state $I^S(1, 1)$, where $a > 0$ and $s > 0$ may either terminate, loop
 852 indefinitely, or go to state $I^S(0, 1)$ and terminate there immediately. With the unjustified
 853 dashed edges, however, it wrongly seems possible that the program may also raise an
 854 exception by passing through $I^S(-1, 1)$. This overapproximation would be particularly
 855 unfortunate since state $I^S(1, 1)$ 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 58, 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 $s - a \stackrel{\circ}{=} signvar$, and rewrite $s > a$ to $signvar > 0$. The linear part of ϕ_I then becomes:

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

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:

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

The remainder of ϕ_I can be rewritten as follows:

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

856 It is not clear whether the conjunction of both parts is a $h_{\mathbb{B}}$ -mixed system, since it is not
 857 clear how to show the $h_{\mathbb{B}}$ -invariance of the equation $result \stackrel{\circ}{=} s * ret_f + ret_I$. Still we can
 858 apply the overapproximation algorithm of the sign abstraction from Proposition 58. It
 859 indeed improves on John's approximation, ruling out both unjustified solutions. The
 860 details are worked out in Appendix 11.

861 In the general case, linear equation systems are not enough, in which case our
 862 algorithm from Theorem 6 for computing sign abstractions cannot be applied. But
 863 then we can still apply the overapproximation algorithm from Proposition 58 which
 864 rewrites a linear part of the formula exactly. As illustrated by the present example, this
 865 overapproximation is often way more precise than John's.

866 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:

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

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

The decomposition of the linear subsystem $\text{dec}_v(\phi_{\mathbb{I}}^{\text{lin}})$ 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:

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

The additional constraints on the decomposition variables are:

$$\begin{aligned} & v_{\oplus}(s) * v_{\ominus}(s) \stackrel{\circ}{=} 0 \\ \wedge & v_{\oplus}(a) * v_{\ominus}(a) \stackrel{\circ}{=} 0 \\ \wedge & v_{\oplus}(\text{signvar}) * v_{\ominus}(\text{signvar}) \stackrel{\circ}{=} 0 \\ \wedge & v_{\oplus}(a_{\text{rec}}) * v_{\ominus}(a_{\text{rec}}) \stackrel{\circ}{=} 0 \\ \wedge & v_{\oplus}(s_{\text{rec}}) * v_{\ominus}(s_{\text{rec}}) \stackrel{\circ}{=} 0 \\ \wedge & v_{\oplus}(\text{result}) * v_{\ominus}(\text{result}) \stackrel{\circ}{=} 0 \\ \wedge & v_{\oplus}(\text{ret}_{\mathbb{I}}) * v_{\ominus}(\text{ret}_{\mathbb{I}}) \stackrel{\circ}{=} 0 \\ \wedge & v_{\oplus}(\text{ret}_{\mathbb{F}}) * v_{\ominus}(\text{ret}_{\mathbb{F}}) \stackrel{\circ}{=} 0 \end{aligned}$$

867 The elementary mode rewriting $\text{emr}(\text{dec}_v(\phi_{\mathbb{I}}^{\text{lin}}))$ is the following \mathbb{R}_+ -equivalent
868 $h_{\mathbb{B}}$ -exact Σ_{bool} -formula obtained via Corollary 21:

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

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

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

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 $\sigma \in h_{\mathbb{S}} \circ \text{sol}^{\mathbb{R}}(\phi_{\mathbb{I}})$ according to Theorem 6:

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

872

873 12. Conclusion and Future Work

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

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

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

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

901 Another interesting question for future work is how to compute more quantitative
902 abstractions exactly, as for instance with intervals. In this case however the structure of
903 abstract values is infinite, therefore finite domain constraint programming is no longer
904 sufficient to compute the set of abstract solutions.

905 **Appendix A**

906 *Appendix A.1*

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 :

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

907

908 **References**

- 909 1. Cousot, P.; Cousot, R. Systematic Design of Program Analysis Frameworks. Conference
910 Record of the Sixth Annual ACM Symposium on Principles of Programming Languages, San
911 Antonio, Texas, USA, January 1979; Aho, A.V.; Zilles, S.N.; Rosen, B.K., Eds. ACM Press,
912 1979, pp. 269–282. doi:10.1145/567752.567778.
- 913 2. Paulevé, L.; Sené, S. Non-Deterministic Updates of Boolean Networks. 27th IFIP WG 1.5
914 International Workshop on Cellular Automata and Discrete Complex Systems, AUTOMATA
915 2021, July 12-14, 2021, Aix-Marseille University, France; Castillo-Ramirez, A.; Guillon, P.;
916 Perrot, K., Eds. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, Vol. 90, *OASlcs*, pp.
917 10:1–10:16. doi:10.4230/OASlcs.AUTOMATA.2021.10.
- 918 3. Paulevé, L. Most Permissive Reaction Networks. Preprint available at [https://loicpauleve.
919 name/md/ak8WJ5d2TqKpmJBtP_8BaQ#](https://loicpauleve.name/md/ak8WJ5d2TqKpmJBtP_8BaQ#).
- 920 4. Cousot, P.; Halbwachs, N. Automatic Discovery of Linear Restraints Among Variables
921 of a Program. Conference Record of the Fifth Annual ACM Symposium on Principles
922 of Programming Languages, Tucson, Arizona, USA, January 1978; Aho, A.V.; Zilles, S.N.;
923 Szymanski, T.G., Eds. ACM Press, 1978, pp. 84–96. doi:10.1145/512760.512770.
- 924 5. Granger, P. Static Analysis of Linear Congruence Equalities among Variables of a Program.
925 TAPSOFT'91: Proceedings of the International Joint Conference on Theory and Practice of
926 Software Development, Brighton, UK, April 8-12, 1991, Volume 1: Colloquium on Trees in
927 Algebra and Programming (CAAP'91); Abramsky, S.; Maibaum, T.S.E., Eds. Springer, 1991,
928 Vol. 493, *Lecture Notes in Computer Science*, pp. 169–192. doi:10.1007/3-540-53982-4_10.
- 929 6. Allart, E.; Niehren, J.; Versari, C. Computing Difference Abstractions of Linear
930 Equation Systems. *Theoretical Computer Science* 2021. Journal extension of [26], doi:
931 <https://doi.org/10.1016/j.tcs.2021.06.030>.

- 932 7. Niehren, J.; Versari, C.; John, M.; Coutte, F.; Jacques, P. Predicting changes of reaction
933 networks with partial kinetic information. *Biosyst.* **2016**, *149*, 113–124. doi:
934 10.1016/j.biosystems.2016.09.003.
- 935 8. John, M.; Nebut, M.; Niehren, J. Knockout Prediction for Reaction Networks with Partial
936 Kinetic Information. Verification, Model Checking, and Abstract Interpretation, 14th Interna-
937 tional Conference, VMCAI 2013, Rome, Italy, January 20–22, 2013. Proceedings; Giacobazzi,
938 R.; Berdine, J.; Mastroeni, I., Eds. Springer, 2013, Vol. 7737, *Lecture Notes in Computer Science*,
939 pp. 355–374. doi:10.1007/978-3-642-35873-9_22.
- 940 9. Giacobazzi, R.; Ranzato, F.; Scozzari, F. Making abstract interpretations complete. *J. ACM*
941 **2000**, *47*, 361–416. doi:10.1145/333979.333989.
- 942 10. Nethercote, N.; Stuckey, P.J.; Becket, R.; Brand, S.; Duck, G.J.; Tack, G. MiniZinc: Towards a
943 Standard CP Modelling Language. Principles and Practice of Constraint Programming - CP
944 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23–27, 2007,
945 Proceedings; Bessiere, C., Ed. Springer, 2007, Vol. 4741, *Lecture Notes in Computer Science*, pp.
946 529–543. doi:10.1007/978-3-540-74970-7_38.
- 947 11. Motzkin, T.; Raiffa, H.; Thompson, G.; Thrall, R. The double description method. Contribu-
948 tions to theory of games; Kuhn, H.; A.W.Tucker., Eds. Princeton University Press, 1953,
949 Vol. 2.
- 950 12. Fukuda, K.; Prodon, A. Double Description Method Revisited. Combinatorics and Computer
951 Science, 8th Franco-Japanese and 4th Franco-Chinese Conference, Brest, France, July 3–5,
952 1995, Selected Papers; Deza, M.; Euler, R.; Manoussakis, Y., Eds. Springer, 1995, Vol. 1120,
953 *Lecture Notes in Computer Science*, pp. 91–111. doi:10.1007/3-540-61576-8_77.
- 954 13. Gagneur, J.; Klamt, S. Computation of elementary modes: a unifying framework and the
955 new binary approach. *BMC Bioinform.* **2004**, *5*, 175. doi:10.1186/1471-2105-5-175.
- 956 14. Zanghellini, D.; Ruckerbauer, D.E.; Hanscho, M.; Jungreuthmayer, C. Elementary flux
957 modes in a nutshell: Properties, calculation and applications. *Biotechnology Journal* **2013**,
958 *8*(9), 1009–1016. doi:10.1002/biot.201200269.
- 959 15. Bagnara, R.; Hill, P.M.; Zaffanella, E. The Parma Polyhedra Library: Toward a complete set
960 of numerical abstractions for the analysis and verification of hardware and software systems.
961 *Sci. Comput. Program.* **2008**, *72*, 3–21. doi:10.1016/j.scico.2007.08.001.
- 962 16. Rendl, A.; Guns, T.; Stuckey, P.J.; Tack, G. MiniSearch: A Solver-Independent Meta-Search
963 Language for MiniZinc. Principles and Practice of Constraint Programming - 21st Inter-
964 national Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings;
965 Pesant, G., Ed. Springer, 2015, Vol. 9255, *Lecture Notes in Computer Science*, pp. 376–392. doi:
966 10.1007/978-3-319-23219-5_27.
- 967 17. Allart, E.; Niehren, J.; Versari, C. Reaction Networks to Boolean Networks. Preprint available
968 at <https://hal.archives-ouvertes.fr/hal-02279942>.
- 969 18. Dines, L.L. On Positive Solutions of a System of Linear Equations. *Annals of Mathematics*
970 **1926**, *28*, 386–392.
- 971 19. Miné, A. A Few Graph-Based Relational Numerical Abstract Domains. Static Analysis, 9th
972 International Symposium, SAS 2002, Madrid, Spain, September 17–20, 2002, Proceedings;
973 Hermenegildo, M.V.; Puebla, G., Eds. Springer, 2002, Vol. 2477, *Lecture Notes in Computer*
974 *Science*, pp. 117–132. doi:10.1007/3-540-45789-5_11.
- 975 20. Cousot, P.; Cousot, R. Static determination of dynamic properties of programs. Proceedings
976 of the Second International Symposium on Programming. Dunod, Paris, France, 1976, pp.
977 106–130.
- 978 21. Granger, P. Static analysis of arithmetical congruences. *International Journal of Com-*
979 *puter Mathematics* **1989**, *30*, 165–190, [<https://doi.org/10.1080/00207168908803778>]. doi:
980 10.1080/00207168908803778.
- 981 22. Karr, M. Affine relationships among variables of a program. *Acta Informatica* **1976**, *6*, 133–151.
- 982 23. Klamt, S.; Stelling, J.; Ginkel, M.; Gilles, E.D. FluxAnalyzer: exploring structure,
983 pathways, and flux distributions in metabolic networks on interactive flux maps.
984 *Bioinformatics* **2003**, *19*, 261–269, [[https://academic.oup.com/bioinformatics/article-](https://academic.oup.com/bioinformatics/article-pdf/19/2/261/1059937/190261.pdf)
985 [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.
- 986 24. Avis, D.; Jordan, C. mplrs: A scalable parallel vertex/facet enumeration code. *Math. Program.*
987 *Comput.* **2018**, *10*, 267–302. doi:10.1007/s12532-017-0129-y.
- 988 25. Fukuda, K. cddlib – An efficient implementation of the Double Description Method, 2018.
989 Available at <https://github.com/cddlib/cddlib>.

-
- 990 26. Allart, E.; Versari, C.; Niehren, J. Computing Difference Abstractions of Metabolic Networks
991 Under Kinetic Constraints. CMSB 2019 - 17th International Conference on Computational
992 Methods in Systems Biology; Luca Bortolussi and Guido Sanguinetti, Springer: Trieste, Italy,
993 2019; Vol. 11773, *Lecture Notes in Computer Science*, pp. 266–285. doi:10.1007/978-3-030-31304-
994 3_14.

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