Non-Numerical Weakly Relational Domains

Helmut Seidl · **Julian Erhard** · **Sarah Tilscher** · **Michael Schwarz**

Abstract The weakly relational domain of *Octagons* offers a decent compromise between precision and efficiency for numerical properties. Here, we are concerned with the construction of non-numerical relational domains. We provide a general construction of weakly relational domains, which we exemplify with an extension of constant propagation by disjunctions. Since for the resulting domain of 2-disjunctive formulas, satisfiability is NP-complete, we provide a general construction for a further, more abstract weakly relational domain where the abstract operations of restriction and least upper bound can be efficiently implemented.

In the second step, we consider a relational domain that tracks conjunctions of inequalities between variables, and between variables and constants for arbitrary partial orders of values. Examples are sub(multi)sets, as well as prefix, substring or scattered substring orderings on strings. When the partial order is a lattice, we provide precise polynomial algorithms for satisfiability, restriction, and the best abstraction of disjunction. Complementary to the constructions for lattices, we find that, in general, satisfiability of conjunctions is NP-complete. We therefore again provide polynomial abstract versions of restriction, conjunction, and join. By using our generic constructions, these domains are extended to weakly relational domains that additionally track *disjunctions*.

For all our domains, we indicate how abstract transformers for assignments and guards can be constructed.

Helmut Seidl · Julian Erhard · Sarah Tilscher · Michael Schwarz
Technische Universität München, Garching, Germany
E-mail: {helmut.seidl, julian.erhard, sarah.tilscher, m.schwarz}@tum.de

Keywords weakly relational domains, 2-decomposable relational domains, 2-disjunctive constants, directed domains

1 Introduction

Relational analyses have been observed to be indispensable for verifying intricate program properties. In particular, this is the case when for the purpose of verification, *ghost* variables have been introduced which must be related to program variables. Termination may be verified by introducing a *ghost* loop counter, which can be proven bounded by a relational domain relating it to the actual bounded iteration variable [2]. The validity of string operations on null-terminated strings as employed, e.g., in the programming language C, may be verified by introducing *ghost* variables for the length of a buffer as well as for tracking the position of the null byte in the buffer [12]. It also has been observed that *monolithic* relational domains such as the polyhedra abstract domain [10] scale badly to larger programs. Therefore, *weakly relational* domains have been proposed which can only express simple relational properties, but have the potential to scale better [16]. Examples of weakly relational *numerical* properties are the *Two Variables Per Inequality* domain [23], or domains given by a finite set of *linear templates* [20]. The most prominent example of a template numerical domain is the *Octagon* domain [15, 17] which allows tracking upper and lower bounds not only of program variables but also of sums and differences of *two* program variables. One such octagon abstract relation could, e.g., be given by the conjunction

$$(-x \leqslant -5) \wedge (x \leqslant 10) \wedge (x + y \leqslant 0) \wedge (x - z \leqslant 1)$$

Octagons thus can be considered as a mild extension of the non-relational domain of *Intervals* for program variables, and a variety of efficient algorithms have been provided [4, 5, 7, 22]. Here, we are concerned with constructing *non-numerical* abstract domains.

For that, we provide a general technique to construct from *every* relational domain a weakly relational domain. As one instance of the general construction, we consider 2-disjunctive constants as mentioned in [21]. This weakly relational domain allows, e.g., to relate the names of functions with function pointers as in the formula

$$x = \texttt{"foo"} \wedge y = \&\texttt{foo} \;\vee\; x = \texttt{"bar"} \wedge y = \&\texttt{bar}$$

Since satisfiability of formulas from that domain turns out to be NP-complete, we provide a further mild abstraction, again for arbitrary relational domains, to provide us with a weakly relational domain where all required operations become tractable.

Another family of relational non-numerical domains has been introduced by Arceri et al. [3]. Based on a partial order of values, conjunctions of ordering constraints $x \sqsubseteq y$ for program variables x, y are considered. They observe that analyses of prefixes or the substring relation could be helpful for programs in programming languages supporting high-level operations on strings. Here, we study this kind of directed domains in greater detail. For conjunctions of inequalities over some partial order P, we extend the constraints from Arceri et al. [3] by allowing for variables both lower and upper bounds from P. For arbitrary partial orders, though, we find that then satisfiability is NP-complete. Partial orders p that are lattices form a notable exception. An instance of this are subsets of some universe or multisets. For lattices, we show that satisfiability is decidable in polynomial time. Moreover, we provide polynomial constructions both for restriction as well as the optimal join operation. Turning to general partial orders of values, we thus cannot hope for polynomial algorithms. Therefore, we provide a meaningful abstraction so that both abstract restriction as well as join is again polynomial. This family of relational domains is already weakly relational. Still, our generic constructions can be applied to obtain more expressive weakly relational domains that additionally support disjunctions at a limited amount of extra costs.

The paper is organized as follows: Section 2 provides background definitions on relational domains. It formally introduces our notion of weakly relational domains and provides a general construction of weakly relational domains. Section 3 is dedicated to disjunctive constants. When applying the generic construction from the last section to this relational domain, the weakly relational domain of 2-disjunctive constants is obtained. Here, we prove that satisfiability for these formulas still is NP-complete. Therefore, a generic abstraction technique is presented so that, when applied to disjunctive constants, normalization, projection, as well as least upper bounds all turn out to be polynomial time.

Finally, abstract transformers for assignments as well as guards are derived. Section 4 then introduces *directed domains* which do not track equalities but inequalities over a partial order of values. While the first subsection provides polynomial constructions for the case that the partial order for values is a lattice, the second subsection is concerned with arbitrary partial orders as value domain. Since satisfiability, in general, turns out to be NP-complete, again a polynomial abstraction is provided. In a further subsection, we indicate how the generic constructions from the last sections provide us with weakly relational domains that additionally support disjunctions of inequalities. We exemplify the resulting domains with conjunctions and disjunctions of inequalities over the integers. In the final subsection, dedicated abstract transformers are constructed for assignments, while the last subsection discusses the treatment of guards. Section 5 summarizes the contributions and sketches further directions of research.

2 Weakly Relational Domains

Let us recall basic definitions for relational domains. We mostly follow the notation used in previous work [21], where the notion of 2-decomposability has been introduced. Let $\mathcal{X}$ be some finite set of variables. A *relational domain* $\mathcal{R}$ maintains relations between variables in $\mathcal{X}$. We require that a relational domain is a bounded lattice, i.e., has a partial order $\sqsubseteq$, a least element $\bot$, a greatest element $\top$, as well as binary operators for the greatest lower bound (meet) $\sqcap$ and the least upper bound (join) $\sqcup$. We do not demand relational domains to be *complete* lattices, i.e., to provide for *every* subset of elements a least upper bound: the polyhedral domain, e.g., is not complete [10]. However, we demand that a relational domain supports the following monotonic operations:

$$\begin{aligned} [\![x := e]\!]^\sharp &: \mathcal{R} \to \mathcal{R} \text{ (assignment of } e \text{ to } x\text{)} \\ \cdot|_Y &: \mathcal{R} \to \mathcal{R} \text{ (restriction to } Y \subseteq \mathcal{X}\text{)} \\ [\![?c]\!]^\sharp &: \mathcal{R} \to \mathcal{R} \text{ (guard for condition } c\text{)} \end{aligned}$$

where e and c are from some expression and condition language, respectively.

The abstract transformers for basic actions of programs are given by these functions. Restricting a relation r to a subset Y of variables amounts to *forgetting* all information about variables in $\mathcal{X}\backslash Y$. Thus, we require that

$$\begin{array}{ll} r|_{\mathcal{X}} &= r \\ r|_{\varnothing} &= \begin{cases} \bot \text{ if } r = \bot \\ \top \text{ otherwise} \end{cases} \\ r|_{Y_1} &\sqsupseteq r|_{Y_2} \quad \text{when } Y_1 \subseteq Y_2 \\ (r|_{Y_1})|_{Y_2} &= r|_{Y_1 \cap Y_2} \end{array} \tag{1}$$

A restriction $\cdot|_Y$ to some set Y therefore is an *idempotent* operation. We remark that from these axioms it follows that $\bot|_Y = \bot$ and $\top|_Y = \top$ for any $Y \subseteq \mathcal{X}$. Given that there is some relation $r_c \in \mathcal{R}$ describing all states satisfying the condition c, the transformation for the guard $?c$ can be described by

$$[\![?c]\!]^\sharp r = r \sqcap r_c \tag{2}$$

– at least, if there is a concretization function γ such that

$$\gamma\,(r_1 \sqcap r_2) = \gamma\, r_1 \cap \gamma\, r_2 \tag{3}$$

i.e., the binary meet operation is *precise.*

Example 1 For numerical variables, a variety of such relational domains have been proposed, e.g., (conjunctions of) *affine equalities* [14, 18, 19] or *affine inequalities* [10]. For affine equalities or inequalities, restriction to a subset of Y of variables corresponds to the geometric projection onto the subspace defined by Y, combined with arbitrary values for variables $z \notin Y$. □

One way to tackle the high cost of relational domains is to track the relationships not between all variables, but only between *subclusters* of variables. We call such domains *Weakly Relational Domains.*

For a subset $Y \subseteq \mathcal{X}$, let $\mathcal{R}^Y = \{r|_Y \mid r \in \mathcal{R}\}$ be the set of all abstract values from $\mathcal{R}$ that contains only information on those variables in Y. For any collection $\mathcal{S} \subseteq 2^{\mathcal{X}}$ of *clusters* of variables, a relation $r \in \mathcal{R}$ can be *approximated* by a meet of relations from $\mathcal{R}^Y, Y \in \mathcal{S}$ since for every $r \in \mathcal{R}$,

$$r \sqsubseteq \bigsqcap_{Y \in \mathcal{S}} r|_Y \tag{4}$$

holds, as $r \sqsubseteq r|_Y$ holds for each $Y \in S$. In fact, the right-hand side of (4) is the *best* approximation of r by some meet over abstract relations $s_Y, Y \in \mathcal{S}$, with $s_Y \in \mathcal{R}^Y$, i.e., with $s_Y|_Y = s_Y$, since

$$\begin{array}{ll} r|_Y &\sqsubseteq (\bigsqcap Y' \in \mathcal{S} s_{Y'})|_Y \\ &\sqsubseteq s_Y|_Y \qquad \text{(by monotonicity of restriction)} \\ &= s_Y \end{array}$$

holds for all $Y \in \mathcal{S}$.

Schwarz et al. [21] have introduced 2-*decomposable* relational domains. These are domains where the full value r can be recovered from the restrictions of r to all clusters p from the set $\mathcal{S} = [\mathcal{X}]_2$ of non-empty clusters of variables of size at most 2. Furthermore, Schwarz et al. [21] ask for binary least upper bounds to be determined by computing within these clusters only. More precisely, this amounts to requiring the following two properties

$$r = \bigsqcap\nolimits_{p \in [\mathcal{X}]_2} r|_p \tag{5}$$

$$(r_1 \sqcup r_2)\,|_p = r_1|_p \sqcup r_2|_p \qquad (p \in [\mathcal{X}]_2) \tag{6}$$

to hold for all abstract relations $r, r_1, r_2 \in \mathcal{R}$. The most prominent example of a 2-decomposable domain is the *octagon* domain [15] – either over rationals or integers, while *affine equalities* or *affine inequalities* are examples of domains that are not 2-decomposable.

Any relational domain $\mathcal{R}$, however, which satisfies (6) gives rise to a 2-decomposable domain $\mathcal{R}_2$ of its 2-cluster approximations.

For $r \in \mathcal{R}$, let $\overline{r} = \bigsqcap_{p \in [\mathcal{X}]_2} r|_p$ denote the approximation of r by the meet of its restrictions to clusters $p \in [\mathcal{X}]_2$. Let $\mathcal{R}_2$ denote the subset of $\mathcal{R}$ of all abstract relations of the form $\overline{r}, r \in \mathcal{R}$, where the ordering is inherited from $\mathcal{R}$. In particular, $\bot$ as well as $\top$ from $\mathcal{R}$ are also in $\mathcal{R}_2$.

Theorem 1 *Assume that $\mathcal{R}$ is an abstract relational domain which satisfies* (6)*. Then the following holds:*

1. *$r = \overline{r}$ for all conjunctions $r = \bigsqcap_{p \in [\mathcal{X}]_2} s_p$ with $s_p \in \mathcal{R}^p, p \in [\mathcal{X}]_2$, i.e., all such conjunctions are contained in $\mathcal{R}_2$.*
2. *For $r_1, r_2 \in \mathcal{R}_2$, the abstract relation $r_1 \sqcap r_2$, as provided by $\mathcal{R}$, is in $\mathcal{R}_2$.*
3. *The binary least upper bound operation in $\mathcal{R}_2$ exists and is given by*
$$r_1 \sqcup_2 r_2 = \bigsqcap_{p \in [\mathcal{X}]_2} (r_1|_p \sqcup r_2|_p)$$
4. *For $\mathcal{R}_2$, the best approximation $r|_{Y,2}$ to the restriction $r|_Y$ of $r \in \mathcal{R}_2$ onto some subset $Y \subseteq \mathcal{X}$ of variables is given by*
$$r|_{Y,2} = \bigsqcap_{p \in [\mathcal{X}]_2} r|_{p \cap Y}$$
5. *the partial order $\mathcal{R}_2$ with the given binary greatest lower and least upper bounds is a 2-decomposable relational domain.*

Proof For a proof of statement (1), we first observe that for each $p \in [\mathcal{X}]_2$,

$$r|_p = \left(\bigsqcap_{p \in [\mathcal{X}]_2} s_p \right) \Bigg|_p \sqsubseteq s_p|_p = s_p$$

by monotonicity and idempotence of restriction. Thus,

$$r \sqsubseteq \overline{r} = \bigsqcap_{p\in[\mathcal{X}]_2} r|_p \sqsubseteq \bigsqcap_{p\in[\mathcal{X}]_2} s_p = r$$

where the first inequality follows from Eq. (4). Thus, statement (1) follows.

For a proof of statement (2), consider elements $r, s \in \mathcal{R}_2$. Then

$$r \sqcap s = \bigsqcap_{p\in[\mathcal{X}]_2} r|_p \sqcap \bigsqcap_{p\in[\mathcal{X}]_2} s|_p = \bigsqcap_{p\in[\mathcal{X}]_2} (r|_p \sqcap s|_p)$$

Now, we claim that for every $p \in [\mathcal{X}]_2$,

$$r|_p \sqcap s|_p = (r|_p \sqcap s|_p)\Big|_p$$

To prove the claim, we argue that

$$\begin{aligned} r|_p \sqcap s|_p &\sqsubseteq (r|_p \sqcap s|_p)\Big|_p && \text{(by monotonicity)}\\ &\sqsubseteq (r|_p)\Big|_p \sqcap (s|_p)\Big|_p && \text{(by monotonicity)}\\ &= r|_p \sqcap s|_p && \text{(by idempotence)} \end{aligned}$$

and the claim follows. So far, we have proven that

$$r \sqcap s = \bigsqcap_{p\in[\mathcal{X}]_2} t_p$$

for some $t_p \in \mathcal{R}^p$, $p \in [\mathcal{X}]_2$. Then, statement (2) follows from statement (1).

For a proof of statement (3), we note that any upper bound of r_1, r_2 in $\mathcal{R}_2$ is also an upper bound of $r_1 \sqcup r_2$ in $\mathcal{R}$. Therefore, the *least* upper bound od r_1, r_2 in $\mathcal{R}_2$ is given by $\overline{r_1 \sqcup r_2}$. We calculate:

$$\begin{aligned} \overline{r_1 \sqcup r_2} &= \textstyle\bigsqcap_{p\in[\mathcal{X}]_2} (r_1 \sqcup r_2)|_p && \text{(by definition)}\\ &= \textstyle\bigsqcap_{p\in[\mathcal{X}]_2} (r_1|_p \sqcup r_2|_p) && \text{(by (6))} \end{aligned}$$

and statement (3) follows.

The best approximation of $r|_Y$ in $\mathcal{R}_2$ is given by $\overline{r|_Y}$. Thus, we have

$$r|_{Y,2} = \bigsqcap_{p\in[\mathcal{X}]_2} (r|_Y)|_p = \bigsqcap_{p\in[\mathcal{X}]_2} r|_{Y\cap p} = \bigsqcap_{p\in[\mathcal{X}]_2} (r|_p)\Big|_Y$$

i.e., it can be determined by applying the restriction onto variables from Y for each cluster $p \in [\mathcal{X}]_2$ separately. This implies statement (4).

Statement (5) is an immediate consequence of statements (3) and (4). □

The polyhedral domain, e.g., satisfies (6). Applied to the polyhedral relational domain, the construction from Theorem 1 results in the domain of affine inequalities with at most two variables per inequality [23].

According to Theorem 1, every value r from the 2-decomposable relational domain $\mathcal{R}_2$ can be represented as the meet of its restrictions to 2-clusters, i.e., by the collection $\langle r|_p\rangle_{p\in[\mathcal{X}]_2}$. We call this representation *normal*, and an algorithm that computes it *normalization*. Consider now an *arbitrary* collection $\langle s_p\rangle_{p\in[\mathcal{X}]_2}$ with $s_p \in \mathcal{R}^p$ with $r = \bigsqcap_{p\in[\mathcal{X}]_2} s_p$. Then $r|_p \sqsubseteq s_p$ always holds, while equality need not hold. In the *Octagon* domain over the rationals or the integers, the normal representation of an octagon value corresponds to its *closure* as introduced in previous work [4, 15]. While for rational Octagons, closure in cubic time was already proposed by Miné [15], it is much more recent that a corresponding algorithm was provided for the case when constraints are interpreted over integers [4, 5].

Subsequently, we introduce non-numerical weakly relational domains and provide polynomial algorithms for these.

3 Disjunctive Constants

Constant propagation relies on a domain that maintains conjunctions of atomic propositions $x = a$ where x is a program variable and a is from a finite set U of possible values. In the following, we consider a (mild) generalization of this domain where also *disjunctions* of at most two atomic propositions are allowed.

Assume we are given a finite set U representing possible values for variables from $\mathcal{X}$. We consider propositions of the form $(x \in A)$ for $A \subseteq U$ which correspond to the disjunction of atomic propositions $x = a, a \in A$. Thus, the proposition $x \in A$ for some $A \subseteq U$ can be understood as an atomic proposition of a *multi-valued* propositional logic where A serves as the set of logical values of the propositional variable x [6]. Every monotonic Boolean combination Ψ of propositions $x \in A$ with $x \in \mathcal{X}, A \subseteq U$, represents a function $[\![\Psi]\!] : (\mathcal{X} \to U) \to \mathcal{B}$ defined by

$$\begin{aligned} [\![x \in A]\!]\,\sigma &= (\sigma\, x) \in A\\ [\![\Psi_1 \vee \Psi_2]\!]\,\sigma &= [\![\Psi_1]\!]\,\sigma \vee [\![\Psi_2]\!]\,\sigma\\ [\![\Psi_1 \wedge \Psi_2]\!]\,\sigma &= [\![\Psi_1]\!]\,\sigma \wedge [\![\Psi_2]\!]\,\sigma \end{aligned}$$

Let $\mathcal{C}[U]$ denote the complete lattice of all equivalence classes of formulas Ψ where the ordering is semantic implication. The least element in this ordering can be represented by the empty disjunction or $\bot$ (*false*), while the greatest element is equivalent to the empty conjunction or $\top$ (*true*). Each formula Ψ has an equivalent CNF as well as an equivalent DNF where each clause (conjunction) contains at most one proposition $x \in A$ for every variable x. Converting Ψ into DNF allows checking satisfiability and computing the restriction $\Psi|_Y$ onto a subset $Y \subseteq \mathcal{X}$ of variables. A formula for $\Psi|_Y$ is obtained from a DNF for Ψ where each conjunction contains at most one proposition for each variable by the following steps: First, every conjunction which contains $y \in \varnothing$ for some y is removed. From each remaining conjunction, then every proposition $y \in A$ with $y \notin Y$ is removed. It follows that $\Psi|_Y$ is distributive, i.e., commutes with binary least upper bounds.

For an arbitrary $\Psi \in \mathcal{C}[U]$, computing an equivalent DNF is an exponential time operation. The same holds if all restrictions $\Psi|_{\{x,y\}}$ are computed via this normal form. Let $\mathcal{C}_2[U]$ denote the 2-decomposable domain obtained from $\mathcal{C}[U]$ according to theorem 1. The lattice $\mathcal{C}_2[U]$ consists of all elements Ψ which can be represented as conjunctions of clauses with at most two propositions $x \in A_x$ per clause. According to theorem 1, the least upper bound operation $\sqcup_2$ for $\mathcal{C}_2[U]$ can be realized by a clusterwise disjunction. In particular, it does *not* coincide with logical disjunction – but is an over-approximation of it.

Example 2 Let $\Psi_1 \equiv (x \in \{a\})$ and $\Psi_2 \equiv (y \in \{b\} \vee z \in \{c\})$. Then both Ψ_1 and Ψ_2 are from $\mathcal{C}_2[U]$, but their disjunction is not. In fact, the least upper bound in $\mathcal{C}_2[U]$ for

$$(x \in \{a\}) \vee (y \in \{b\}) \vee (z \in \{c\})$$

is $\top$. $\square$

3.1 *Approximating 2-disjunctive Conjunctions*

Any CNF Ψ over some set Y of variables of bounded size can, in polynomial time, be transformed into a DNF Ψ'. Each DNF over two distinct variables x, y can be brought into the canonical normal form

$$\bigvee_{(a,b)\in L} (x = a) \wedge (y = b) \tag{7}$$

for some $L \subseteq U \times U$. Conjunction and disjunction of two such normal forms then correspond to intersection and union of the respective subsets of $U \times U$.

For arbitrary sets Y of variables, though, it is nontrivial even to decide whether a given conjunction is different from $\perp$.

Theorem 2 *To decide for a formula $\Psi \in \mathcal{C}_2[U]$ whether or not Ψ is satisfiable, i.e., different from $\perp$, is NP-complete.*

Proof Since a satisfying assignment for Ψ can be guessed and then checked in polynomial time, satisfiablity of Ψ is in NP. NP-hardness, on the other hand, follows by a reduction from 3-colorability of graphs [6]. We illustrate the reduction with an example.

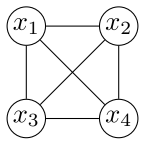
Example 3 For $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$, consider the formula Ψ

$$\bigwedge_{\{x_i,x_j\}\in E} \begin{array}{l}(x_i \in \{b,c\} \vee x_j \in \{b,c\}) \;\wedge \\ (x_i \in \{a,c\} \vee x_j \in \{a,c\}) \;\wedge \\ (x_i \in \{a,b\} \vee x_j \in \{a,b\})\end{array}$$

where E is given by

$$E = \{\{x_1,x_2\},\{x_1,x_4\},\{x_2,x_3\},\{x_3,x_4\},\{x_1,x_3\}\}$$

Then Ψ is satisfiable iff the undirected graph $(\mathcal{X}, E)$ has a 3-coloring. In the given example, the graph



cannot be colored by three colors. Therefore, Ψ is equivalent to $\perp$. $\square$

Exact normalization (as defined in Section 2) of a relation represented by some 2-CNF thus, in general, may be difficult to compute. Instead of giving dedicated further abstraction techniques, we prefer to provide for an *arbitrary* relational domain $\mathcal{R}$, a *general* construction to approximate the 2-decomposable domain $\mathcal{R}_2$ further by a 2-decomposable domain $\mathcal{R}_2^\sharp$. This construction is based on *approximate* normalization.

Assume that an element in $\mathcal{R}_2$ is given by the meet $\bigsqcap R$ where R is the collection $\langle s_p \rangle_{p\in[\mathcal{X}]_2}$ with $s_p \in \mathcal{R}^p$ $(p \in [\mathcal{X}]_2)$. According to Theorem 1, $(\bigsqcap R)|_p \sqsubseteq s_p$ for all $p \in [\mathcal{X}]_2$. As we have seen for 2-disjunctive constants, however, *exact* normalization of $\bigsqcap R$, i.e., the values $(\bigsqcap R)|_p$ may be hard to compute *precisely*. For an approximate normalization, we introduce a constraint system in unknowns $r_p, p \in [\mathcal{X}]_2$ with the constraints

$$\begin{array}{lll} r_{\{x,y\}} \sqsubseteq s_{\{x,y\}} & & (x, y \in \mathcal{X}) \\ r_{\{x,y\}} \sqsubseteq (r_{\{x,z\}} \sqcap r_{\{z,y\}})|_{\{x,y\}} & & (x, y, z \in \mathcal{X}) \end{array} \tag{8}$$

This constraint system has already been considered for the normalization of 2-projective domains [22]. As all right-hand sides are monotonic, the constraint system has a greatest solution – whenever each $\mathcal{R}^p, p \in [\mathcal{X}]_2$, is a complete lattice.

In case that there is a greatest solution $\langle r_p \rangle_{p\in[\mathcal{X}]_2}$, $(\bigsqcap R)|_p \sqsubseteq r_p$ holds for all p, since $\langle (\bigsqcap R)|_p \rangle_{p\in[\mathcal{X}]_2}$ is also a solution of the system (8). Then we call the collection $\langle r_p \rangle_{p\in[\mathcal{X}]_2}$ the *approximate* normal form of the collection R. Here, we are not only interested in the *existence* of a greatest solution of (8) but also that it can be effectively computed. For that, we consider the sets of values possibly occurring during some fixpoint iteration for a particular collection $R = \langle s_p \rangle_{p\in[\mathcal{X}]_2}$.

Let $I_{\mathcal{R}}[R]^p, p \in [\mathcal{X}]_2$, be the least collection of sets such that

– $s_p \in I_{\mathcal{R}}[R]^p$;
– If $r, r' \in \mathcal{R}_R^p$ then also $r \sqcap r' \in \mathcal{R}_R^p$;
– If $r \in I_{\mathcal{R}}[R]^{\{x,z\}}$ and $r' \in I_{\mathcal{R}}[R]^{\{z,y\}}$, then $(r \sqcap r')|_{\{x,y\}} \in I_{\mathcal{R}}[R]^{\{x,y\}}$ for all $x, y, z \in \mathcal{X}$.

The sets $I_{\mathcal{R}}[R]^p$ collect the potential iterates occurring during greatest fixpoint iteration of (8). By construction, each set $I_{\mathcal{R}}[R]^p$ has a greatest element, namely, s_p, and is closed under binary $\sqcap$. For the termination of Kleene fixpoint iteration for (8), it suffices for each set $I_{\mathcal{R}}[R]^p$ to have a *least* element – whose collection then coincides with the greatest solution of (8). This observation is summarized in the following proposition.

Proposition 1 *The following two statements are equivalent:*

1. *For each $p \in [\mathcal{X}]_2$, $I_{\mathcal{R}}[R]^p$ has a least element;*
2. *The constraint system* (8) *has a greatest solution which can be attained by Kleene fixpoint iteration.*

Proof Assume that for each $p \in [\mathcal{X}]_2$, there is a least element $d_p \in I_{\mathcal{R}}[R]^p$. We claim that $\underline{R} = \langle d_p \rangle_{p \in [\mathcal{X}]_2}$ is the greatest solution of (8). Since for each $p \in [\mathcal{X}]_2$, d_p is a lower bound to all elements in $I_{\mathcal{R}}[R]^p$, all constraints of (8) are satisfied. Therefore, $\underline{R}$ is a solution. By induction on the definition of the sets $I_{\mathcal{R}}[R]^p$, any other solution $R' = \langle r'_p \rangle_{[\mathcal{X}]_2}$ consists of lower bounds of these sets, i.e., $r'_p \sqsubseteq \bigsqcap I_{\mathcal{R}}[R]^p = d_p$ – implying our claim. To conclude statement (2), it remains to prove that the greatest solution $\underline{R}$ can be reached by Kleene iteration. For every p, d_p is an element of the set $I_{\mathcal{R}}[R]^p$, and therefore, has arrived there after finitely many applications of the inductive rule of their definitions. Let h be an upper bound to these numbers for all $d_p, p \in [\mathcal{X}]_2$. Then, Kleene iteration for the constraint system (8) will also reach these values after at most h iterations.

For the reverse direction, assume that Kleene iteration for the greatest solution of (8) terminates after h iterations with a collection $\underline{R} = \langle d_p \rangle_{p \in [\mathcal{X}]_2}$. By induction on the number j of rounds, we find each value $d_p^{(j)}$ attained for r_p, $p \in [\mathcal{X}]_2$, after j rounds, is an element of $I_{\mathcal{R}}[R]^p$. Therefore, $d_p = d_p^{(h)} \in I_{\mathcal{R}}[R]^p$ for all p. It remains to prove that d_p is also a lower bound of $I_{\mathcal{R}}[R]^p$. To show this, we again proceed by induction, this time on the number i of applications of the inductive rule for the construction of the $I_{\mathcal{R}}[R]^p$, and prove that for all i and any value d' added to some set $I_{\mathcal{R}}[R]^p$ in the ith step, it holds that $d_p^{(i)} \sqsubseteq d'$. Therefore, d_p is a lower bound to $I_{\mathcal{R}}[R]^p$ for all p, and statement (1) follows. □

If all operations on abstract relations $r \in \mathcal{R}^Y$ for clusters Y of size at most 3 are constant time and the height of all $\mathcal{R}[R]^p$ are bounded by h, then the greatest solution of the constraint system (8) can be computed in time polynomial in h and the number of variables.

We call a relational domain 2-*nice*, if the statements of Proposition 1 are satisfied for each collection $R = \langle s_p \rangle_{p \in [\mathcal{X}]_2}$ with $s_p \in \mathcal{R}^p$.

Let us instantiate this construction to 2-disjunctive constants. First, we note that the relational domain $\mathcal{C}[U]$ is finite and thus, in particular, 2-nice. Let $\Psi = \langle s_p \rangle_{p \in [\mathcal{X}]_2}$ denote a collection with $s_p \in \mathcal{C}[U]^p$ for all p. Assume that $\mathcal{X}$ consists of n variables, and let m be the number of constants occurring in any of the s_p. According to the normal form (7), the lattice $I_{\mathcal{C}[U]}[\Psi]^p$ has height at most m if p consists of a single variable, and height bounded by m^2 if p is a two-element set. Since there are $\frac{1}{2}n(n+1)$ clusters, fixpoint iteration will terminate after $\mathcal{O}(n^2 \cdot m^2)$ updates. □
Due to NP-hardness of satisfiability, we cannot expect the greatest solution of the constraint system for 2-disjunctive constants to always return the exact normal form. For the formula from Example 3, e.g., it returns for each pair $\{x_i, x_j\} \in E$, $i \neq j$,

$$(x_i = a \wedge x_j \in \{b, c\}) \vee (x_i = b \wedge x_j \in \{a, c\}) \vee \\ (x_i = c \wedge x_j \in \{a, b\})$$

– which is different from $\perp$.

For a relational domain $\mathcal{R}$, we call a collection $R = \langle r_p \rangle_{p \in [\mathcal{X}]_2}$ with $r_p \in \mathcal{R}^p$ for all p, *stable* if it is a solution of the constraint system (8) with $s_p \equiv r_p$. We remark that stability of R implies that, if $r_p = \perp$ for some p, then $r_{p'} = \perp$ for all other $p' \in [\mathcal{X}]_2$ as well. Now we introduce for a relational domain $\mathcal{R}$ the domain $\mathcal{R}_2^\sharp$ of all stable collections. The ordering $\sqsubseteq^\sharp$ on the domain $\mathcal{R}_2^\sharp$ is defined by $R \sqsubseteq^\sharp R'$ if $r_p \sqsubseteq r'_p$ for all $p \in [\mathcal{X}]_2$ when $R = \langle r_p \rangle_{p \in [\mathcal{X}]_2}$ and $R' = \langle r'_p \rangle_{p \in [\mathcal{X}]_2}$. Thus, $(\bigsqcap R) \sqsubseteq (\bigsqcap R')$ whenever $R \sqsubseteq^\sharp R'$.

Abstract join as well as abstract restriction for $\mathcal{R}_2^\sharp$ then is modeled along the definitions of join and restriction for $\mathcal{R}_2$, but refers to the representation as solution to the constraint system (8). For $R = \langle r_p \rangle_{p \in [\mathcal{X}]_2}$, $R' = \langle r'_p \rangle_{p \in [\mathcal{X}]_2}$ in $\mathcal{R}_2^\sharp$, we define the abstract join by

$$R \sqcup^\sharp R' = \langle r_p \sqcup r'_p \rangle_{p \in [\mathcal{X}]_2}$$

while for $Y \subseteq \mathcal{X}$, and $R = \langle r_p \rangle_{p \in [\mathcal{X}]_2}$, we define abstract restriction by

$$\begin{aligned} \langle r_p \rangle_{p \in [\mathcal{X}]_2}\big|_Y^\sharp &= \langle r_p|_Y \rangle_{p \in [\mathcal{X}]_2} \\ &= \langle r_p|_{Y \cap p} \rangle_{p \in [\mathcal{X}]_2} \end{aligned}$$

where the latter equality follows since for $r_p \in \mathcal{R}^p$, $r_p|_p = r_p$. We have:

Proposition 2 *Assume that $\mathcal{R}$ is 2-nice and satisfies* (6). *Then we have:*

1. *For each $R, R' \in \mathcal{R}_2^\sharp$, also $R \sqcup^\sharp R'$ is again in $\mathcal{R}_2^\sharp$ and is the least upper bound of R, R'. Moreover,*
$$(\bigsqcap R) \sqcup (\bigsqcap R') \sqsubseteq \bigsqcap(R \sqcup^\sharp R')$$
2. *For each $R \in \mathcal{R}_2^\sharp$ and $Y \subseteq \mathcal{X}$, $R|^\sharp_Y$ is again in $\mathcal{R}_2^\sharp$ where*
$$(\bigsqcap R)\Big|_Y \sqsubseteq \bigsqcap(R|^\sharp_Y)$$
holds.
3. *For each $R = \langle r_p\rangle_{p\in[\mathcal{X}]_2}$,$R' = \langle r'_p\rangle_{p\in[\mathcal{X}]_2}$ in $\mathcal{R}_2^\sharp$, the greatest lower bound $R \sqcap^\sharp R' = \langle r''_p\rangle_{p\in[\mathcal{X}]_2}$ is determined as the greatest solution of* (8) *with start values $s_p = r_p \sqcap r'_p$ ($p \in [\mathcal{X}]_2$).*

Proof For the first statement, let $R = \langle r_p\rangle_{p\in[\mathcal{X}]_2}$ and $R' = \langle r'_p\rangle_{p\in[\mathcal{X}]_2}$. As the ordering on $\mathcal{R}_2^\sharp$ is componentwise, it suffices to prove that $R \sqcup^\sharp R'$ is again in $\mathcal{R}_2^\sharp$, i.e., the collection $r_p \sqcup r'_p, p \in [\mathcal{X}]_2$, is a solution of the constraints in (8). For this, we calculate:

$$\begin{aligned} r_{\{x,y\}} &\sqcup r'_{\{x,y\}} \\ &\sqsubseteq (r_{\{x,z\}} \sqcap r_{\{z,y\}})\big|_{\{x,y\}} \sqcup (r'_{\{x,z\}} \sqcap r'_{\{z,y\}})\Big|_{\{x,y\}} \\ &\sqsubseteq ((r_{\{x,z\}} \sqcup r'_{\{x,z\}}) \sqcap (r_{\{z,y\}} \sqcup r'_{\{z,y\}}))\Big|_{\{x,y\}} \end{aligned}$$

for all variables $x, y, z \in \mathcal{X}$. From that, the statement follows.

To prove the second statement, we must verify that the collection $r_p|_{Y\cap p}, p \in [\mathcal{X}]_2$ satisfies all constraints in (8). Indeed, we find by monotonicity,

$$\begin{aligned} r_{\{x,y\}}\big|_Y &\sqsubseteq (r_{\{x,z\}} \sqcap r_{\{z,y\}})\big|_{\{x,y\}\cap Y} \\ &\sqsubseteq (r_{\{x,z\}}\big|_Y \sqcap r_{\{z,y\}}\big|_Y)\big|_{\{x,y\}\cap Y} \end{aligned}$$

for all $x, y, z \in \mathcal{X}$, and the claim follows. The final statement then follows from the definition. □

Elements of $\mathcal{R}_2^\sharp$ are collections $\langle r_p\rangle_{p\in[\mathcal{X}]_2}$. For every $p \in [\mathcal{X}]_2$, we can consider elements $r_p \in \mathcal{R}^p$ as elements of $\mathcal{R}_2^\sharp$ as well by assuming that r_p represents the stable collection $\langle r_p|_q\rangle_{q\in[\mathcal{X}]_2}$.

According to Proposition 2, both joins and restrictions can be computed componentwise. As a consequence, we find:

Theorem 3 *For a 2-nice relational domain $\mathcal{R}$ which satisfies* (6)*, the domain $\mathcal{R}_2^\sharp$ is a 2-decomposable relational domain.* □

Fig. 1 shows the abstract relational domains $\mathcal{R}, \mathcal{R}_2$, and $\mathcal{R}_2^\sharp$ together with the mappings between them. According to Theorem 3, the domain $\mathcal{C}_2^\sharp[U]$ of abstract 2-disjunctive constants is indeed 2-decomposable. The given construction provides us with polynomial algorithms for least upper bound, greatest lower bound, and projection.

$$\begin{tikzcd} \boxed{\mathcal{R}} \arrow[r, shift left, "\overline{(.)}"] & \boxed{\mathcal{R}_2} \arrow[l, shift left, "\mathsf{id}"] \arrow[r, shift left, "\langle\cdot|_p\rangle_{p\in[\mathcal{X}]_2}"] & \boxed{\mathcal{R}_2^\sharp} \arrow[l, shift left, "\bigsqcap\cdot"] \end{tikzcd}$$

Fig. 1: The relationship between abstract relational domains.

3.2 *Assignments*

Let us return to the relational domain $\mathcal{C}_2[U]$ of 2-disjunctive constants and indicate how abstract transformers for assignments $x := s$ can be tailored. For 2-disjunctive constants, we only consider right-hand sides s where s is either ? (unknown value), or of the form $A|y_1|\ldots|y_k$ where A is a set of constants and $y_1, \ldots, y_k \in \mathcal{X}$ are variables. The concrete semantics of such an assignment is given by

$$\begin{aligned} [\![x := ?]\!]\,\Sigma &= \{\sigma \oplus \{x \mapsto c\} \mid \sigma \in \Sigma, c \in U\} \\ [\![x := A|y_1|\ldots|y_k]\!]\,\Sigma &= \{\sigma \oplus \{x \mapsto a\} \mid \sigma \in \Sigma, a \in A\} \cup \\ &\quad \textstyle\bigcup_{j=1}^k \{\sigma \oplus \{x \mapsto \sigma\, y_j\} \mid \sigma \in \Sigma\} \end{aligned}$$

Generalizing the corresponding abstract semantics for (copy) constant propagation, we define the logic transformer for $\mathcal{C}_2[U]$ by

$$\begin{aligned} [\![x := ?]\!]_2\,\Psi &= \Psi|_{\mathcal{X}\setminus\{x\}} \\ [\![x := A|y_1|\ldots|y_k]\!]_2\,\Psi &= (x \in A) \wedge \Psi|_{\mathcal{X}\setminus\{x\}} \sqcup_2 \\ &\quad \textstyle\bigsqcup_2{}_{j=1}^{k}\ [\![x := y_j]\!]_2\,\Psi \end{aligned}$$

Proposition 3 1. *The logic transformer $[\![x := ?]\!]_2$ is precise, i.e.,*
$$[\![x := ?]\!]\,(\gamma\,\Psi) = \gamma\,([\![x := ?]\!]_2\,\Psi) \tag{9}$$
In particular, it is distributive and commutes with $\bot$.
2. *The logic transformer $[\![x := A \mid y_1|\ldots|y_k]\!]_2$ is precise, if the logic transformers for $x := y_j$, $j = 1, \ldots, k$, are.*

Thus, we have reduced the construction of logic transformers for assignments to restriction and the construction of logic transformers for variable-variable assignments $x := y$. For $y \equiv x$, the assignment is the identity, i.e., we set $[\![x := x]\!]_2\,\Psi = \Psi$. Therefore, assume that y is different from x, and assume that $\Psi|_{\mathcal{X}\setminus\{x\}} = \Psi'$. Let B denote the set of constants so that $\Psi'|_{\{y\}}$ equals $y \in B$. Let Ψ_y denote the conjunction of all formulas $\Psi'|_p$ for $p \in [\mathcal{X}]_2$ with $y \in p$. Let $\Psi'' = \Psi_y[x/y]$ denote the formula obtained from Ψ_y by renaming each occurrence of the variable y with x. Then we define

$$[\![x := y]\!]_2\,\Psi = \Psi' \wedge \left(\bigvee_{a\in B} x = a \wedge y = a\right) \wedge \Psi''$$

Let $\bar{\Psi}$ denote the formula returned by that transformer for Ψ. Intuitively, our definition means for $x \notin p$, that $\bar{\Psi}\big|_p = \Psi\big|_p$, i.e., $\Psi\big|_p$ is preserved while additionally, $\bar{\Psi}\big|_{\{x\}} = \Psi\big|_{\{y\}}[x/y]$, $\bar{\Psi}\big|_{\{x,y\}} = \bigvee_{a\in B} x = b \wedge y = b$, and for $z \notin \{x, y\}$, $\bar{\Psi}\big|_{\{x,z\}} = \Psi\big|_{\{y,z\}}[x/y]$.

Proposition 4 *The logic transformer $[\![x := y]\!]_2$ is precise, i.e.,*

$$[\![x := y]\!]\,(\gamma\,\Psi) = \gamma\,([\![x := y]\!]_2\,\Psi) \tag{10}$$

holds. □

The same construction allows us to construct *abstract* logic transformers $[\![x := s]\!]_2^\sharp : C_2^\sharp[U] \to C_2^\sharp[U]$ – only that the least upper bound operation and projection of $\mathcal{C}_2[U]$ must be replaced by the corresponding operations of $\mathcal{C}_2^\sharp[U]$. The abstract transformer then, however, is only *sound* and no longer *precise*, since the projection operation of $\mathcal{C}_2^\sharp[U]$ may return for an abstract relation R whose concretization is *empty* an abstract relation with a *non-empty* concretization. Accordingly, Eq. (9) and Eq. (10) may be violated.

3.3 *Guards*

It remains to provide the semantics of guards. Again, we first consider the domain $\mathcal{C}_2[U]$ of 2-disjunctive formulas (modulo logical equivalence), ordered by implication. We consider positive guards of the form $x \in A$, and conversely, negative guards of the form $x \notin A$. Positive guards thus can directly be expressed in $\mathcal{C}_2[\mathcal{U}]$. Thus we set

$$[\![?(x \in A)]\!]\,\Psi = \Psi \wedge (x \in A) \tag{11}$$

Negative guards on the other hand cannot be directly expressed in $\mathcal{C}_2[U]$ – at least if there are unknown constant values beyond the finite universe U. To deal with this, we introduce a dedicated fresh symbol $\bullet \notin U$ with the understanding that $\bullet$ repesents any value $a \notin U$. The property $x \notin A$ then can equivalently be represented by

$$x \in (U \cup \{\bullet\})\backslash A$$

allowing us to deal with such co-finite sets of possible values in the same way as we did for finite sets of values alone.

4 Directed Relational Domains

Instead of plain equalities, let us now consider *in*equalities between variables and constants instead of equalities and abandon disjunctions. We will, however, add disjunctions in the end as well. Thus for now, we just consider finite conjunctions of inequalities of the form

$$d \sqsubseteq x, \quad x \sqsubseteq y, \quad \text{or} \quad x \sqsubseteq d$$

for variables $x, y \in \mathcal{X}$ and constant values d. As usual, we consider conjunctions only up to semantic equivalence. We call inequalities of the form $d \sqsubseteq x$ lower bound constraints, and d a lower bound for x. Analogously for upper bounds. Inequalities of the form $x \sqsubseteq y$ are called variable constraints.

Assume we are given a partial order (po), i.e., a set P partially ordered by some relation $\leqslant$. Examples of partial orders of interest are

Subsets. The set 2^U of all subsets of some finite universe U where the ordering is subset inclusion $\subseteq$;

Integers. The set $\mathbb{Z}$ of integers equipped with the natural ordering $\leqslant_\mathbb{Z}$;

Multisets. Multisets, i.e., the set of all mappings $\mu : U \to \mathbb{N}$ from elements in U to their multiplicities ordered by multiset inclusion $\subseteq_\mathcal{N}$.

Strings. The set of all strings Σ^* for some finite alphabet Σ. Several partial orderings are of interest:
- the *prefix* ordering $\leqslant_p$; e.g., ab $\leqslant_p$ abcd;
- the *substring* ordering $\leqslant_s$, e.g., bc $\leqslant_s$ abcde;
- the *scattered substring* ordering $\leqslant_{ss}$, e.g., bd $\leqslant_{ss}$ abcde.

Much more expressive constraints on strings have been studied, e.g., in [1, 8, 11, 13]. In particular, for a fragment containing the prefix ordering, decision procedures are known based on (synchronous) multi-tape finite automata [24]. Due to their expressiveness, these techniques come with a considerable computational effort. Instead, we follow Arceri et al. [3] where basic relational domains are considered for reasoning about variables of string type, sets (of characters), or integers (lengths of strings). Their analyses relate program variables only according to some partial order, and also consider lower bounds. Here, these considerations are complemented by taking *upper bounds* into account as well and, eventually, by adding disjunctions.

A mapping $\sigma : \mathcal{X} \to P$ is a model of Ψ (relative to P), written as $\sigma \models \Psi$, if $\Psi \neq \bot$, and

- $d \leqslant \sigma\,x$ (in P) for each constraint $d \sqsubseteq x$ in Ψ;
- $\sigma\,x \leqslant d$ (in P) for each constraint $x \sqsubseteq d$ in Ψ; and
- $\sigma\,x \leqslant \sigma\,y$ (in P) for each constraint $x \sqsubseteq y$ in Ψ.

Let $\mathcal{D}[P]$ denote all finite conjunctions over P modulo semantic equivalence where the ordering on $\mathcal{D}[P]$ is semantic implication. As before, normal forms of conjunctions will be considered up to reordering of atomic propositions. Thus, syntactic equality of conjunctions here means equality of the respective *sets* of propositions. Let Ψ denote a finite conjunction where $V \subseteq P$ is the set of values occurring in Ψ as lower or upper bounds. To provide a first normal form for Ψ, we proceed in two steps. First, we determine the transitive closure $(\leqslant \cup \sqsubseteq)^+$ on the set $\mathcal{X} \cup V$ of the constraints provided by Ψ. In case that $(a, b) \in (\leqslant \cup \sqsubseteq)^+$ for $a, b \in V$ where $a \leqslant b$ does not hold in P, then Ψ is unsatisfiable and therefore represented by the dedicated element $\Psi' = \bot$. If this is not the case, let Ψ' denote the conjunction of all inequalities $s_1 \sqsubseteq s_2$ where $(s_1, s_2) \in (\leqslant \cup \sqsubseteq)^+$ and either s_1 or s_2 or both are in $\mathcal{X}$.

In the second step, when $\Psi' \neq \bot$, we remove all redundant constraints. These are constraints of the form

- $x \sqsubseteq x$ for $x \in \mathcal{X}$, as these constraints hold vacuously;
- $a \sqsubseteq x$ for $a \in V$ and $x \in \mathcal{X}$ if there is also a constraint $b \sqsubseteq x$ with $a \leqslant b$, i.e., there is a stricter lower bound;
- $x \sqsubseteq b$ for $b \in V$ and $x \in \mathcal{X}$ if there is also a constraint $x \sqsubseteq a$ with $a \leqslant b$, i.e., there is a stricter upper bound.

Additionally, we set Ψ' to $\bot$ whenever for some variable x,

- there is no lower bound in P for the set of upper bounds provided for x by Ψ; or
- there is no upper bound in P for the set of lower bounds provided for x by Ψ.

Assume, e.g., that Ψ is given by

$$(\mathsf{abc} \sqsubseteq x) \wedge (\mathsf{abd} \sqsubseteq x)$$

where we consider the prefix order $\leqslant_p$ on strings. Since $\mathsf{abc}, \mathsf{abd}$ cannot be prefixes of the same string, this conjunction is considered equivalent to $\bot$.

Let us denote the resulting conjunction Ψ' by $\mathsf{nf}_0[\Psi]$ and call it the 0-normal form of Ψ. Assuming that comparisons of values as well as checks for common lower or upper bounds are constant-time operations, 0-normal forms can be computed in polynomial time.

4.1 *Lattice Domains*

An important special case is when P is a *lattice*, i.e., a po where every two elements a, b both have a least upper bound $a \vee b$ and a greatest lower bound $a \wedge b$.

Example 4 The po 2^U ordered by subset inclusion is a complete lattice and thus, in particular, a lattice. The integers $\mathbb{Z}$ with the natural ordering is another example of a lattice, this time without *least* or *greatest* element. Yet another example are *multisets*: this lattice has a *least*, but no *greatest* element.

The po Σ^* of strings ordered by the prefix relation is not a lattice. Σ^* provides a least element ϵ, as well as greatest lower bounds, namely, the maximal common prefix, but does not have least upper bounds to all pairs of strings. There is, for example, no upper bound to abc and abd in Σ^*. □

When P is a lattice, we can provide a dedicated normal form which, however, may now use constants from P which did not occur in Ψ before. Assume now that Ψ' is the 0-normal form of Ψ. If P has a least element $\bot_P$, we add the vacuous constraint $\bot_P \sqsubseteq x$ to every variable x. Likewise, if P has a greatest element $\top_P$, we add the constraint $x \sqsubseteq \top_P$.

If Ψ' is different from $\bot$, we subsequently simplify Ψ' further by replacing for each variable $x \in \mathcal{X}$,

- the set of upper bound constraints occurring in Ψ', if it is non-empty and consists of $(x \sqsubseteq b_1) \wedge \ldots \wedge (x \sqsubseteq b_r)$, with the single constraint $(x \sqsubseteq (\bigwedge_{i=1}^{r} b_i))$;
- the set of lower bound constraints in Ψ', if it is non-empty and consists of $(a_1 \sqsubseteq x) \wedge \ldots \wedge (a_r \sqsubseteq x)$, with the single constraint $((\bigvee_{i=1}^{r} a_i) \sqsubseteq x)$.

Let us denote the resulting formula by $\mathsf{nf}_1[\Psi]$ and call it the 1-normal form of Ψ. The 1-normal form of Ψ can be computed in polynomial time as well – given that comparisons as well as pairwise least upper bounds and greatest lower bounds in P are constant time. We have:

Theorem 4 *Assume that the po P is a lattice. Then the following holds:*

1. *A conjunction Ψ is satisfiable over P iff $\mathsf{nf}_1[\Psi] \neq \bot$.*
2. *For arbitrary conjunctions Ψ_1, Ψ_2 over P, $\Psi_1 \implies \Psi_2$ iff $\mathsf{nf}_1[\Psi_1] = \mathsf{nf}_1[\Psi_1 \wedge \Psi_2]$.*

Satisfiability as well as implication are decidable in polynomial time. □

Proof If $\Psi' = \mathsf{nf}_1[\Psi] = \bot$, then Ψ cannot be satisfiable since any of the simplification steps preserves the set of satisfying assignments. So, assume that Ψ' is syntactically different from $\bot$. Let σ be the variable assignment which maps each variable x to its lower bound $a_x \in P$ – if it exists, and to some fixed element $\underline{a}$ which is less or equal to any other lower bound mentioned in Ψ'. Then all single variable constraints are satisfied as well as, by transitivity, all constraints $x \sqsubseteq y$ occurring in Ψ'. Therefore, $\sigma \models \Psi$ – implying that Ψ is satisfiable. From this, statement (1) follows.

To prove statement (2), consider conjunctions Ψ'_1, Ψ'_2 both in 1-normal form. If these syntactically coincide, then obviously also $\Psi'_1 \iff \Psi'_2$ holds. For the reverse direction, we prove that if Ψ'_i are distinct, then they cannot be equivalent. From that, the assertion follows. If one of them equals $\bot$ and the other not, then by statement (1), they cannot be equivalent. Therefore, assume that both are satisfiable and thus, different from $\bot$. We consider all cases how the Ψ_i may differ.

Lower bounds. First, assume that there are constraints $a_i \sqsubseteq x$, $i = 1, 2$, for some variable x in Ψ'_i where a_1 is different from a_2. Assume w.l.o.g. that $a_1 \not\leqslant a_2$ holds. Let L_x denote the set consisting of x together with variables $z \in \mathcal{X}$ where Ψ'_2 has a constraint $z \sqsubseteq x$. Let σ denote some assignment with $\sigma \models \Psi'_2$. Then we construct a variable assignment σ' such that $\sigma' \models \Psi'_2$ but $\sigma' \not\models \Psi'_1$ by

$$\sigma'\, z = \begin{cases} \sigma\, z \wedge a_2 & \text{if } z \in L_x \\ \sigma\, z & \text{otherwise} \end{cases}$$

Then still $\sigma' \models \Psi'_2$. But since $a_1 \not\leqslant a_2$, it follows that σ' does not satisfy $a_1 \sqsubseteq x$ and thus it does not model Ψ'_1.
If there is a constraint $a_1 \sqsubseteq x$ in Ψ'_1, but no lower bound constraint for x in Ψ'_2, then there is some value $\underline{\bot} \in P$ different from a_1 so that $\underline{\bot} \leqslant a_1 \wedge \sigma\, x$ holds. This value allows us to construct an analogous distinguishing assignment σ' where we use $\underline{\bot}$ instead of a_2.

Upper bounds. First, assume that there are constraints $x \sqsubseteq b_i$, $i = 1, 2$, for some variable x in Ψ'_i where b_1 is different from b_2. W.l.o.g., assume that $b_2 \not\leqslant b_1$. Let $U_x \subseteq \mathcal{X}$ denote the subset consisting of x together with all unknowns z where Ψ'_2 has a constraint $x \sqsubseteq z$. Let σ denote some assignment with $\sigma \models \Psi'_2$. Then we construct a variable assignment σ' by:

$$\sigma'\, z = \begin{cases} \sigma\, z \vee b_2 & \text{if } z \in U_x \\ \sigma\, z & \text{otherwise} \end{cases}$$

Then still $\sigma' \models \Psi'_2$ holds. But since $b_2 \not\leqslant b_1$, σ' does not satisfy Ψ'_1.
If there is a constraint $x \sqsubseteq b_1$ in Ψ'_1, but no upper bound constraint for x in Ψ'_2, we introduce a value $\overline{\top} \in P$ which is different from b_1 with $(b_1 \vee \sigma\, x) \leqslant \overline{\top}$, and construct an analogous distinguishing assignment σ' only that we use $\overline{\top}$ instead of b_2.

Variable Constraints. Assume that, w.l.o.g., Ψ'_1 has a constraint $(x \sqsubseteq y)$ for $x, y \in \mathcal{X}$ which does not occur in Ψ'_2 where we assume that for every variable $z \in \mathcal{X}$ both lower and upper bounds are provided by Ψ'_1 iff they are provided by Ψ'_2 and that, whenever they are provided, they agree. Consider again the set U_x of x together with all variables z with constraints $x \sqsubseteq z$, and the set L_y of y together with all variables z with constraints $z \sqsubseteq y$ occurring in Ψ'_2. Since $x \sqsubseteq y$ does not occur in Ψ'_2, $U_x \cap L_y = \emptyset$.
Let σ denote an assignment with $\sigma \models \Psi'_2$. First assume that Ψ'_2 has constraints $x \sqsubseteq b$ and $a \sqsubseteq y$. From $x \sqsubseteq y$ not occurring in Ψ'_2, it follows that $b \not\leqslant a$. Now we construct an assignment σ' by:

$$\sigma'\, z = \begin{cases} b \vee \sigma\, z & \text{if } z \in U_x \cup \{x\} \\ a \wedge \sigma\, z & \text{if } z \in L_y \cup \{y\} \\ \sigma\, z & \text{otherwise} \end{cases}$$

Then $\sigma' \models \Psi'_2$, while $\sigma'\, x = b$ and $\sigma'\, y = a$. As $b \not\leqslant a$, σ' does not fulfill the constraint $x \sqsubseteq y$ from Ψ'_1.
If no upper bound of x is provided, we choose some value b strictly larger than $\sigma\, x \vee \sigma\, y$, and define a variable assignment σ' by $\sigma'\, z = b \vee \sigma\, z$ for $z \in U_x$, and $\sigma'\, z = \sigma\, z$ otherwise. Then $\sigma' \models \Psi'_2$. In order to additionally satisfy $x \sqsubseteq y$, we would have $\sigma'\, x = b \vee \sigma\, x = b \leqslant \sigma'\, y$ – which is impossible.
Likewise, if no lower bound of y is provided, we choose some value a strictly less than $\sigma\, x \wedge \sigma\, y$, and define a variable assignment σ' by $\sigma'\, z = a \wedge \sigma\, z$ for $z \in L_y$, and $\sigma'\, z = \sigma\, z$ otherwise. Then $\sigma' \models \Psi'_2$. In order to additionally satisfy $x \sqsubseteq y$, we would have $\sigma'\, x = \sigma\, x \leqslant \sigma'\, y = a$ – which again is impossible.

□

For lattices, therefore, the construction of normal forms allows deciding satisfiability as well as semantic implication. From our examples, sets, integers, and multisets are lattices. Strings, ordered by the prefix relation, on the other hand, already do not form a lattice anymore. This po, however, is *bounded-complete.* Recall that a po P is bounded-complete if every subset $A \subseteq P$ which has some upper bound, also has a *least* upper bound. When P is bounded-complete, then we at least know that

– every non-empty subset $B \subseteq P$ has a *greatest* lower bound; and
– P has a *least* element $\bot_P$.

Thus, every formula Ψ over a bounded-complete po P which provides some upper bound to every variable $x \in \mathcal{X}$ also can be brought into 1-normal form. Let us call such conjunctions *bounded.* We obtain:

Proposition 5 *Given a po P that is bounded-complete, the following holds:*

1. *A bounded conjunction Ψ is satisfiable over P iff $\mathit{nf}_1[\Psi] \neq \bot$.*
2. *For arbitrary bounded conjunctions Ψ_1, Ψ_2 over P, $\Psi_1 \implies \Psi_2$ iff $\mathit{nf}_1[\Psi_1] = \mathit{nf}_1[\Psi_1 \wedge \Psi_2]$.* □

When we drop the extra assumption that conjunctions are bounded, Proposition 5 need no longer hold.

Example 5 For prefixes of strings, consider the conjunction

$$(\mathsf{ab} \sqsubseteq x) \wedge (x \sqsubseteq \mathsf{abc}) \wedge (\mathsf{abd} \sqsubseteq y) \wedge (x \sqsubseteq y)$$

This formula is semantically equivalent to

$$(\mathsf{ab} \sqsubseteq x) \wedge (x \sqsubseteq \mathsf{ab}) \wedge (\mathsf{abd} \sqsubseteq y) \wedge (x \sqsubseteq y)$$

although the formulas are syntactically different.

Even without upper bounds, not all implications can be inferred via transitive closure alone. Again for prefixes of strings, consider

$$(\mathsf{abc} \sqsubseteq y_1) \wedge (\mathsf{abd} \sqsubseteq y_2) \wedge (x \sqsubseteq y_1) \wedge (x \sqsubseteq y_2) \wedge (\mathsf{ab} \sqsubseteq z)$$

The first four constraints imply that $x \sqsubseteq \text{AB}$, which, by the last constraint, implies that $x \sqsubseteq z$ must hold as well. □

For a conjunction Ψ and a subset $Y \subseteq \mathcal{X}$ of variables, let $\Psi|^\sharp_Y$ yield $\perp$ if Ψ equals $\perp$, and otherwise, yield the conjunction of all constraints in Ψ that only uses variables from Y.

For conjunctions Ψ_1, Ψ_2 in 1-normal form and different from $\perp$, we define the *abstract join* $\Psi_1 \sqcup^\sharp \Psi_2$ as the conjunction of the following constraints:

- all constraints $x \sqsubseteq y$, $x, y \in \mathcal{X}$, which occur both in Ψ_1 and Ψ_2;
- all constraints $(d_1 \wedge d_2) \sqsubseteq x$, $d_1, d_2 \in P$, $x \in \mathcal{X}$ where $d_i \sqsubseteq x$ occurs in Ψ_i;
- all constraints $x \sqsubseteq (d_1 \vee d_2)$, $d_1, d_2 \in P$, $x \in \mathcal{X}$ where $x \sqsubseteq d_i$ occurs in Ψ_i.

Then we have:

Theorem 5 *Assume that P is a lattice.*

1. *If Ψ is a conjunction in 1-normal form, then for every subset $Y \subseteq \mathcal{X}$, $\Psi|_Y$ is given by $\Psi|^\sharp_Y$ where the latter conjunction is again in 1-normal form.*
2. *For Ψ_1, Ψ_2 in 1-normal form, $\Psi_1 \sqcup^\sharp \Psi_2$ is the least upper bound of Ψ_1, Ψ_2 in $\mathcal{D}[P]$.*
3. *The domain $\mathcal{D}[P]$ is a 2-decomposable relational domain.* □

While statement (1) of Theorem 5 remains true also for bounded conjunctions over a bounded-complete po, the least upper bound of two bounded conjunctions need no longer be bounded, as the least upper bounds of the respective upper bounds need not exist. For the prefix ordering on Σ^*, e.g., we have

$$(x \sqsubseteq \mathsf{abc}) \sqcup (x \sqsubseteq \mathsf{abd}) = \top$$

i.e., all information about upper bounds is lost.

4.2 *The General Case*

For general (even finite) partial orders, the dedicated constructions for lattices cannot be directly applied. Already the problem of determining whether or not a conjunction is satisfiable, turns out to be surprisingly difficult. Assume that elements in P can be represented and compared in polynomial time. Then we find:

Theorem 6 *The problem of determining for a given partial order P and a conjunction Ψ, whether Ψ is satisfiable over P, is NP-complete.*

Proof Since a satisfying assignment for a conjunction Ψ can be guessed in polynomial time, it remains to prove the hardness part. For that, consider the problem of 3-colorability of an undirected finite graph $G = (V, E)$. Let $v_1, \ldots, v_n$ be an enumeration of the vertices in V. Then, we construct a partial order P consisting of the elements

$$\begin{aligned}\{\langle v_i, c\rangle \mid i = 1, \ldots, n, c = 1, 2, 3\} &\mathbin{\dot\cup} \{\underline{v}_i \mid i = 1, \ldots, n\}\\ &\mathbin{\dot\cup} \{\overline{v}_i \mid i = 1, \ldots, n\}\end{aligned}$$

where the partial ordering $\leqslant$ of P is the least partial order satisfying

$$\begin{array}{lll}\langle v_i, c\rangle \leqslant \langle v_j, c'\rangle & \text{whenever } \{v_i, v_j\} \in E \wedge i < j \wedge c \neq c' \\ \langle v_i, c\rangle \leqslant \overline{v}_i & \text{whenever } \exists\, j > i.\, \{i, j\} \in E \\ \underline{v}_j \quad\;\; \leqslant \langle v_j, c\rangle & \text{whenever } \exists\, i < j.\, \{i, j\} \in E\end{array}$$

For P, we define a conjunction Ψ in the variables $x_i, i = 1, \ldots, n$, by

$$\textstyle\bigwedge_{\{v_i, v_j\} \in E, i<j} (x_i \sqsubseteq \overline{v}_i) \wedge (x_i \sqsubseteq x_j) \wedge (\underline{v}_j \sqsubseteq x_j)$$

Both P and Ψ can be constructed from G in polynomial time. Moreover, it holds that $\sigma \models \Psi$ iff $\sigma\, x_i = \langle v_i, c_i\rangle$ for some coloring $\gamma : V \to \{1, 2, 3\}$ with $\gamma\, v_i = c_i$. It follows that Ψ is satisfiable iff G has a 3-coloring. In summary, we obtain a polynomial time reduction from the problem of 3-colorability of undirected finite graphs into satisfiability of finite conjunctions over some partial order. This concludes the proof. □.

For general partial orders P, however, we still may rely on the 0-normal form nf_0 and otherwise perform the same constructions as we did for lattices with the 1-normal form. Thus, we define an abstract ordering by

$$\Psi_1 \sqsubseteq^\sharp \Psi_2 \qquad \text{iff} \qquad \mathsf{nf}_0[\Psi_1] = \mathsf{nf}_0[\Psi_1 \wedge \Psi_2] \tag{12}$$

Let us denote the resulting abstract domain by $\mathcal{D}[P]_0$. We have:

Theorem 7 *For an arbitrary po P, the following holds:*

1. *If a conjunction Ψ is satisfiable over P then $\mathsf{nf}_0[\Psi] \neq \bot$.*
2. *For all conjunctions Ψ_1, Ψ_2, $\mathsf{nf}_0[\Psi_1] = \mathsf{nf}_0[\Psi_1 \wedge \Psi_2]$ implies that $\Psi_1 \implies \Psi_2$.*

□

For arbitrary po P, we define the abstract projection in the same way as for conjunctions over a lattice P – only that we now rely on formulas in 0-normal form. For such a formula Ψ the projection $\Psi|^{\sharp}_Y$ onto a subset $Y \subseteq \mathcal{X}$ of variables, is again defined by removing all constraints mentioning variables not in Y.

It is for the abstract join operation that we must find a more general definition, since least upper bounds or greatest lower bounds of sets of values in P are no longer at hand. Assume that Ψ_1, Ψ_2 are in 0-normal form and different from $\bot$. Then, we define the abstract join $\Psi_1 \sqcup^{\sharp} \Psi_2$ as the conjunction of the following constraints

– all constraints $x \sqsubseteq y$, $x, y \in \mathcal{X}$, which occur both in Ψ_1 and Ψ_2;
– all constraints $d_i \sqsubseteq x$, $d_1, d_2 \in P$, $x \in \mathcal{X}$ where $d_i \sqsubseteq x$ occurs in Ψ_i for $i = 1, 2$ and $d_i \leqslant d_{3-i}$;
– all constraints $x \sqsubseteq d_i$, $d_1, d_2 \in P$, $x \in \mathcal{X}$ where $x \sqsubseteq d_i$ occurs in Ψ_i for $i = 1, 2$ and $d_{3-i} \leqslant d_i$.

This definition essentially amounts to keeping those ordering constraints between variables in which Ψ_1 and Ψ_2 agree and only keep a lower or upper bound if it is more liberal than a corresponding bound of the other formula.

Example 6 For the po Σ^* with the substring ordering, consider the formulas

$$\Psi_1 = (\mathsf{ab} \sqsubseteq x) \wedge (y \sqsubseteq \mathsf{ab}) \wedge (y \sqsubseteq z)$$
$$\Psi_2 = (\mathsf{abc} \sqsubseteq x) \wedge (y \sqsubseteq \mathsf{abc})$$

Then, according to our definition,

$$\Psi_1 \sqcup^{\sharp} \Psi_2 = (\mathsf{ab} \sqsubseteq x) \wedge (y \sqsubseteq \mathsf{abc})$$

□

With these definitions, the binary operation $\sqcup^{\sharp}$ returns the least upper bound of its arguments w.r.t. the ordering $\sqsubseteq^{\sharp}$. Moreover, $\mathcal{D}[P]_0$ turns into a 2-decomposable relational domain as well.

Theorem 8 *For every po P, $\mathcal{D}[P]_0$ is a 2-decomposable relational domain.* □

4.3 *Directed Domains with Disjunctions*

Subsequently, we extend the relational domain $\mathcal{D}[P]$ for lattices P (resp. $\mathcal{D}[P]_0$ for arbitrary po's) with disjunctions. This extension corresponds to the *disjunctive completion* of $\mathcal{D}[P]$ (resp. $\mathcal{D}[P]_0$) [9]. The elements of the resulting relational domain are disjunctions of normal form conjunctions (1-normal forms if P is a lattice, and 0-normal forms in general) where for $Y \subseteq \mathcal{X}$, the restriction $\Psi|_Y$ of the disjunction Ψ is defined as the disjunction of the restrictions $c|_Y$ of the normal form conjunctions c contained in Ψ. By definition, restrictions therefore are distributive. Let $\overline{\mathcal{D}}[P]$ (resp. $\overline{\mathcal{D}}[P]_0$) denote the resulting relational abstract domains. If P is infinite, these relational domains have infinite strictly ascending chains, and therefore must have also strictly descending chains of unbounded length. For the lattice $\mathbb{Z}$, e.g., there are even *infinite* strictly descending chains, e.g.,

$$(0 \sqsubseteq x),\ (1 \sqsubseteq x),\ (2 \sqsubseteq x),\ \ldots$$

Nonetheless, we have:

Proposition 6 *1. For every po P, $\overline{\mathcal{D}}[P]_0$ is 2-nice.*
2. For every lattice P, $\overline{\mathcal{D}}[P]$ is 2-nice.

Proof Let D denote an arbitrary collection $\langle d_p \rangle_{p \in [\mathcal{X}]_2}$ with $d_p \in \overline{\mathcal{D}}[P]^p_0$. Consider an arbitrary formula d'_p from the set $I_{\overline{\mathcal{D}}[P]}[D]^p$. It consists of disjunctions of conjunctions each of which may only mention variables from p or constants occurring in any of the $d_{p'}, p' \in [\mathcal{X}]_2$. Since the number of these formulas is finite, statement (1) follows.

The proof of the second statement is analogous – only that the occurring constants now may also be finite meets of constants occurring in upper-bound constraints of the initial collection or finite joins of constants occurring in lower-boudn constraints. Still, the number of possible formulas remains finite. □

Due to Proposition 6, the construction from Section 3 can be applied resulting in the 2-decomposable relational domains $\overline{\mathcal{D}}^{\sharp}_2[P]$ (in case of lattices P) and $\overline{\mathcal{D}}^{\sharp}_2[P]_0$ (for arbitrary pos).

We exemplify the construction for the lattice $\mathbb{Z}$ of integers, i.e., for $\overline{\mathcal{D}}^{\sharp}_2[\mathbb{Z}]$. One-variable properties expressible in this lattice are disjunctions of interval constraints such as

$$(x \sqsubseteq 3) \vee (5 \sqsubseteq x) \wedge (x \sqsubseteq 7)$$

Two-variable properties expressible in this lattice are, e.g.,

$$\begin{array}{l}(x \sqsubseteq -1) \wedge (x \sqsubseteq y) \ \vee \\ (0 \sqsubseteq x) \wedge (x \sqsubseteq 5) \wedge (2 \sqsubseteq y) \ \vee \\ (6 \sqsubseteq x) \wedge (y \sqsubseteq x) \wedge (y \sqsubseteq 19)\end{array}$$

Arbitrary elements in $\overline{D}^{\sharp}_2[\mathbb{Z}]$ can be understood as representations of *conjunctions* of such properties.

Assume that we are given a collection $Z = \langle s_p \rangle_{p \in [\mathcal{X}]_2}$ with $s_p \in \overline{\mathcal{D}}[\mathbb{Z}]^p$ – which is not yet *stable*, and we would like to determine the corresponding stable collection by performing a fixpoint iteration to determine the greatest solution of Eq. (8). During that iteration, we only need to consider upper and lower bounds for each variable x which have already occurred in the formulas s_p. Therefore, the length of each intermediate formula is bounded by a polynomial in the input, and each unknown r_p is updated only polynomially often. As a consequence, all operations abstract join, abstract meet and abstract projection for $\overline{\mathcal{D}}_2^\sharp[\mathbb{Z}]$ are polynomial. For arbitrary lattice or po P, we may proceed analogously. Efficiency of the fixpoint iteration, though, remains to be checked separately for every P.

4.4 *Assignments*

Let us turn to the construction of abstract transformers for assignments. We only describe these for the relational domains $\mathcal{D}[P]$ and $\mathcal{D}[P]_0$, respectively. We first consider three simple cases: assignments of unknown values; assignments of constants; and copying one variable into the other.

$$\begin{aligned}
[\![x := ?]\!]^\sharp \Psi &= \Psi|^\sharp_{\mathcal{X}\backslash\{x\}} \\
[\![x := d]\!]^\sharp \Psi &= \Psi|^\sharp_{\mathcal{X}\backslash\{x\}} \wedge (d \sqsubseteq x) \wedge (x \sqsubseteq d) \\
[\![x := y]\!]^\sharp \Psi &= \Psi|^\sharp_{\mathcal{X}\backslash\{x\}} \wedge (x \sqsubseteq y) \wedge (y \sqsubseteq x)
\end{aligned} \tag{13}$$

for $d \in P$ and $x, y \in \mathcal{X}$ with $x \not\equiv y$. Again, we realize the assignment of unknown values by restriction. For assigning constants and variables, we remark that equality can be expressed via a pair of inequalities.

Individual partial orders, though, may support further forms of right-hand sides in assignments. Subsequently, we enumerate more general forms of assignments for sets and for the prefix, substring, and scattered substring partial orders on strings.

Sets. For sets, we consider right-hand sides of the form $y_1 \cap y_2$ or $y_1 \cup y_2$ for $y_1, y_2 \in \mathcal{X}$ with $x \notin \{y_1, y_2\}$. We define

$$\begin{aligned}
[\![x := y_1 \cap y_2]\!]^\sharp \Psi &= \Psi|^\sharp_{\mathcal{X}\backslash\{x\}} \wedge (x \sqsubseteq y_1) \wedge (x \sqsubseteq y_2) \\
[\![x := y_1 \cup y_2]\!]^\sharp \Psi &= \Psi|^\sharp_{\mathcal{X}\backslash\{x\}} \wedge (y_1 \sqsubseteq x) \wedge (y_2 \sqsubseteq x)
\end{aligned}$$

Thus, we obtain after the assignment as new upper (lower) bounds of x in terms of the variables y_1 and y_2. An analogous construction can also be applied to multisets. We remark that the given right-hand sides do *not* entail that the equalities $x = y_1 \cap y_2$ and $x = y_1 \cup y_2$, respectively, hold after the assignments.

Prefixes. In this case, right-hand sides of interest are concatenations of a constant or variable, possibly followed by some further value, i.e., are of the form $s\,?$ for s either in Σ^*, or in $\mathcal{X}\backslash\{x\}$, with "?" again denoting unknown input. We define

$$[\![x := s\,?]\!]^\sharp \Psi = \Psi|^\sharp_{\mathcal{X}\backslash\{x\}} \wedge (s \sqsubseteq x)$$

i.e., we only obtain information about lower bounds for x after the assignment but lose all information about upper bounds.

Substrings. Again, we consider right-hand sides which are concatenations of constants or variables with further values. These now are of the form $?\,s_1\,?\ldots?\,s_k\,?$ $(s_i \in \Sigma^* \cup \mathcal{X}\backslash\{x\})$. We define

$$\begin{aligned}
[\![x := ?\,s_1\,?\ldots?\,s_k\,?]\!]^\sharp \Psi = \Psi|^\sharp_{\mathcal{X}\backslash\{x\}} \wedge \\
(s_1 \sqsubseteq x) \wedge \ldots \wedge (s_k \sqsubseteq x)
\end{aligned}$$

For scattered substrings, we proceed similarly. In both cases, no information is obtained for upper bounds to the left-hand side variable x after the assignment.

So far, we have assumed that the right-hand side s does not contain the variable x from the left-hand side. In case that x occurs in s, we split the assignment into the sequence

tmp := s; x := tmp;

for some fresh variable tmp, i.e., first store the value of the right-hand side s in tmp whose value only then is assigned to the left-hand side variable x.

These abstract tranformers for the relational domains $\mathcal{D}[P]$ (resp. $\mathcal{D}[P]_0$) are readily lifted to corresponding transformers for the weakly relational domains $\overline{\mathcal{D}}_2^\sharp[P]$ (resp. $\overline{\mathcal{D}}_2^\sharp[P]_0$).

4.5 *Guards and Negated Inequalities*

Let us now turn to a treatment of guards $?c$ for the directed domain $\overline{\mathcal{D}}_2^\sharp[P]$ where P is a lattice. The case for $\overline{\mathcal{D}}_2^\sharp[P]_0$ (when P is not a lattice) is analogous.

A condition c which consists of an inequality $s_1 \sqsubseteq s_2$ for s_i being variables or constants already represents an abstract relation. Therefore, Eq. (2) can be used to define the abstract effect of $[\![?c]\!]^\sharp$.

If the condition c is a *negated* inequality $s_1 \not\sqsubseteq s_2$, this is not immediately possible. Assume that the variables occurring in c all occur in $p \in [\mathcal{X}]_2$. Now consider an arbitrary element $D = \langle d_{p'} \rangle_{p' \in [\mathcal{X}]_2}$. In particular, $d_p \in \overline{\mathcal{D}}[P]^p$, i.e., $d_p = e_1 \vee \ldots \vee e_k$ for conjunctions $e_1, \ldots, e_k$ all using variables from p only. In this case, we define

$$[\![?c]\!]^\sharp D = D \sqcap \bigvee\{e_j \mid e_j \;\not\Longrightarrow\; (s_1 \sqsubseteq s_2)\}$$

Thus, the negated inequality c allows to improve the abstract relation D by possibly removing those conjuncts e_j from d_p which contradict c.

5 Conclusion

We considered a construction of 2-decomposable relational domains from arbitrary relational domains and exemplified this construction by deriving 2-disjunctive constants from the relational domain of disjunctive constants. For 2-disjunctive constants, it turned out that normalization is prohibitively expensive. Therefore, we provided a second general construction of 2-decomposable relational domains, now based on greatest solutions of constraint systems, which – in the case of disjunctive constants – results in a 2-decomposable domain where the operations join, meet, and restriction are polynomial.

In the second part, we then considered directed domains as conjunctions of inequalities over lattices or general partial orders. For lattices, we provided the 1-normal form for a syntactic characterization of semantic equivalence. We showed that the resulting domain is 2-decomposable and provided precise polynomial algorithms for 1-normalization, projection, join, and meet. For arbitrary partial orders, we use a weaker form of normalization for constructing a weaker 2-decomposable relational domain, for which we again provided polynomial algorithms, now for 0-normalization, projection, join, and meet. Only in the very last step, we added disjunctions by applying the general construction of 2-decomposable domain based on approximate normalization from the previous section. Both for 2-disjunctive constants and for directed domains, we indicated how transfer functions for assignments and guards can be constructed.

Our results can be extended in several directions. In the case of constants, one may, e.g., additionally, track equalities as well as disequalities between variables; likewise for directed domains, an extensive study of the impact of negated inequalities could be of interest. Here, we only studied lattice operations and transfer functions. Directed domains, though, may have infinite strictly ascending chains. Therefore, tailored widening and narrowing operators are of interest when these domains are employed for practical static analysis.

Acknowledgements. This work has been supported by Shota Rustaveli National Science Foundation of Georgia under the project FR-21-7973 and by Deutsche Forschungsgemeinschaft (DFG) – 378803395/2428 ConVeY.

References

1. Abdulla, P.A., Atig, M.F., Diep, B.P., Holík, L., Janku, P.: Chain-free string constraints. In: Chen, Y., Cheng, C., Esparza, J. (eds.) Automated Technology for Verification and Analysis - 17th International Symposium, ATVA 2019, Taipei, Taiwan, October 28-31, 2019, Proceedings, *Lecture Notes in Computer Science*, vol. 11781, pp. 277–293. Springer (2019). URL `https://doi.org/10.1007/978-3-030-31784-3_16`
2. Albert, E., Arenas, P., Genaim, S., Puebla, G., Román-Díez, G.: Conditional termination of loops over heap-allocated data. Sci. Comput. Program. **92**, 2–24 (2014). URL `https://doi.org/10.1016/j.scico.2013.04.006`
3. Arceri, V., Olliaro, M., Cortesi, A., Ferrara, P.: Relational string abstract domains. In: Finkbeiner, B., Wies, T. (eds.) Verification, Model Checking, and Abstract Interpretation - 23rd International Conference, VMCAI 2022, Philadelphia, PA, USA, January 16-18, 2022, Proceedings, *Lecture Notes in Computer Science*, vol. 13182, pp. 20–42. Springer (2022). URL `https://doi.org/10.1007/978-3-030-94583-1_2`
4. Bagnara, R., Hill, P.M., Zaffanella, E.: An improved tight closure algorithm for integer octagonal constraints. In: Logozzo, F., Peled, D.A., Zuck, L.D. (eds.) Verification, Model Checking, and Abstract Interpretation, pp. 8–21. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
5. Bagnara, R., Hill, P.M., Zaffanella, E.: Weakly-relational shapes for numeric abstractions: improved algorithms and proofs of correctness. Formal Methods Syst. Des. **35**(3), 279–323 (2009). URL `https://doi.org/10.1007/s10703-009-0073-1`
6. Beckert, B., Hähnle, R., Manyà, F.: The 2-sat problem of regular signed CNF formulas. In: 30th IEEE International Symposium on Multiple-Valued Logic, ISMVL 2000, Portland, Oregon, USA, May 23-25, 2000, Proceedings, pp. 331–336. IEEE Computer Society (2000). URL `https://doi.org/10.1109/ISMVL.2000.848640`
7. Chawdhary, A., Robbins, E., King, A.: Incrementally closing octagons. Formal Methods Syst. Des. **54**(2), 232–277 (2019). URL `https://doi.org/10.1007/s10703-017-0314-7`
8. Chen, T., Chen, Y., Hague, M., Lin, A.W., Wu, Z.: What is decidable about string constraints with the replaceall function. Proc. ACM Program. Lang. **2**(POPL), 3:1–3:29 (2018). URL `https://doi.org/10.1145/3158091`

9. Cousot, P., Cousot, R.: Abstract interpretation frameworks. Journal of logic and computation **2**(4), 511–547 (1992)
10. Cousot, P., Halbwachs, N.: Automatic discovery of linear restraints among variables of a program. In: Aho, A.V., Zilles, S.N., Szymanski, T.G. (eds.) Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages, Tucson, Arizona, USA, January 1978, pp. 84–96. ACM Press (1978). URL `https://doi.org/10.1145/512760.512770`
11. Day, J.D., Ganesh, V., Grewal, N., Manea, F.: On the expressive power of string constraints. Proc. ACM Program. Lang. **7**(POPL), 278–308 (2023). URL `https://doi.org/10.1145/3571203`
12. Dor, N., Rodeh, M., Sagiv, S.: Cleanness checking of string manipulations in C programs via integer analysis. In: Cousot, P. (ed.) Static Analysis, 8th International Symposium, SAS 2001, Paris, France, July 16-18, 2001, Proceedings, pp. 194–212. Springer, LNCS 2126 (2001). URL `https://doi.org/10.1007/3-540-47764-0_12`
13. Ganesh, V., Minnes, M., Solar-Lezama, A., Rinard, M.: What is decidable about strings? (2011)
14. Karr, M.: Affine relationships among variables of a program. Acta Informatica **6**, 133–151 (1976). URL `https://doi.org/10.1007/BF00268497`
15. Miné, A.: The octagon abstract domain. In: WCRE' 01, p. 310. IEEE Computer Society (2001). DOI 10.1109/WCRE.2001.957836
16. Miné, A.: Weakly relational numerical abstract domains. (domaines numériques abstraits faiblement relationnels). Ph.D. thesis, École Polytechnique, Palaiseau, France (2004). URL `https://tel.archives-ouvertes.fr/tel-00136630`
17. Miné, A.: The octagon abstract domain. Higher Order Symbol. Comput. **19**(1), 31–100 (2006). URL `https://doi.org/10.1007/s10990-006-8609-1`
18. Müller-Olm, M., Seidl, H.: Precise interprocedural analysis through linear algebra. In: Jones, N.D., Leroy, X. (eds.) Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004, Venice, Italy, January 14-16, 2004, pp. 330–341. ACM (2004). URL `https://doi.org/10.1145/964001.964029`
19. Müller-Olm, M., Seidl, H.: Analysis of modular arithmetic. ACM Trans. Program. Lang. Syst. **29**(5), 29 (2007). URL `https://doi.org/10.1145/1275497.1275504`
20. Sankaranarayanan, S., Sipma, H.B., Manna, Z.: Scalable analysis of linear systems using mathematical programming. In: Cousot, R. (ed.) Verification, Model Checking, and Abstract Interpretation, *LNCS*, vol. 3385, pp. 25–41. Springer, Berlin, Heidelberg (2005)
21. Schwarz, M., Saan, S., Seidl, H., Erhard, J., Vojdani, V.: Clustered relational thread-modular abstract interpretation with local traces. In: Wies, T. (ed.) Programming Languages and Systems - 32nd European Symposium on Programming, ESOP 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings, *Lecture Notes in Computer Science*, vol. 13990, pp. 28–58. Springer (2023). URL `https://doi.org/10.1007/978-3-031-30044-8_2`
22. Schwarz, M., Seidl, H.: Octagons revisited. In: Hermenegildo, M.V., Morales, J.F. (eds.) Static Analysis, pp. 485–507. Springer Nature Switzerland, Cham (2023)
23. Simon, A., King, A., Howe, J.M.: Two variables per linear inequality as an abstract domain. In: Leuschel, M. (ed.) Logic Based Program Synthesis and Transformation, 12th International Workshop, LOPSTR 2002, Madrid, Spain, September 17-20,2002, Revised Selected Papers, *LNCS*, vol. 2664, pp. 71–89. Springer (2002). URL `https://doi.org/10.1007/3-540-45013-0_7`
24. Yu, F., Bultan, T., Hardekopf, B.: String abstractions for string verification. In: Groce, A., Musuvathi, M. (eds.) Model Checking Software - 18th International SPIN Workshop, Snowbird, UT, USA, July 14-15, 2011. Proceedings, *Lecture Notes in Computer Science*, vol. 6823, pp. 20–37. Springer (2011). URL `https://doi.org/10.1007/978-3-642-22306-8_3`