TPLP: Page 1–27.

doi:10.1017/S1471068424000152

Optimal Matching for Sharing and Linearity Analysis

GIANLUCA AMATO and FRANCESCA SCOZZARI
University of Chieti–Pescara, Pescara, Italy
(*e-mails:* gianluca.amato@unich.it, francesca.scozzari@unich.it)

submitted 27 October 2022; revised 12 January 2024; accepted 26 June 2024

Abstract

Static analysis of logic programs by abstract interpretation requires designing abstract operators which mimic the concrete ones, such as unification, renaming, and projection. In the case of goal-driven analysis, where goal-dependent semantics are used, we also need a backward-unification operator, typically implemented through matching. In this paper, we study the problem of deriving optimal abstract matching operators for sharing and linearity properties. We provide an optimal operator for matching in the domain $\texttt{ShLin}^\omega$, which can be easily instantiated to derive optimal operators for the domains $\texttt{ShLin}^2$ by Andy King and the reduced product $\texttt{Sharing} \times \texttt{Lin}$.

KEYWORDS: static analysis, sharing, linearity, matching

1 Introduction

In the field of static analysis of logic programs, sharing information is one of the most interesting and widely used property. The goal of sharing analysis is to detect sets of variables which share a common variable. For instance, in the substitution $\{x/f(z,a), y/g(z)\}$, the variables x and y share the common variable z. Sharing may also track and infer groundness in the same way as the `Def` domain (de la Banda and Hermenegildo, 1993; Armstrong et al., 1998). Typical applications of sharing analysis are in the fields of optimization of unification (Søndergaard, 1986) and parallelization of logic programs (Hermenegildo and Rossi, 1995).

It is widely recognized that the pioneering abstract domain `Sharing` (Langen, 1990; Jacobs and Langen, 1992) is not very precise, so it is often combined with other domains for tracking freeness, linearity, groundness, or structural information (see Bagnara et al. (2005); Codish et al. (1995) for comparative evaluations).

Any domain for static analysis of logic programs must be equipped with four standard operators: renaming, projection, union, and unification. The theory of abstract interpretation (Cousot and Cousot, 1979, 1992a) ensures the existence of the optimal (best correct) abstract operator for each concrete operator. Nevertheless, while finding optimal operators for renaming, projection, and union is trivial most of the time, devising an optimal abstract unification is much harder.

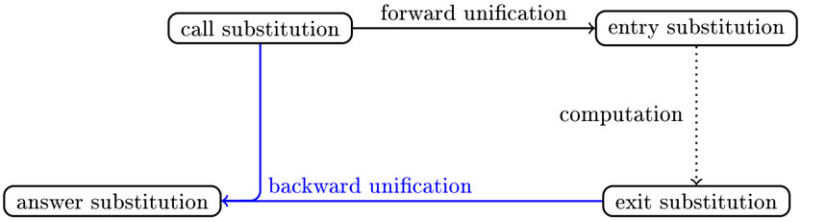

Fig. 1. The role of forward and backward unification in goal-dependent analysis.

Amato and Scozzari (2010, 2014) have proposed a new (infinite) domain $\texttt{ShLin}^\omega$ that precisely represents the interaction between sharing and linearity properties, while discharging other structural, irrelevant information. All the abstract operators for $\texttt{ShLin}^\omega$ are shown to be optimal. From $\texttt{ShLin}^\omega$, the authors derive, for the first time, optimal abstract unification for well-known domains combining sharing and linearity, such as $\texttt{ShLin}^2$ (King, 1994) and $\texttt{Sharing} \times \texttt{Lin}$ (Muthukumar and Hermenegildo, 1992).

In this paper, we extend the $\texttt{ShLin}^\omega$ framework to the case of goal-dependent analysis. In this setting, the unification operator is used twice (see Figure 1):

forward unification: performs parameter passing by unifying the goal and the *call substitution* with the head of the chosen clause. The result is called *entry substitution*.

backward unification: propagates back to the goal the *exit substitution* (i.e., the result of the sub-computation), obtaining the *answer substitution*.[1]

Despite its name, backward unification may be implemented through *matching*, exploiting the property that the exit substitution is always more instantiated than the call substitution. Analyses with matching are strictly more precise than analyses which do not use matching (see Bruynooghe (1991) and Amato and Scozzari (2009) for a thorough discussion of the problem). This idea has been implemented in real abstract interpreters such as GAIA (Le Charlier and Van Hentenryck, 1994) and PLAI (Muthukumar and Hermenegildo, 1992).

However, except for Amato and Scozzari (2009), none of the papers that are based on matching (Bruynooghe, 1991; Hans and Winkler, 1992; Muthukumar and Hermenegildo, 1992; Le Charlier and Van Hentenryck, 1994; King and Longley, 1995) has ever proved optimality of the proposed abstract operators. In particular, there is no known optimal matching operator for any domain combining sharing and linearity.

The lack of optimal operators brings two kinds of disadvantages: first, the analysis obviously loses precision when using suboptimal abstract operators; second, computing approximated abstract objects can lead to a speed down of the analysis. The latter is typical of sharing analysis, where abstract domains are usually defined in such a way that the less information we have, the more abstract objects are complex. This is not the case for other kinds of analyses, such as groundness analysis, where the complexity

[1] We follow Cortesi et al. (1996) for the terminology of *forward* and *backward unification*. Bruynooghe (1991) and Hans and Winkler (1992) use *procedure entry* and *procedure exit*. Muthukumar and Hermenegildo (1991) use *call_to_entry* and *exit_to_success*.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

of abstract objects may grow according to the amount of groundness information they encode. Moreover, knowing the optimal abstract operator, even if we do not plan to implement it, is useful to understand the potentiality and limits of the abstract domain in use and to guide the search for a more precise (or more efficient) domain.

For this reason, in this paper, we define a matching operator for $\texttt{ShLin}^{\omega}$ and prove its optimality. Moreover, from this operator, we derive, for the first time, optimal matching operators for domains combining sharing and linearity information, such as $\texttt{ShLin}^{2}$ and $\texttt{Sharing} \times \texttt{Lin}$.

2 Notations

We fix a first order signature that includes a constant symbol and a function symbol of arity at least two; otherwise, every term has at most one variable, and the structure of terms is trivial (we need this assumption in the proofs of optimality). The signature also includes a denumerable set of variables $\mathcal{V}$. Given a term or other syntactic object o, we denote by $\mathrm{vars}(o)$ the set of variables occurring in o. Given a set A, we denote by $\mathscr{P}(A)$ the powerset of A and by $\mathscr{P}_f(A)$ the set of finite subsets of A.

2.1 Multisets

A *multiset* is a set where repetitions are allowed. We denote by $\{\!\{x_1, \ldots, x_m\}\!\}$ a multiset, where $x_1, \ldots, x_m$ is a sequence with (possible) repetitions, and by $\{\!\{\}\!\}$ the empty multiset. We will often use the polynomial notation $v_1^{i_1} \ldots v_n^{i_n}$, where $v_1, \ldots, v_n$ is a sequence without repetitions, to denote a multiset A whose element v_j appears i_j times. The set $\{v_j \mid i_j > 0\}$ is called the *support* of A and is denoted by $\lfloor\!\lfloor A \rfloor\!\rfloor$. We also use the functional notation $A : \{v_1, \ldots, v_n\} \to \mathbb{N}$, where $A(v_j) = i_j$.

In this paper, we only consider multisets whose support is *finite*. We denote with $\mathscr{P}_m(X)$ the set of all the multisets whose support is *any finite subset* of X. For example, a^3c^5 and $a^3b^2c^1$ are elements of $\mathscr{P}_m(\{a, b, c\})$. The fundamental operation for multisets is the *sum*, defined as

$$A \uplus B = \lambda v \in \lfloor\!\lfloor A \rfloor\!\rfloor \cup \lfloor\!\lfloor B \rfloor\!\rfloor . A(v) + B(v) \ .$$

Note that we also use $\uplus$ to denote disjoint union for standard sets. The context will allow us to discern the proper meaning of $\uplus$. Given a multiset A and $X \subseteq \lfloor\!\lfloor A \rfloor\!\rfloor$, the *restriction* of A over X, denoted by $A_{|X}$, is the only multiset B such that $\lfloor\!\lfloor B \rfloor\!\rfloor = X$ and $B(v) = A(v)$ for each $v \in X$.

2.2 The domain of existential substitutions

We work in the framework of existential substitutions (Amato and Scozzari, 2009), which allows us to simplify those semantic definitions which are heavily based on renaming apart objects and to avoid variable clashes. In this framework, all the details concerning renamings are moved to the inner level of the semantic domain, where they are more easily manageable. We briefly recall the basic definitions of the domain.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

The set of substitutions, idempotent substitutions, and renamings are denoted by *Subst*, *ISubst*, and *Ren*, respectively. Given $\theta_1, \theta_2 \in \mathit{Subst}$ and $U \in \mathscr{P}_f(\mathcal{V})$, the preorder $\preceq_U$ is defined as follows:

$$\theta_1 \preceq_U \theta_2 \iff \exists \delta \in \mathit{Subst}. \forall x \in U.\ \theta_1(x) = \delta(\theta_2(x)) \ .$$

The notation $\theta_1 \preceq_U \theta_2$ states that θ_1 is an instance of θ_2 w.r.t. the variables in U. The equivalence relation induced by the preorder $\preceq_U$ is given by

$$\theta_1 \sim_U \theta_2 \iff \exists \rho \in \mathit{Ren}. \forall x \in U.\ \theta_1(x) = \rho(\theta_2(x)) \ .$$

Let $\mathit{ISubst}_{\sim_U}$ be the quotient set of *ISubst* w.r.t. $\sim_U$. The domain $\mathit{ISubst}_\sim$ of *existential substitutions* is defined as the disjoint union of all the $\mathit{ISubst}_{\sim_U}$ for $U \in \mathscr{P}_f(\mathcal{V})$, namely:

$$\mathit{ISubst}_\sim = \biguplus_{U \in \mathscr{P}_f(\mathcal{V})} \mathit{ISubst}_{\sim_U} \ .$$

In the following, we write $[\theta]_U$ for the equivalence class of θ w.r.t. $\sim_U$. The partial order $\preceq$ over $\mathit{ISubst}_\sim$ is given by

$$[\theta]_U \preceq [\theta']_V \iff U \supseteq V \wedge \theta \preceq_V \theta' \ .$$

Intuitively, $[\theta]_U \preceq [\theta']_V$ means that θ is an instance of θ' w.r.t. the variables in V, provided that they are all variables of interest of θ.

To ease notation, we often omit braces from the sets of variables of interest when they are given extensionally. So we write $[\theta]_{x,y}$ or $[\theta]_{xy}$ instead of $[\theta]_{\{x,y\}}$ and $\sim_{x,y,z}$ instead of $\sim_{\{x,y,z\}}$. When the set of variables of interest is clear from the context or when it is not relevant, it will be omitted. Finally, we omit the braces that enclose the bindings of a substitution when the latter occurs inside an equivalence class; that is, we write $[x/y]_U$ instead of $[\{x/y\}]_U$.

2.2.1 Unification

Given $U, V \in \mathscr{P}_f(\mathcal{V})$, $[\theta_1]_U, [\theta_2]_V \in \mathit{ISubst}_\sim$, the most general unifier between these two classes is defined as the mgu of suitably chosen representatives, where variables not of interest are renamed apart. In formulas:

$$\mathsf{mgu}([\theta_1]_U, [\theta_2]_V) = [\mathsf{mgu}(\theta_1', \theta_2')]_{U \cup V} \ , \tag{1}$$

where $\theta_1 \sim_U \theta_1' \in \mathit{ISubst}$, $\theta_2 \sim_V \theta_2' \in \mathit{ISubst}$, and $(U \cup \mathrm{vars}(\theta_1')) \cap (V \cup \mathrm{vars}(\theta_2')) \subseteq U \cap V$. The last condition is needed to avoid variable clashes between the chosen representatives θ_1' and θ_2'. It turns out that mgu is the greatest lower bound of $\mathit{ISubst}_\sim$ ordered by $\preceq$.

A different version of unification is obtained when one of the two arguments is an existential substitution and the other one is a standard substitution. In this case, the latter argument may be viewed as an existential substitution where all the variables are of interest:

$$\mathsf{mgu}([\theta]_U, \delta) = \mathsf{mgu}([\theta]_U, [\delta]_{\mathrm{vars}(\delta)}) \ . \tag{2}$$

Note that deriving the general unification in (1) from the special case in (2) is not possible. This is because there are elements in $\mathit{ISubst}_\sim$, which cannot be obtained as $[\delta]_{\mathrm{vars}(\delta)}$ for any $\delta \in \mathit{ISubst}$.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

This is the form of unification that is better suited for analysis of logic programs, where existential substitutions are the denotations of programs while standard substitutions are the result of unification between goals and heads of clauses. Devising optimal abstract operators for (2) in different abstract domains is the topic of Amato and Scozzari (2010).

2.2.2 Matching

Given $U_1, U_2 \in \mathscr{P}_f(\mathcal{V})$, $[\theta_1]_{U_1} \in \mathit{ISubst}_\sim$, and $[\theta_2]_{U_2} \in \mathit{ISubst}$, the matching of $[\theta_1]_{U_1}$ with $[\theta_2]_{U_2}$ (Amato and Scozzari, 2009) is defined in the same way as unification, as soon as none of the variables in U_1 get instantiated in the result. If this is not the case, the matching is undefined.

Definition 2.1 (Matching).
Given $[\theta_1]_{U_1}, [\theta_2]_{U_2} \in \mathit{ISubst}_\sim$, we have that

$$\mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2}) = \begin{cases} \mathsf{mgu}([\theta_1]_{U_1}, [\theta_2]_{U_2}) & \text{if } \theta_1 \preceq_{U_1 \cap U_2} \theta_2, \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{3}$$

Note that the condition $\theta_1 \preceq_{U_1 \cap U_2} \theta_2$ is equivalent to $[\theta_1]_{U_1} = \mathsf{mgu}([\theta_1]_{U_1}, [\theta_2]_{U_2})_{|U_1}$ (Amato and Scozzari, 2009).

Example 2.2.
If we unify $[\theta_1]_{x,y} = [x/a, y/b]_{x,y}$ with $[\theta_2]_{y,z} = [z/r(y)]_{y,z}$, we obtain $[\theta_3]_{x,y,z} = [x/a, y/b, z/r(b)]_{x,y,z}$. Note that the variables y and z in θ_3 are instantiated w.r.t. θ_2; therefore, $\mathsf{match}([\theta_2]_{y,z}, [\theta_1]_{x,y})$ is undefined. However, x and y in θ_3 are not instantiated w.r.t. θ_1; therefore, $\mathsf{match}([\theta_1]_{x,y}, [\theta_2]_{y,z}) = [\theta_3]_{x,y,z}$.

Most of the time, when matching is applied in goal-dependent analysis of logic programs, we have that $U_1 \subseteq U_2$. This is because U_1 is the set of variables in a clause, while U_2 contains both the variables in the clause and in the call substitution. Nonetheless, we study here the general case so that it can be applied in any framework.

2.2.3 Other operations

Given $V \in \mathscr{P}_f(\mathcal{V})$ and $[\theta]_U \in \mathit{ISubst}_\sim$, we denote by $([\theta]_U)_{|V}$ the *projection* of $[\theta]_U$ on the set of variables V, defined as:

$$([\theta]_U)_{|V} = [\theta]_{U \cap V} \ . \tag{4}$$

2.3 Abstract interpretation

Given two sets C and A of concrete and abstract objects, respectively, an *abstract interpretation* (Cousot and Cousot, 1992b) is given by an approximation relation $\triangleright \subseteq A \times C$. When $a \triangleright c$ holds, this means that a is a correct abstraction of c. We are interested in the case when $(A, \leq_A)$ is a poset and $a \leq_A a'$ means that a is more precise than a'. In this case, we require that if $a \triangleright c$ and $a \leq_A a'$, then $a' \triangleright c$, too. In more detail, we require what Cousot and Cousot (1992b) call the *existence of the best abstract approximation*

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

assumption; that is, the existence of a map $\alpha : C \to A$ such that for all $a \in A, c \in C$, it holds that $a \triangleright c \iff \alpha(c) \leq_A a$. The map α is called the *abstraction function* and maps each c to its best approximation in A.

Given a (possibly partial) function $f : C \to C$, we say that $\tilde{f} : A \to A$ is a correct abstraction of f, and write $\tilde{f} \triangleright f$, whenever

$$a \triangleright c \Rightarrow \tilde{f}(a) \triangleright f(c) \ ,$$

assuming that $\tilde{f}(a) \triangleright f(c)$ is true whenever $f(c)$ is not defined. We say that $\tilde{f} : A \to A$ is the *optimal* abstraction of f when it is the best correct approximation of f, that is, when $\tilde{f} \triangleright f$ and

$$\forall f' : A \to A.\ f' \triangleright f \Rightarrow \tilde{f} \leq_{A \to A} f' \ .$$

In some cases, we prefer to deal with a stronger framework, in which the domain C is also endowed with a partial order $\leq_C$ and $\alpha : C \to A$ is a left adjoint to $\gamma : A \to C$, that is,

$$\forall c \in C. \forall a \in A. \alpha(c) \leq_A a \iff c \leq_C \gamma(a) \ .$$

The pair $\langle \alpha, \gamma \rangle$ is called a *Galois connection*. In particular, we will only consider the case of *Galois insertions*, which are Galois connections such that $\alpha \circ \gamma$ is the identity map. If $\langle \alpha, \gamma \rangle$ is a Galois insertion and $f : C \to C$ is a monotone map, the optimal abstraction $\tilde{f}$ always exists, and it is definable as $\tilde{f} = \alpha \circ f \circ \gamma$.

3 Abstract matching over ShLin$^\omega$

The domain ShLin$^\omega$ (Amato and Scozzari, 2010) generalizes Sharing by recording multiplicity of variables in sharing groups. For example, the substitution $\theta = \{x/s(u,v), y/g(u,u,u), z/v\}$ is abstracted on Sharing into $\{uxy, vxz\}$, where the sharing group uxy means that $\theta(u)$, $\theta(x)$, and $\theta(y)$ share a common variable, namely u. In ShLin$^\omega$ the same substitution would be abstracted as $\{uxy^3, vxz\}$, with the additional information that the variable u occurs three times in $\theta(y)$. For the sake of completeness, in the following section, we recall the basic definitions.

3.1 The domain ShLin$^\omega$

We call *ω-sharing group* a multiset of variables, that is, an element of $\mathscr{P}_m(\mathcal{V})$. Given a substitution θ and a variable $v \in \mathcal{V}$, we denote by $\theta^{-1}(v)$ the ω-sharing group $\lambda w \in \mathcal{V}.occ(v, \theta(w))$, which maps each variable w to the number of occurrences of v in $\theta(w)$.

Given a set of variables U and a set of ω-sharing groups $S \subseteq \mathscr{P}_m(U)$, we say that $[S]_U$ *correctly approximates* a substitution $[\theta]_W$ if $U = W$ and, for each $v \in \mathcal{V}$, $\theta^{-1}(v)_{|U} \in S$. We write $[S]_U \triangleright [\theta]_W$ to mean that $[S]_U$ correctly approximates $[\theta]_W$. Therefore, $[S]_U \triangleright [\theta]_U$ when S contains all the ω-sharing groups in θ, restricted to the variables in U.

*Definition 3.1 (*ShLin$^\omega$*).*
The domain ShLin$^\omega$ is defined as

$$\text{ShLin}^\omega = \{[S]_U \mid U \in \mathscr{P}_f(\mathcal{V}), S \subseteq \mathscr{P}_m(U), S \neq \emptyset \Rightarrow \{\!\{\}\!\} \in S\} \ , \tag{5}$$

and ordered by $[S_1]_{U_1} \leq_\omega [S_2]_{U_2}$ iff $U_1 = U_2$ and $S_1 \subseteq S_2$.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

The existence of the empty multiset, when S is not empty, is required in order to have a surjective abstraction function.

In order to ease the notation, we write $[\{\{\}, B_1, \ldots, B_n\}]_U$ as $[B_1, \ldots, B_n]_U$ by omitting the braces and the empty multiset, and variables in each ω-sharing group are sorted lexicographically. Moreover, if $X \in \mathtt{ShLin}^\omega$, we write $B \in X$ in place of $X = [S]_U \wedge B \in S$. Analogously, if $S' \in \mathtt{ShLin}^\omega$, we write $S' \subseteq X$ in place of $X = [S]_U \wedge S' \subseteq S$. The best correct abstraction of a substitution $[\theta]_U$ is

$$\alpha_\omega([\theta]_U) = [\{\theta^{-1}(v)_{|U} \mid v \in \mathcal{V}\}]_U \quad . \tag{6}$$

Example 3.2.
Given $\theta = \{x/s(y,u,y), z/s(u,u), v/u\}$ and $U = \{w,x,y,z\}$, we have $\theta^{-1}(u) = uvxz^2$, $\theta^{-1}(y) = x^2y$, $\theta^{-1}(z) = \theta^{-1}(v) = \theta^{-1}(x) = \{\{\}\}$, and $\theta^{-1}(o) = o$ for all the other variables (w included). Projecting over U, we obtain $\alpha_\omega([\theta]_U) = [x^2y, xz^2, w]_U$.

3.2 Matching operator

In the matching operation, when $[\theta_1]_{U_1}$ is matched with $[\theta_2]_{U_2}$, sharing groups in θ_1 and θ_2 are joined together in the resulting substitution. However, not all combinations are allowed. Assume $\alpha_\omega([\theta_i]_{U_i}) = [S_i]_{U_i}$ for $i \in \{1,2\}$. If $\mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2})$ is defined, θ_1 will not be further instantiated and thus $\alpha_\omega(\mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2})_{|U_1}) \subseteq S_1$. Moreover, the sharing groups in S_2 which do not contain any variable in U_1 are not affected by the unification, since the corresponding existential variable does not appear in $\theta_2(v)$ for any $v \in U_1$.

Example 3.3.
Let $\theta_1 = \{x/r(w_1,w_2,w_2,w_3,w_3), y/a, z/r(w_1)\}$ with $U_1 = \{x,y,z\}$ and $\theta_2 = \{x/r(w_4,w_5,w_6,w_8,w_8), u/r(w_4,w_7), v/r(w_7,w_8)\}$ with $U_2 = \{u,v,x\}$. We have that

$$[\theta]_U = \mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2}) = [u/r(w_1,w_7), v/r(w_7,w_3), x/r(w_1,w_2,w_2,w_3,w_3), y/a, z/r(w_1)]_U \quad ,$$

with $U = \{u,v,x,y,z\}$. At the abstract level, we have $[S_1]_{U_1} = \alpha_\omega([\theta_1]_{U_1}) = [x^2, xz]_{U_1}$, $[S_2]_{U_2} = \alpha_\omega([\theta_2]_{U_2}) = [uv, ux, vx^2, x]_{U_2}$, and $[S]_U = \alpha_\omega([\theta]_U) = [uv, uxz, vx^2, x^2]_U$.

Note that the new sharing group uxz has the property that its restriction to U_1 is in S_1. More generally, if we abstract θ_1 w.r.t. the variables in U_1, we get $\alpha_\omega([\theta]_{U_1}) = [x^2, xz]_{U_1}$. This is equal to $[S_1]_{U_1}$, showing that no new sharing group has been introduced by the matching operation relative to the variables in U_1. Moreover, the sharing group uv, which does not contain variables in U_1, is brought unchanged from S_2 to S.

Following the idea presented above, we may design an abstract matching operation for the domain $\mathtt{ShLin}^\omega$.

Definition 3.4 (Matching over $\mathtt{ShLin}^\omega$).
Given $[S_1]_{U_1}, [S_2]_{U_2} \in \mathtt{ShLin}^\omega$, we define

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

$$\mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2}) = [S_2' \cup \{X \in \mathscr{P}_m(U_1 \cup U_2) \mid X_{|U_1} \in S_1 \wedge X_{|U_2} \in (S_2'')^*\}]_{U_1 \cup U_2}$$

where

$$S_2' = \{B \in S_2 \mid B_{|U_1} = \emptyset\} \qquad S_2'' = S_2 \setminus S_2' \qquad S^* = \{\uplus \mathcal{S} \mid \mathcal{S} \in \mathscr{P}_m(S)\} \ .$$

We now show an example of computing the matching over $\mathtt{ShLin}^\omega$.

Example 3.5.
Consider $[S_1]_{U_1} = [x^2, xz]_{xyz}$ and $[S_2]_{U_2} = [uv, ux, vx^2, x]_{uvx}$ as in Example 3.3. We show that from the definition of match_ω, it holds

$$\mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2}) = [uv, uxz, xz, u^2x^2, ux^2, vx^2, x^2]_{uvxyz} \ .$$

First, we have that $S_2' = \{uv\}$ and $S_2'' = \{ux, vx^2, x\}$. Apart from uv, which directly comes from S_2', all the other ω-sharing groups in the result may be obtained by choosing a multiset $\mathcal{M}$ of sharing groups in S_2'' and summing them together, obtaining $\uplus\mathcal{M}$. Then, we consider if it is possible to add to $\uplus\mathcal{M}$ some occurrences of the variables y and z, for instance, n occurrences of y and m of z, in such a way that $(\uplus\mathcal{M}) \uplus \{\!\{y^n z^m\}\!\}$ restricted to U_1 is a sharing group in S_1.

We start by considering $\mathcal{M}$ with a single ω-sharing group.

- If $\mathcal{M} = \{\!\{ux\}\!\}$, then $\uplus\mathcal{M} = ux$ but $(\uplus\mathcal{X})|_{U_1} = x \notin S_1$. However, we can add the variable z to get $uxz \in S_1$; hence uxz is an ω-sharing group in the result of match_ω.
- If $\mathcal{M} = \{\!\{vx^2\}\!\}$, then $\uplus\mathcal{M} = vx^2$ and $(\uplus\mathcal{M})|_{U_1} = x^2$, which is already an element of S_1. Therefore, vx^2 is in the result of match_ω.
- If $\mathcal{M} = \{\!\{x\}\!\}$, then $\uplus\mathcal{M} = x$, but $(\uplus\mathcal{M})|_{U_1} = x \notin S_1$. However, we can add the variable z to get $xz \in S_1$; hence xz is in the result of match_ω.

We now consider the cases when $\mathcal{M}$ has two (possibly equal) elements. Note that if $\mathcal{M}$ contains vx^2, it cannot contain anything else; otherwise, $\uplus\mathcal{M}$ would contain at least three occurrences of x, and no sharing group in S_1 could be matched. Therefore, the only choices are

- if $\mathcal{M} = \{\!\{ux, ux\}\!\}$, then $\uplus\mathcal{M} = u^2x^2$ and $(\uplus\mathcal{M})|_{U_1} = x^2$, which is already in S_1; hence u^2x^2 is in the result of match_ω;
- if $\mathcal{M} = \{\!\{x, x\}\!\}$, then $\uplus\mathcal{M} = x^2$ and $(\uplus\mathcal{M})|_{U_1} = x^2$, which is already in S_1; hence x^2 is in the result of match_ω;
- if $\mathcal{M} = \{\!\{ux, x\}\!\}$, then $\uplus\mathcal{M} = ux^2$ and $(\uplus\mathcal{M})|_{U_1} = x^2$, which is already in S_1; hence ux^2 is in the result of match_ω.

In theory, we should also consider the case when $\mathcal{M}$ has more than two elements, but in the example, this would not lead to new results, for the same reason why vx^2 may only be used alone.

We will prove that match_ω is correct and therefore comes as no surprise that

$$\alpha_\omega(\mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2}))) = [uv, uxz, vx^2, x^2]_U \leq [uv, uxz, xz, u^2x^2, ux^2, vx^2, x^2]_U \ .$$

The lack of equality means that match_ω is not (α-)complete (Giacobazzi et al., 2000; Amato and Scozzari, 2011). We will show later that it is optimal.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

We try to give the intuition behind the definition of match, especially in the context of the backward unification. First note that the operator is additive on the first argument, namely:

$$\mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2}) = \left[\bigcup_{B\in S_1} \mathsf{match}_\omega([\{B\}]_{U_1}, [S_2]_{U_2})\right]_{U_1\cup U_2}$$

This immediately implies that we can reason on matching considering one sharing group at a time. Given a sharing group $B \in S_1$, which represents the exit substitution from a sub-computation, we try to guess which of the sharing groups in S_2 are part of the entry substitution that has generated the sub-computation ending with B. In the simple case, when $U_1 \supseteq U_2$, we simply check that B can be generated by the sharing groups in S_2; that is, there exists $\mathcal{S} \in \mathscr{P}_m(S_2)$ such that $B_{|U_2} = \uplus\mathcal{S}$.

The difficult case is when U_2 contains some variables that are not in U_1. These variables come from the call substitution; they are removed from the abstraction before entering the sub-computation and now should be re-introduced as precisely as possible. In this case, we build a new sharing group X such that $X_{|U_1}$ coincides with B and $X_{|U_2}$ is generated by S_2; namely, there exists $\mathcal{S} \in \mathscr{P}_m(S_2)$ such that $X_{|U_2} = \uplus\mathcal{S}$. This condition ensures that we pair each exit substitution θ_1 (in the concretization of B) with some entry substitution θ_2 (in the concretization of $\mathcal{S}$), which is less instantiated than θ_1.

Note that, although the abstract unification operator mgu_ω defined in Amato and Scozzari (2010) takes an abstract object and a substitution as inputs, the operator match_ω is designed in such a way that both the arguments are abstract objects. The reason for this choice is that these are the variants needed for static analysis of logic programs. However, it would be possible to devise a variant of mgu_ω with two abstract arguments and variants of match_ω with one abstract argument and a concrete one.

In order to prove the correctness of match_ω, we first extend the definition of θ^{-1} to the case when it is applied to a sharing group B, elementwise as

$$\theta^{-1}(v_1^{i_1}\cdots v_n^{i_n}) = \biguplus\{\!\{(\theta^{-1}(v_1))^{i_1}, \ldots, (\theta^{-1}(v_n))^{i_n}\}\!\} \ . \tag{7}$$

It turns out that

$$(\eta\circ\theta)^{-1}(B) = \theta^{-1}(\eta^{-1}(B)) \ , \tag{8}$$

a result that has been proved in Amato and Scozzari (2010).

Example 3.6.
Given $\eta = \{x/s(y,u,y), z/s(u,u), v/u\}$ and $\theta = \{v/a, w/s(x,x)\}$, we have $\eta\circ\theta = \{v/a, w/s(s(y,u,y), s(y,u,y)), x/s(y,u,y), z/s(u,u))\}$ and $(\eta\circ\theta)^{-1}(u) = uw^2xz^2$. The same result may be obtained as $\theta^{-1}(\eta^{-1}(u))$ since $\eta^{-1}(u) = uvxz^2$ and

$$\begin{aligned}\theta^{-1}(uvxz^2) &= \biguplus\{\!\{\theta^{-1}(u), \theta^{-1}(v), \theta^{-1}(x), (\theta^{-1}(z))^2\}\!\} \\ &= \biguplus\{\!\{u, \{\!\}, xw^2, (z)^2\}\!\} = uw^2xz^2 \ .\end{aligned}$$

Theorem 3.7 (Correctness of match_ω*).*
match_ω *is correct w.r.t.* match.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

Proof
Given $[S_1]_{U_1} \rhd [\theta_1]_{U_1}$ and $[S_2]_{U_2} \rhd [\theta_2]_{U_2}$ such that $\mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2})$ is defined, we need to prove that

$$\mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2}) \rhd \mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2}) \ .$$

Assume without loss of generality that $\mathrm{dom}(\theta_1) = U_1$, $\mathrm{dom}(\theta_2) = U_2$, and $\mathrm{vars}(\theta_1) \cap \mathrm{vars}(\theta_2) \subseteq U_1 \cap U_2$. In particular, this implies $\mathrm{rng}(\theta_1) \cap \mathrm{rng}(\theta_2) = \emptyset$. By hypothesis $\theta_1 \preceq_{U_1 \cap U_2} \theta_2$, that is, there exists $\delta \in \mathit{ISubst}$ such that $\theta_1(x) = \delta(\theta_2(x))$ for each $x \in U_1 \cap U_2$, $\mathrm{dom}(\delta) = \mathrm{vars}(\theta_2(U_1 \cap U_2))$ and $\mathrm{rng}(\delta) = \mathrm{vars}(\theta_1(U_1 \cap U_2))$. We have

$$\begin{aligned}
&\mathsf{mgu}(\theta_1, \theta_2) \\
&= \mathsf{mgu}(\{\theta_2(x) = \theta_2(\theta_1(x)) \mid x \in U_1\}) \circ \theta_2 \\
&\qquad \text{[by assumptions on the } \theta_i\text{'s]} \\
&= \mathsf{mgu}(\{\theta_2(x) = \theta_1(x) \mid x \in U_1\}) \circ \theta_2 \\
&= \mathsf{mgu}(\{\theta_2(x) = \theta_1(x) \mid x \in U_1 \cap U_2\}) \circ (\theta_1)_{|U_1 \setminus U_2} \circ \theta_2 \\
&\qquad \text{[since } \mathrm{vars}(\theta_2) \cap \mathrm{vars}((\theta_1)_{U_1 \setminus U_2}) = \emptyset\text{]} \\
&= \mathsf{mgu}(\{\theta_2(x) = \delta(\theta_2(x)) \mid x \in U_1 \cap U_2\}) \circ ((\theta_1)_{|U_1 \setminus U_2} \uplus \theta_2) \\
&= \delta \circ ((\theta_1)_{|U_1 \setminus U_2} \uplus \theta_2) \\
&\qquad \text{[since } \mathrm{dom}(\delta) \cap \mathrm{vars}((\theta_1)_{U_1 \setminus U_2}) = \emptyset\text{]} \\
&= (\theta_1)_{|U_1 \setminus U_2} \uplus (\delta \circ \theta_2) \ .
\end{aligned}$$

With an analogous derivation, we obtain

$$\mathsf{mgu}(\theta_1, \theta_2) = \theta_1 \uplus (\delta \circ (\theta_2)_{|U_2 \setminus U_1}) \ .$$

Now, if $\eta = \mathsf{mgu}(\theta_1, \theta_2)$, we need to prove that for each $v \in \mathcal{V}$, $\eta^{-1}(v)_{|U_1 \cup U_2} \in \mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})$. We distinguish several cases.

- $v \notin \mathrm{rng}(\theta_1)$ and $v \notin \mathrm{rng}(\theta_2)$. In this case $v \notin \mathrm{vars}(\delta)$ and $\eta^{-1}(v)_{|U_1 \cup U_2} = \{\!\{\}\!\} \in \mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})$.
- $v \in \mathrm{rng}(\theta_1)$ and $v \notin \mathrm{vars}(\theta_1(U_2))$. In this case $v \notin \mathrm{rng}(\theta_2)$ and $v \notin \mathrm{vars}(\delta)$. We have $\eta^{-1}(v)_{|U_1 \cup U_2} = \theta_1^{-1}(v)_{|U_1} \in S_1 \subseteq \mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})$.
- $v \in \mathrm{rng}(\theta_2)$ and $v \notin \mathrm{vars}(\theta_2(U_1))$. In this case $v \notin \mathrm{rng}(\theta_1)$ and $v \notin \mathrm{vars}(\delta)$. We have $\eta^{-1}(v)_{|U_1 \cup U_2} = \theta_2^{-1}(v)_{|U_2} \in S_2$. Since $v \notin \mathrm{vars}(\theta_2(U_1))$, then $\theta_2^{-1}(v)_{|U_2} \in S_2' \subseteq \mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})$.
- $v \in \mathrm{vars}(\theta_2(U_1 \cap U_2))$. In this case $v \notin \mathrm{rng}(\theta_1)$ and $v \in \mathrm{dom}(\delta)$; therefore, $\eta^{-1}(v)_{|U_1 \cup U_2} = \{\!\{\}\!\}$.
- $v \in \mathrm{vars}(\theta_1(U_1 \cap U_2))$. Now, $v \in \mathrm{rng}(\delta)$ and $\eta^{-1}(v)_{|U_2} = \theta_2^{-1}(\delta^{-1}(v))_{|U_2} = \biguplus X$, where $X \in \mathscr{P}_m(S_2)$ and each sharing group $B \in X$ is of the form $\theta_2^{-1}(w)_{|U_2}$ for some $w \in \lfloor\!\lfloor \delta^{-1}(v) \rfloor\!\rfloor$. Note that every such w is an element of $\mathrm{vars}(\theta_2(U_1 \cap U_2))$; therefore, $\theta_2^{-1}(w)_{|U_2} \in S_2''$, $X \in \mathscr{P}_m(S_2'')$, and $\biguplus X \in (S_2'')^*$. On the other side, since $\eta = \theta_1 \uplus (\delta \circ (\theta_2)_{|U_2 \setminus U_1})$, we have $\eta^{-1}(v)_{|U_1} = \theta_1^{-1}(v)_{|U_1} \in S_1$. Hence, $\eta^{-1}(v)_{|U_1 \cup U_2} \in \mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})$.

This concludes the proof. □

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

Now, we may prove the optimality of match_ω. Actually, we prove a stronger property, a sort of weak completeness of match_ω, which will be used later to derive optimality.

Example 3.8.
Consider again the substitutions and sharing groups in Examples 3.3 and 3.5. Recall that $U_1 = \{x, y, z\}$ and $U_2 = \{u, v, x\}$. We have seen that, although the sharing groups xz, u^2x^2, and ux^2 are in $\mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})$, they are not in $\alpha_\omega(\mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2}))$. If match_ω were optimal, we should be able to find, for each $B \in \{xz, u^2x^2, ux^2\}$, a pair of substitutions θ_3 and θ_4 such that $[S_1]_{U_1} \triangleright [\theta_3]_{U_1}$, $[S_2]_{U_2} \triangleright [\theta_4]_{U_2}$, and $B \in \alpha_\omega(\mathsf{match}([\theta_3]_{U_1}, [\theta_4]_{U_2})$. Actually, we can do better, keep $\theta_4 = \theta_2$ fixed, and only change θ_3 for different sharing groups.

Consider $B = xz$. Note that $B_{|U_1} = xz$ and $B_{|U_2} = x = \uplus\, \mathcal{X}$, with $\mathcal{X} = \{\!\{x\}\!\}$. The sharing group x in θ_2 is generated by the variables w_5 and w_6. Consider the substitution $\theta_3 = \{x/r(a, w, a, a, a), y/a, z/w\}$ where

- the binding $x/r(a, w, a, a, a)$ is obtained by the corresponding binding in θ_2 replacing w_5 with a fresh variable w and all the other variables in the range with the constant symbol a;
- the bindings $\{y/a, z/w\}$ are chosen according to $B_{|U_1 \setminus U_2} = z$.

It is immediate to show that $xz \in \alpha_\omega(\mathsf{mgu}([\theta_3]_{U_1}, [\theta_2]_{U_2}))$ and $[S_1]_{U_1} \triangleright [\theta_3]_{U_1}$.

As another example, let us consider $B = u^2x^2$. In this case, $B_{|U_1} = x^2 \in S_1$ and $B_{|U_2} = u^2x^2 = \uplus\{\!\{ux, ux\}\!\}$. The variable that generates the sharing group ux is w_4. We proceed as before and obtain $\theta_3 = \{x/r(r(w, w), a, a, a, a), y/a, z/a\}$. Note that w_4 has been replaced with $r(w, w)$ since two copies of ux are needed to obtain u^2x^2. Again $[S_1]_{U_1} \triangleright [\theta_3]_{U_1}$ and $u^2x^2 \in \alpha_\omega(\mathsf{mgu}([\theta_3]_{U_1}, [\theta_2]_{U_2}))$.

We distill the idea presented above in the following result.

Lemma 3.9 (Completeness on the second argument).
Given $[S_1]_{U_1} \in \mathtt{ShLin}^\omega$ and $[\theta_2]_{U_2} \in \mathit{ISubst}_\sim$, there exist $\delta_1, \ldots, \delta_n \in \mathit{ISubst}_\sim$ such that for all $i \in \{1, \ldots, n\}$, $[S_1]_{U_1} \triangleright [\delta_i]_{U_1}$ and

$$\mathsf{match}_\omega([S_1]_{U_1}, \alpha_\omega([\theta_2]_{U_2})) = \left[\bigcup_{i \in \{1,\ldots,n\}} \alpha_\omega(\mathsf{match}([\delta_i]_{U_1}, [\theta_2]_{U_2}))\right]_{U_1 \cup U_2}$$

Proof
Since we already know that match_ω is a correct abstraction of match, we only need to prove that, given $[S_1]_{U_1} \in \mathtt{ShLin}^\omega$ and $[\theta_2]_{U_2} \in \mathit{ISubst}_\sim$, for any sharing group $B \in \mathsf{match}_\omega([S_1]_{U_1}, \alpha_\omega([\theta_2]_{U_2}))$, there exists $[\theta_1]_{U_1} \in \mathit{ISubst}_\sim$ such that $[S_1]_{U_1} \triangleright [\theta_1]_{U_1}$ and $B \in \alpha_\omega(\mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2}))$.

In order to ease notation, let $U = U_1 \cup U_2$, $[S_2]_{U_2} = \alpha_\omega([\theta_2]_{U_2})$, and $[S]_U = \mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})$. We may choose θ_2 such that $\mathrm{dom}(\theta_2) = U_2$ without loss of generality. Moreover, S_2' and S_2'' are given as in the definition of abstract matching. We distinguish two cases.

first case) If $B \in S_2'$, there exists $v \in \mathcal{V}$ such that $\theta_2^{-1}(v)_{|U_2} = B$. Let $X = \mathrm{vars}(\theta_2(U_1 \cap U_2))$ and take $\delta = \{x/a \mid x \in X\}$. Then $\theta_1 = (\delta \circ \theta_2)_{|U_1} \uplus \{x/a \mid x \in$

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

$U_1 \setminus U_2\}$ is such that $\theta_1^{-1}(v)_{|U_1} = \{\!\{\}\!\}$ for each $v \in \mathcal{V}$, therefore $[S_1]_{U_1} \triangleright [\theta_1]_{U_1}$. Moreover, since $\theta_1 \preceq_{U_1 \cap U_2} \theta_2$, we have that $\mathsf{match}_\omega([\theta_1]_{U_1}, [\theta_2]_{U_2})$ is defined equal to $\mathsf{mgu}([\theta_1]_{U_1}, [\theta_2]_{U_2})$ and $\mathsf{mgu}([\theta_1]_{U_1}, [\theta_2]_{U_2}) = [\mathsf{mgu}(\theta_1, \theta_2)]_U$ since $\mathrm{vars}(\theta_1) \cap \mathrm{vars}(\theta_2) \subseteq U_1 \cap U_2$. By the proof of Theorem 3.7, we have that $\mathsf{mgu}(\theta_1, \theta_2) = (\theta_1)_{|U_1 \setminus U_2} \uplus (\delta \circ \theta_2)$. Since $B_{|U_1} = \{\!\{\}\!\}$, then $v \notin X = \mathrm{vars}(\delta)$, and therefore, $\mathsf{mgu}(\theta_1, \theta_2)^{-1}(v)_{|U} = \theta_2^{-1}(v)_{|U} = B$. Hence, B is an ω-sharing group in $\alpha_\omega(\mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2}))$, which is what we wanted to prove.

second case) We now assume $B_{|U_1} \in S_1$ and $B_{|U_2} = \biguplus X$ with $\mathcal{X} \in \mathscr{P}_m(S_2'')$. Then, for each $H \in \lfloor\!\lfloor \mathcal{X} \rfloor\!\rfloor$, there exists $v_H \in \mathcal{V}$ such that $\theta_2^{-1}(v_H)_{|U_2} = H$. Since $H \cap U_1 \neq \{\!\{\}\!\}$ for each $H \in \lfloor\!\lfloor \mathcal{X} \rfloor\!\rfloor$, then $v_H \in Y = \mathrm{vars}(\theta_2(U_1 \cap U_2))$. Consider the substitutions

$$\delta = \{v_H/t(\underbrace{v, \ldots, v}_{\mathcal{X}(H) \text{ times}}) \mid H \in \lfloor\!\lfloor \mathcal{X} \rfloor\!\rfloor\} \uplus \{y/a \mid y \in Y \text{ and } \forall H \in \lfloor\!\lfloor \mathcal{X} \rfloor\!\rfloor . y \neq v_H\} \ ,$$

$$\eta = \{w/t(\underbrace{v, \ldots, v}_{B(w) \text{ times}}) \mid w \in U_1 \setminus U_2\} \ ,$$

for a fresh variable v. Let us define $\theta_1 = \eta \uplus (\delta \circ \theta_2)_{|U_1}$. We want to prove that $[S_1]_{U_1} \triangleright [\theta_1]_{U_1}$. Note that $\mathrm{vars}(\theta_1(U_1)) = \{v\}$, hence we only need to check that $\theta_1^{-1}(v)_{|U_1} \in S_1$. We have that $\theta^{-1}(v)_{|U_1} = \eta^{-1}(v)_{|U_1} \uplus (\theta_2^{-1}\delta^{-1}(v))_{|U_1} = \eta^{-1}(v)_{|U_1} \uplus \theta_2^{-1}(\{\!\{v_H^{X(H)} \mid H \in \lfloor\!\lfloor \mathcal{X} \rfloor\!\rfloor\}\!\})_{|U_1} = B_{|U_1 \setminus U_2} \uplus (\biguplus \mathcal{X})_{|U_1 \cap U_2} = B_{|U_1} \uplus B_{|U_1 \cap U_2} = B_{|U_1}$ which, we know, is an element of S_1. Moreover, since $\theta_1 \preceq_{U_1 \cap U_2} \theta_2$ and $\mathrm{vars}(\theta_1) \cap \mathrm{vars}(\theta_2) \subseteq U_1 \cap U_2$, we have that $\mathsf{match}_\omega([\theta_1]_{U_1}, [\theta_2]_{U_2}) = [\mathsf{mgu}(\theta_1, \theta_2)]_U$. If we define $\theta = \mathsf{mgu}(\theta_1, \theta_2)$, by looking at the proof of Theorem 3.7, we have that $\theta = \theta_1 \uplus (\delta \circ (\theta_2)_{|U_2 \setminus U_1})$ and $\theta = (\theta_1)_{|U_1 \setminus U_2} \uplus (\delta \circ \theta_2)$. By the first equality, it is immediate to check that $\theta^{-1}(v)_{|U_1} = \theta_1^{-1}(v)_{|U_1} = B_{|U_1}$. By the second equality, $\theta^{-1}(v)_{|U_2} = \theta_2^{-1}(\{\!\{v_H^{\mathcal{X}(H)} \mid H \in \lfloor\!\lfloor \mathcal{X} \rfloor\!\rfloor\}\!\})_{|U_2} = \biguplus \mathcal{X} = B_{|U_2}$. Therefore, $\theta^{-1}(v)_{|U} = B$.

It is worth noting that, although this proof uses a function symbol of arbitrary arity, it may be easily rewritten using only one constant symbol and one function symbol of arity at least two, as required at the beginning of Section 2.1. □

Theorem 3.10 (Optimality of match_ω*).*
The operation match_ω is optimal w.r.t. match.

Proof
It is enough to prove that for each $[S_1]_{U_1}, [S_2]_{U_2} \in \mathtt{ShLin}^\omega$ and $B \in \mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})$, there are substitutions θ_1, θ_2 such that $[S_1]_{U_1} \triangleright [\theta_1]_{U_1}$, $[S_2]_{U_2 1} \triangleright [\theta_2]_{U_2}$ and $B \in \alpha_\omega(\mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2}))$. Assume $B_{|U_2} = \biguplus \mathcal{X}$ where $\mathcal{X} \in \mathscr{P}_m(S_2'')$. Consider a substitution $[\theta_2]_{U_2}$ such that $[\lfloor\!\lfloor \mathcal{X} \rfloor\!\rfloor]_{U_2} \leq \alpha_\omega([\theta_2]_{U_2}) \leq [S_2]_{U_2}$. It means that, for each $H \in \lfloor\!\lfloor \mathcal{X} \rfloor\!\rfloor$, there is $v_H \in \mathcal{V}$ such that $\theta_2^{-1}(v_H)_{|U_2} = H$. If $\lfloor\!\lfloor \mathcal{X} \rfloor\!\rfloor = \{H_1, \ldots, H_n\}$, we may define θ_2 as the substitution with $\mathrm{dom}(\theta_2) = U_2$ and, for each $u \in U_2$,

$$\theta_2(u) = t(\underbrace{v_{H_1}, \ldots, v_{H_1}}_{H_1(u) \text{ times}}, \underbrace{v_{H_2}, \ldots, v_{H_2}}_{H_2(u) \text{ times}}, \ldots, \underbrace{v_{H_n}, \ldots, v_{H_n}}_{H_n(u) \text{ times}}) \ .$$

By Lemma 3.9, there exists $[\theta_1]_{U_1}$, such that $[S_1]_{U_1} \triangleright [\theta_1]_{U_1}$ and $B \in \alpha_\omega(\mathsf{match}([\theta_1]_{U_1}, [\theta_2]_{U_2}))$. Therefore, match_ω is the optimal approximation of match. □

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

4 Abstract matching over $\mathtt{ShLin}^2$

The domain $\mathtt{ShLin}^\omega$ has been inspired by the domain $\mathtt{ShLin}^2$, which appeared for the first time in King (1994). The novelty of $\mathtt{ShLin}^2$ was to embed linearity information inside the sharing groups, instead of keeping them separate like it was in $\mathtt{Sharing} \times \mathtt{Lin}$.

4.1 The domain $\mathtt{ShLin}^2$

Here we recall the main definitions for the domain $\mathtt{ShLin}^2$, viewed as an abstraction of $\mathtt{ShLin}^\omega$, following the presentation given in Amato and Scozzari (2010).

The idea is to simplify the domain $\mathtt{ShLin}^\omega$ by only recording whether a variable in a sharing group is linear or not, but forgetting its actual multiplicity. Intuitively, we abstract an ω-sharing group by replacing any exponent equal to or greater than 2 with a new symbol ∞.

A *2-sharing group* is a map $o : \mathcal{V} \to \{0, 1, \infty\}$ such that its support $\lfloor\!\lfloor o \rfloor\!\rfloor = \{v \in \mathcal{V} \mid o(v) \neq 0\}$ is finite. We use a polynomial notation for 2-sharing groups as for ω-sharing groups. For instance, $o = xy^\infty z$ denotes the 2-sharing group whose support is $\lfloor\!\lfloor o \rfloor\!\rfloor = \{x, y, z\}$, such that $o(x) = o(z) = 1$ and $o(y) = \infty$. We denote with $\emptyset$ the 2-sharing group with empty support and by $Sg^2(V)$ the set of 2-sharing groups whose support is a subset of V. Note that in King (1994) the number 2 is used as an exponent instead of ∞, but we prefer our notation to be coherent with ω-sharing groups.

An ω-sharing group B may be abstracted into the 2-sharing group $\alpha_2(B)$ given by

$$\alpha_2(B) = \lambda v \in \mathcal{V}. \begin{cases} B(v) & \text{if } B(v) \leq 1, \\ \infty & \text{otherwise.} \end{cases} \tag{9}$$

For instance, the ω-sharing groups $xy^2z, xy^3z, xy^4z, xy^5z, \ldots$ are all abstracted into $xy^\infty z$.

There are operations on 2-sharing groups that correspond to variable projection and multiset union. For projection

$$o_{|V} = \lambda v \in \mathcal{V}. \begin{cases} o(V) & \text{if } v \in V, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

while for multiset union

$$o \uplus o' = \lambda v \in \mathcal{V}. o(v) \oplus o'(v) \ , \tag{11}$$

where $0 \oplus x = x \oplus 0 = x$ and $\infty \oplus x = x \oplus \infty = 1 \oplus 1 = \infty$. We will use $\biguplus \{\!\{o_1, \ldots, o_n\}\!\}$ for $o_1 \uplus \cdots \uplus o_n$. Given a sharing group o, we also define the *delinearization* operator:

$$o^2 = o \uplus o \ . \tag{12}$$

This operator is extended pointwise to sets and multisets. The next proposition shows some properties of these operators.

Proposition 4.1
Given an ω-sharing group B, a set of variables V, and multiset of ω-sharing groups $\mathcal{X}$, the following properties hold:

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

1. $\lfloor\!\lfloor B \rfloor\!\rfloor = \lfloor\!\lfloor \alpha_2(B) \rfloor\!\rfloor$
2. $\alpha_2(B_{|V}) = \alpha_2(B)_{|V}$
3. $\alpha_2(\biguplus \mathcal{X}) = \biguplus \alpha_2(\mathcal{X})$
4. $\alpha_2(B \uplus B) = \alpha_2(B)^2$

where $\alpha_2(\mathcal{X})$ is just the elementwise extension of α_2 to a multiset of ω-sharing groups.

Proof
Given the ω-sharing group B, we have $\lfloor\!\lfloor \alpha_2(B) \rfloor\!\rfloor = \{x \in \mathcal{V} \mid \alpha_2(B) \neq 0\} = \{x \in \mathcal{V} \mid B(x) \neq 0\} = \lfloor\!\lfloor B \rfloor\!\rfloor$, which proves Property 1. For Property 2, given $V \subseteq \mathcal{V}$, we want to prove that $\alpha_2(B_{|V})(v) = (\alpha_2(B)_{|V})(v)$ for each $v \in \mathcal{V}$. The property is easily proved considering the three cases $v \notin V$, $v \in V \setminus \lfloor\!\lfloor B \rfloor\!\rfloor$, and $v \in V \cap \lfloor\!\lfloor B \rfloor\!\rfloor$. Property 3 has been proved in Amato and Scozzari (2010). Property 4 is a trivial consequence of Property 3. □

Since we do not want to represent definite nonlinearity, we introduce an order relation over sharing groups as follows:

$$o \leq o' \iff \lfloor\!\lfloor o \rfloor\!\rfloor = \lfloor\!\lfloor o' \rfloor\!\rfloor \wedge \forall x \in \lfloor\!\lfloor o \rfloor\!\rfloor .\ o(x) \leq o'(x) \ ,$$

and we restrict our attention to downward closed sets of sharing groups. The domain we are interested in is the following:

$$\texttt{ShLin}^2 = \{[T]_U \mid T \in \mathscr{P}_{\downarrow}(Sg^2(U)), U \in \mathscr{P}_f(\mathcal{V}), T \neq \emptyset \Rightarrow \emptyset \in T\} \ ,$$

where $\mathscr{P}_{\downarrow}(Sg^2(U))$ is the powerset of downward closed subsets of $Sg^2(U)$ according to $\leq$ and $[T_1]_{U_1} \leq_2 [T_2]_{U_2}$ iff $U_1 = U_2$ and $T_1 \subseteq T_2$. For instance, the set $\{xy^\infty z\}$ is not downward closed, while $\{xyz, xy^\infty z\}$ is downward closed. There is a Galois insertion of $\texttt{ShLin}^2$ into $\texttt{ShLin}^\omega$ given by the pair of adjoint maps $\gamma_2 : \texttt{ShLin}^2 \to \texttt{ShLin}^\omega$ and $\alpha_2 : \texttt{ShLin}^\omega \to \texttt{ShLin}^2$:

$$\gamma_2([T]_U) = \left[\alpha_2^{-1}(T)\right]_U \qquad \alpha_2([S]_U) = [\downarrow\alpha_2(S)]_U \ ,$$

where $\downarrow T = \{o \mid o' \in T, o \leq o'\}$ is the downward closure of T. With an abuse of notation, we also apply γ_2 and α_2 to subsets of ω-sharing groups and 2-sharing groups, respectively, by ignoring the set of variables of interest. For instance, $\gamma_2([\emptyset, xyz, xy^\infty z]_{x,y,z}) = [\{\!\{\}\!\}, xyz, xy^2z, xy^3z, xy^4z, xy^5z, \ldots]_{x,y,z}$. Moreover, we write $\downarrow[S]_U$ as an alternative form for $[\downarrow S]_U$.

Example 4.2.
Consider the substitution $[\theta]_U = [\{x/s(y,u,y), z/s(u,u), v/u\}]_{w,x,y,z}$ in Example 3.2. Its abstraction in $\texttt{ShLin}^2$ is given by

$$\alpha_2(\alpha_\omega([\theta]_U)) = [\{xy, x^\infty y, xz, xz^\infty, w\}]_U = [\downarrow\{x^\infty y, xz^\infty, w\}]_U \ .$$

Analogously, substitutions $[\theta_1]_{U_1} = [\{x/r(w_1,w_2,w_2,w_3,w_3), y/a, z/r(w_1)\}]_{x,y,z}$ and $[\theta_2]_{U_2} = [\{x/r(w_4,w_5,w_6,w_8,w_8), u/r(w_4,w_7), v/r(w_7,w_8)\}]_{u,v,x}$ from Example 3.3 are abstracted into $[T_1]_{U_1} = \downarrow[x^\infty, xz]_{U_1}$ and $[T_2]_{U_2} = \downarrow[uv, ux, vx^\infty, x]_{U_2}$, respectively.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

4.2 Matching operator

By composition, $\alpha_2 \circ \alpha_\omega$ is an abstraction from $ISubst_\sim$ to $\mathtt{ShLin}^2$. Properties of the Galois connection $\langle \alpha_2, \gamma_2 \rangle$ may be lifted to properties of $\alpha_2 \circ \alpha_\omega$. In our case, we need to define an abstract matching match_2 over $\mathtt{ShLin}^2$, which is an optimal w.r.t. match. However, optimality of match_2 w.r.t. match is immediately derived by optimality w.r.t. match_ω. Since the correspondence between $\mathtt{ShLin}^\omega$ and $\mathtt{ShLin}^2$ is straightforward, the same happens for match_ω and match_2.

Definition 4.3 (Matching over $\mathtt{ShLin}^2$*).*
Given $[T_1]_{U_1}, [T_2]_{U_2} \in \mathtt{ShLin}^2$, we define

$$\mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2}) = [T_2' \cup \downarrow\{o \in Sg^2(\mathcal{V}) \mid o_{|U_1} \in T_1 \wedge o_{|U_2} \in (T_2'')^*\}]_{U_1 \cup U_2}$$

where $T_2' = \{B \in T_2 \mid B_{|U_1} = \emptyset\}$, $T_2'' = T_2 \setminus T_2'$ and $T^* = \{\biguplus X \mid X \subseteq T \cup T^2\}$.

Example 4.4.
Under the hypothesis of Examples 3.5 and 4.2, and according to the definition of match_2, we have that $T_2' = \{uv\}$, $T_2'' = \{ux, vx, vx^\infty, x\}$ and

$$\mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2}) = \downarrow[uv, u^\infty v^\infty x^\infty, uxz, u^\infty x^\infty, v^\infty x^\infty, vxz, x^\infty, xz]_{u,v,x,y,z} \ .$$

Note that

$$\alpha_2(\mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})) = \downarrow[uv, u^\infty x^\infty, uxz, vx^\infty, x^\infty, xz]_{u,v,x,y,z} \leq \mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2}) \ .$$

This is consistent with the fact that match_2 is correct w.r.t. match_ω. The 2-sharing groups $u^\infty v^\infty x^\infty$, $v^\infty x^\infty$ and vxz do not appear in $\alpha_2(\mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2}))$ since match_2 is not complete w.r.t. match_ω

As anticipated before, we prove the optimality of match_2 w.r.t. match_ω, which automatically entails optimality w.r.t. match.

Theorem 4.5 (Correctness and optimality of match_2*).*
The operator match_2 is correct and optimal w.r.t. match_ω.

Proof
We need to prove that, for each $[T_1]_{U_1}, [T_2]_{U_2} \in \mathtt{ShLin}^2$,

$$\mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2}) = \alpha_2(\mathsf{match}_\omega(\gamma_2([T_1]_{U_1}), \gamma_2([T_2]_{U_2}))) \ . \tag{13}$$

To ease notation, we denote $\gamma_2([T_1]_{U_1})$ and $\gamma_2([T_2]_{U_2})$ by $[S_1]_{U_1}$ and $[S_2]_{U_2}$, respectively. Moreover, we denote with S_2', S_2'', T_2', and T_2'' the subsets of S_2 and T_2 given accordingly to Definitions 3.4 and 4.3. Since $\lfloor\!\lfloor B \rfloor\!\rfloor = \lfloor\!\lfloor \alpha_2(B) \rfloor\!\rfloor$, given $B \in S_2$, we have that $B \in S_2'$ iff $\alpha_2(B) \in T_2'$.

Let $o \in \mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2})$. If $o \in T_2'$, consider any $B \in \alpha_2^{-1}(o) \subseteq S_2$. Then, $B \in S_2' \subseteq \mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})$. Therefore, $o = \alpha_2(B)$ is in the right-hand side of (13). If $o \notin T_2'$, then $o \leq o'$ where $o'_{|U_1} \in T_1$ and $o'_{|U_2} \in (T_2'')^*$; that is, there is $X \subseteq T_2'' \cup (T_2'')^2$ such that $o'_{|U_2} = \biguplus X$. For each $o'' \in X \cap T_2$, let $B_{o''} \in \alpha_2^{-1}(o'')$. For each $o'' \in X \setminus T_2$, we have $o'' = (o''')^2$ for some $o''' \in T_2$, and let $B_{o''} \in \alpha_2^{-1}(o''')$. Let $\mathcal{X}$ be a multiset containing a single copy of each B_o'' for $o \in X \cap T_2$ and two copies of B_o'' for each

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

$o \in X \setminus T_2$, for example, $\mathcal{X} = \{\!\{ B_{o''} \mid o'' \in X \cap T_2 \}\!\} \uplus \biguplus \{\!\{ \{\!\{ B_{o''}, B_{o''} \}\!\} \mid o'' \in X \setminus T_2 \}\!\})$. Note that $\alpha_2(\biguplus \mathcal{X}) = \biguplus \alpha_2(X) = o'_{|U_2}$ by (3) of Proposition 4.1. Then, consider the ω-sharing group C such that

$$C(v) = \begin{cases} (\biguplus \mathcal{X})(v) & \text{for } v \in U_2, \\ o'(v) & \text{if } v \in U_1 \setminus U_2 \text{ and } o(v) \leq 1, \\ 2 & \text{otherwise.} \end{cases}$$

It is clear that $\alpha_2(C_{|U_1}) = o'_{|U_1}$, hence $C_{|U_1} \in S_1$. Moreover, $C_{|U_2} = \biguplus X$ with $\mathcal{X} \in \mathscr{P}_m(S''_2)$. Therefore, we have $C \in \mathsf{match}_\omega([T_1]_{U_1}, [T_2]_{U_2})$ and $\alpha_2(C) = o'$ is in the right-hand side of (13). The same holds for o by downward closure of α_2.

Conversely, let $o \in \alpha_2(\mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2}))$. As a consequence, there exists $B \in \mathsf{match}_\omega([S_1]_{U_1}, [S_2]_{U_2})$ such that $o \leq o'$ and $o' = \alpha_2(B)$. It is enough to prove that $o' \in \mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2})$. If $B \in S'_2$ then $o' \in T'_2$, hence $o' \in \mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2})$. If $B \notin S'_2$, then $B_{|U_1} \in S_1$ and $B_{|U_2} = \biguplus \mathcal{X}$, where $\mathcal{X} \in \mathscr{P}_m(S''_2)$. By $B_{|U_1} \in S_1$, we have $o'_{|U_1} = \alpha_2(B_{|U_1}) \in T_1$. For each $C \in \mathcal{X}$ with $\mathcal{X}(C) = 1$, we define $o_C = \alpha_2(C) \in T''_2$, while for each C with $\mathcal{X}(C) > 1$, we define $o_C = \alpha_2(C)^2 \in (T''_2)^2$. We have that $o''_{|U_2} = \alpha_2(B)_{|U_2} = \alpha_2(B_{|U_2}) = \alpha_2(\biguplus \mathcal{X}) = \biguplus X$ where $X = \{o_C \mid C \in \mathcal{X}\}$ is an element of $\mathscr{P}(T''_2 \cup (T''_2)^2)$. This means that $o' \in \mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2})$. □

Although match_2 is not complete w.r.t. either match_ω or match, we claim it enjoys a property analogous to the one in Lemma 3.9.

4.3 Optimization

In a real implementation, we would like to encode an element of $\mathtt{ShLin}^2$ with the set of its maximal elements. This works well only if we may compute match_2 starting from its maximal elements, without implicitly computing the downward closure. We would also like match_2 to compute as few non-maximal elements as possible. We provide a new algorithm for match_2 following this approach.

Definition 4.6.
Given T_1, T_2 sets of 2-sharing groups and $U_1, U_2 \subseteq \mathcal{V}$, we define

$$\mathsf{match}'_2(T_1, U_1, T_2, U_2) = T'_2 \cup \bigcup_{o \in T_1} \mathsf{match}'_2(o) \ ,$$

where

$$\mathsf{match}'_2(o) = \left\{ \left(o \wedge \biguplus X \right) \uplus \biguplus (X \cap \overline{T}) \mid X \subseteq T''_2, \left(\biguplus \llbracket X \rrbracket \right)_{|U_1} \leq o_{|U_2} \right\} \ ,$$

with T'_2 and T''_2 as in Definition 4.3, $\overline{T} = \{o' \in T''_2 \mid \forall v \in o' \cap U_1, o(v) = \infty\}$ and

$$o \wedge o' = \lambda v. \begin{cases} o(v) & \text{if } v \in U_1 \setminus U_2, \\ min(o(v), o'(v)) & \text{if } v \in U_1 \cap U_2, \\ o'(v) & \text{otherwise.} \end{cases}$$

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

The operator match'_2 aims at computing the set of maximal 2-sharing groups in $\mathsf{match}_2([\downarrow T_1]_{U_1}, [\downarrow T_2]_{U_2})$. It works by considering one sharing group $o \in T_1$ at a time and calling an auxiliary operator that computes the maximal 2-sharing groups compatible with o. A 2-sharing group o' is compatible with o if $\lfloor\!\lfloor o' \rfloor\!\rfloor \cap U_1 = \lfloor\!\lfloor o \rfloor\!\rfloor \cap U_1$.

When computing the auxiliary operator $\mathsf{match}'_2(o)$ we choose a subset X of T''_2. Given $o' \in X$, note that $\lfloor\!\lfloor o' \rfloor\!\rfloor$ may be viewed as a 2-sharing group which is the linearized version of o': it has the same support as o', but all variables are linear. If $o' \in T''_2$, its linearization is in $\downarrow T_2$. The choice of X is valid if the multiset sum of all its linearizations is smaller than o for all the variables in $U_1 \cap U_2$. Once established that X is a valid choice, we do not take directly $\biguplus \lfloor\!\lfloor X \rfloor\!\rfloor$ as the resulting 2-sharing group, but we try to find an $o'' \geq \biguplus X$.

To this purpose, we observe that, given $o' \in X$, if $o(v) = \infty$ for each $v \in \lfloor\!\lfloor o' \rfloor\!\rfloor \cap U_1$, then we may take o' twice. We denote with $\overline{T}$ the set of all the sharing groups in T''_2, which is such a property, and we unconditionally add to the result the sharing groups in $X \cap \overline{T}$ so that all these sharing groups are taken twice. Therefore, the biggest element compatible with o is $(o \wedge \biguplus X) \uplus \biguplus (X \cap \overline{T})$.

Example 4.7.
Under the hypothesis of Examples 3.5 and 4.2, let us define $T_1 = \{x^\infty, xz\}$ and $T_2 = \{uv, ux, vx^\infty, x\}$. We have $T'_2 = \{uv\}$ and $T''_2 = \{ux, vx^\infty, x\}$.

Let us compute $\mathsf{match}'_2(x^\infty)$. We have $\overline{T} = T''_2$. If we take $X = \{ux, xv^\infty\}$, we have $(\biguplus \lfloor\!\lfloor X \rfloor\!\rfloor)_{|U_1} = x^\infty$, hence the choice is valid. The corresponding result is $(x^\infty \wedge \biguplus X) \uplus \biguplus X = uv^\infty x^\infty \uplus uv^\infty x^\infty = u^\infty v^\infty x^\infty$. Overall, we have $\mathsf{match}'_2(x^\infty) = \{u^\infty x^\infty, v^\infty x^\infty, x^\infty, u^\infty x^\infty v^\infty\}$. Note that some results are computed by different choices of X. For example, both $\{ux, x\}$ and $\{ux\}$ generates $u^\infty x^\infty$.

Let us compute $\mathsf{match}'_2(xz)$ and take $X = \{vx^\infty\}$. We have that $(\biguplus \lfloor\!\lfloor X \rfloor\!\rfloor)_{|U_1} = x = xz_{|U_2}$. In this case, $\overline{T''_2} = \emptyset$. Therefore, the result is $x \wedge vx^\infty = vx$. Note that if we take $X = \{xz, x\}$, then $(\biguplus \lfloor\!\lfloor X \rfloor\!\rfloor)_{|U_1} = x \uplus x = x^\infty$, and the choice is not valid. This shows that, due to the downward closure, choosing in X a sharing group with a nonlinear variable is very different from choosing the same variable twice. At the end, we have $\mathsf{match}'_2(xz) = \{uxz, vxz, xz\}$. Finally,

$$\mathsf{match}_2(T_1, U_1, T_2, U_2) = \{uv, u^\infty x^\infty, v^\infty x^\infty, x^\infty, u^\infty x^\infty v^\infty, uxz, vxz, xz\} \ .$$

This is exactly the set of maximal elements in $\mathsf{match}_2([\downarrow T_1]_{U_1}, [\downarrow T_2]_{U_2})$.

We can show that the correspondence between match_2 and match'_2 in the previous example was not by chance, proving the following:

Theorem 4.8.
Given T_1, T_2 sets of 2-sharing groups such that $\downarrow[T_1]_{U_1}, \downarrow[T_2]_{U_2} \in \mathtt{ShLin}^2$, we have

$$\mathsf{match}_2(\downarrow[T_1]_{U_1}, \downarrow[T_2]_{U_2}) = \downarrow[\mathsf{match}'_2(T_1, U_1, T_2, U_2)]_{U_1 \cup U_2} \ .$$

Proof
We start by proving that if o is an element in $\mathsf{match}_2(\downarrow[T_1]_{U_1}, \downarrow[T_2]_{U_2})$, then $o \in \downarrow \mathsf{match}'_2(T_1, U_1, T_2, U_2)$. If $o \in \downarrow T'_2$, this is trivial since $T'_2 \subseteq \mathsf{match}'_2(T_1, U_1, T_2, U_2)$. Otherwise, $o_{|U_1} \in \downarrow T_1$ and $o_{|U_2} \in (\downarrow T''_2)^*$, according to Definition 4.3. Let $o_1 \in T_1$ such that $o_1 \geq o_{|U_1}$, we want to prove that there exists $\bar{o} \in \mathsf{match}'_2(o_1)$ such that $o \leq \bar{o}$.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

Since $o_{|U_2} \in (\downarrow T_2'')^*$, there are $X_a \subseteq \downarrow T_2''$ and $X_b \subseteq (\downarrow T_2'')^2$ such that $o_{|U_2} = \biguplus X_a \uplus \biguplus X_b$. For each $o_a \in X_a$, consider $o_a' \in T_2''$ such that $o_a' \geq o_a \in T_2''$. Let Y_a be the set of all those o_a'. For each $o_b \in X_b$, consider an $o_b' \in T_2''$ such that $(o_b')^2 = o_b$. Let Y_b be the set of all those o_b'. By construction $\biguplus Y_a \geq \biguplus X_a \geq \biguplus [\![Y_a]\!]$ and $\biguplus Y_b \uplus \biguplus Y_b = \biguplus [\![Y_b]\!] \uplus \biguplus [\![Y_b]\!] = \biguplus X_b$.

Let $Y = Y_a \cup Y_b$. Obviously $Y \subseteq T_2''$ and $(\biguplus [\![Y]\!])_{|U_1} = \biguplus [\![Y_a]\!] \uplus \biguplus [\![Y_b]\!] \leq (\biguplus X_a \uplus \biguplus X_b)_{|U_1} = (o_{|U_2})_{|U_1} = (o_{|U_1})_{|U_2} \leq (o_1)_{|U_2}$. Therefore, $\bar{o} = (o_1 \wedge \biguplus Y) \uplus \biguplus (Y \cap \overline{T}) \in \mathsf{match}_2'(o_1)$. We now prove that $o \leq \bar{o}$.

Given any $o_b \in X_b$, we have that $o_1(v) = o_b(v) = \infty$ for each $v \in o_b \cap U_1$. This implies that $Y_b \subseteq \overline{T}$. Therefore, $\bar{o} \geq (o_1 \wedge \biguplus Y_a \uplus \biguplus Y_b) \uplus \biguplus Y_b = (o_1 \wedge \biguplus Y_a) \uplus \biguplus Y_b \uplus \biguplus Y_b \geq (o_1 \wedge \biguplus X_a) \uplus \biguplus X_b = o_1 \wedge (\biguplus X_a \uplus \biguplus X_b) = o_1 \wedge o_{|U_2}$. Now, if $v \in U_1 \setminus U_2$, we have $\bar{o}(v) = o_1(v) \geq o(v)$. If $v \in U_2 \setminus U_1$, we have $\bar{o}(v) \geq o_{|U_2}(v) = o(v)$. Finally, if $v \in U_1 \cap U_2$, then $o_1(v) \geq o_{|U_1}(v) = o_{|U_2}(v) = o(v)$, hence $\bar{o}(v) \geq o(v)$. This concludes one side of the proof.

For the other side of the equality, assume $o \in \mathsf{match}_2'(T_1, U_1, T_2, U_2)$ and prove $o \in \mathsf{match}_2(\downarrow [T_1]_{U_1}, \downarrow [T_2]_{U_2})$. If $o \in T_2'$, this is trivial since $\mathsf{match}_2(\downarrow [T_1]_{U_1}, \downarrow [T_2]_{U_2}) \supseteq \downarrow T_2'$. Otherwise, $o = (o_1 \wedge \biguplus X) \uplus \biguplus (X \cap \overline{T})$ for some $o_1 \in T_1$ and $X \subseteq T_2''$ satisfying the properties of Definition 4.6. Consider $Y = (X \setminus \overline{T}) \cup (X \cap \overline{T})^2$, which is an element of $T_2'' \cup (T_2'')^2$ such that $\biguplus Y = \biguplus X \uplus \biguplus (X \cap \overline{T}) \in (T_2'')^*$. It is enough to prove that $o_{|U_1} = o_1$ and $o_{|U_2} = \biguplus Y$. If $v \in U_1 \setminus U_2$, we have $o(v) = o_1(v)$. Note that if $\bar{o} \in X \cap \overline{T}$ and $v \in \bar{o} \cap U_1$, then $o_1(v) = \infty$. Therefore, for each $v \in U_1 \cap U_2$, we have $o(v) = o_1(v)$. Finally, if $v \in U_2$, we have $o(v) = \biguplus X \uplus \biguplus (X \cap \overline{T}) = \biguplus Y$. □

5 Abstract matching over Sharing × Lin

The reduced product $\mathtt{ShLin} = \mathtt{Sharing} \times \mathtt{Lin}$ has been used for a long time in the analysis of aliasing properties since it was recognized quite early that the precision of these analyses could be greatly improved by keeping track of the linear variables. In the following, we briefly recall the definition of the abstract domain following the presentation in Amato and Scozzari (2010).

5.1 The domain ShLin

The domain ShLin couples an object of Sharing with the set of variables known to be linear. Each element of ShLin is therefore a triple: the first component is an object of Sharing; the second component is an object of Lin, that is, the set of variables that are linear in all the sharing groups of the first component; and the third component is the set of variables of interest. It is immediate that ShLin is an abstraction of $\mathtt{ShLin}^2$ (and thus of $\mathtt{ShLin}^\omega$).

$$\mathtt{ShLin} = \{[S, L, U] \mid S \subseteq \mathscr{P}(U), (S \neq \emptyset \Rightarrow \emptyset \in S), L \supseteq U \setminus \mathrm{vars}(S), U \in \mathscr{P}_f(\mathcal{V})\} \ ,$$

with the approximation relation $\leq_{sl}$ defined as $[S, L, U] \leq_{sl} [S', L', U']$ iff $U = U'$, $S \subseteq S'$, $L \supseteq L'$. There is a Galois insertion of ShLin into $\mathtt{ShLin}^2$ given by the pair of maps:

$$\alpha_{sl}([T]_U) = [\{[\![o]\!] \mid o \in T\}, \{x \in U \mid \forall o \in T.\ o(x) \leq 1\}, U] \ ,$$
$$\gamma_{sl}([S, L, U]) = [\downarrow \{B_L \mid B \in S\}]_U \ ,$$

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

where B_L is the 2-sharing group that has the same support of B, with linear variables dictated by the set L:

$$B_L = \lambda v \in \mathcal{V}. \begin{cases} \infty & \text{if } v \in B \setminus L, \\ 1 & \text{if } v \in B \cap L, \\ 0 & \text{otherwise.} \end{cases}$$

The functional composition of α_ω, α_2, and α_{sl} gives the standard abstraction map from substitutions to `ShLin`. We still use the polynomial notation to represent sharing groups, but now all the exponents are fixed to one. Note that the last component U in $[S, L, U]$ is redundant since it can be retrieved as $L \cup \mathrm{vars}(S)$. This is because the set L contains all the ground variables.

Example 5.1.
Consider the substitution $[\theta]_U = [\{x/s(y,u,y), z/s(u,u), v/u\}]_{w,x,y,z}$ in Example 3.2. Its abstraction in `ShLin` is given by

$$\alpha_{sl}(\alpha_2(\alpha_\omega([\theta]_U))) = [\{xy, xz, w\}, \{y, w\}, U] \ .$$

Analogously, substitutions $[\theta_1]_{U_1} = [\{x/r(w_1,w_2,w_2,w_3,w_3), y/a, z/r(w_1)\}]_{x,y,z}$ and $[\theta_2]_{U_2} = [\{x/r(w_4,w_5,w_6,w_8,w_8), u/r(w_4,w_7), v/r(w_7,w_8)\}]_{u,v,x}$ from Example 3.3 are abstracted into $[S_1, L_1, U_1] = [\{x, xz\}, \{y, z\}, U_1]$ and $[S_2, L_2, U_2] = [\{uv, ux, vx, x\}, \{u, v\}, U_2]$, respectively.

5.2 Matching operator

We want to provide an optimal abstract matching operator for `ShLin`. We may effectively compute match_{sl} by composing γ_{sl}, match_2, and α_{sl}. However, we provide a more direct characterization of match_{sl}, which may potentially improve performance.

First, we define the auxiliary function $nl : \mathscr{P}(\mathscr{P}(\mathcal{V})) \to \mathscr{P}(\mathcal{V})$ which takes a set X of sharing groups and returns the set of variables that appear in X more than once. In formulas:

$$nl(X) = \{v \in \mathcal{V} \mid \exists B_1, B_2 \in X, B_1 \neq B_2, v \in B_1 \cap B_2\} \tag{14}$$

The name *nl* stands for nonlinear since it is used to recover those variables that, after joining sharing groups in X, are definitively not linear.

Definition 5.2 (Matching over `ShLin`*).*
Given $[S_1, L_1, U_1]$ and $[S_2, L_2, U_2] \in \mathtt{ShLin}$, we define

$$\mathsf{match}_{sl}([S_1, L_1, U_1], [S_2, L_2, U_2]) = [s(S_0' \cup S_0''), l(S_0' \cup S_0''), U] \,,$$

where $U = U_1 \cup U_2$, $S_2' = \{B \in S_2 \mid B \cap U_1 = \emptyset\}$, $S_2'' = S_2 \setminus S_2'$, $\overline{S} = \{B \in S_2'' \mid B \cap L_1 = \emptyset\}$ and

$$\begin{aligned} S_0' &= \{\langle B, L_2 \rangle \mid B \in S_2'\}, \\ S_0'' &= \{ \left\langle B \cup \bigcup X, L_2 \setminus nl(X) \setminus \bigcup (X \cap \overline{S}) \right\rangle \mid B \in S_1, \\ &\qquad X \subseteq S_2'', B \cap U_2 = \left(\bigcup X\right) \cap U_1, L_1 \cap nl(X) = \emptyset\}, \end{aligned}$$

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

$$s(H) = \{B \mid \langle B, L\rangle \in H\}\,,$$
$$l(H) = \bigcap \{L_1 \cup L \cup (U \setminus B) \mid \langle B, L\rangle \in H\}\,.$$

This operator is more complex than previous ones because linearity information is not connected to sharing groups and needs to be handled separately. In view of this and a similar situation that happens for unification (Amato and Scozzari, 2010), it seems that the idea of embedding linearity within sharing groups from King (1994) was particularly insightful.

We give an intuitive explanation of match_{sl}. It essentially works by simulating the optimized version of match_2 starting from $[T_1]_{U_1} = \gamma_2([S_1, L_1, U_1])$ and $[T_2]_{U_2} = \gamma_2([S_2, L_2, U_2])$. The sets S_2' and S_2'' are defined as for the other matching operators. Each element of H, S_0', or S_0'' is a pair $\langle B, L\rangle$ that corresponds to the 2-sharing group $B_{L\cup L_1}$ in $\mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2})$. The component B is the support of the 2-sharing group, while L is the set of linear variables. We only record a subset of the linear variables, since those in L_1 are always linear.

The set S_0' encodes all the maximal 2-sharing groups derived from S_2'. The set S_0'' encodes 2-sharing groups which may be generated by gluing the sharing groups in S_2'' in a way which is compatible with S_1. A given choice of $X \subseteq T_2''$ is compatible with a sharing group $B \in S_1$ only if $\bigcup X$ and B have the same support on the common variables $U_1 \cap U_2$ and if X does not conflict with the linearity of variables given by L_1. This means that we cannot use a variable in L_1 more than once; therefore, $L_1 \cap nl(X) = \emptyset$. Note that we may use a variable $v \in L_1 \setminus L_2$ since $v \notin L_2$ means that v is possibly, but not definitively, nonlinear. On the contrary, if $v \in nl(X)$, then v is definitively nonlinear in the resultant sharing group. Once established that X is compatible with B, the set $\overline{S}$ plays the same role of $\overline{T}$ in Definition 4.6: we may join another copy of the sharing groups in $X \cap \overline{S}$, making all their variables nonlinear.

Finally, the maps s and l extract from S_0' and S_0'', the set of all the sharing groups and the set of variables that are linear in all the sharing groups.

Example 5.3.
Consider the substitutions in Example 5.1. According to the definition of match_{sl}, we have $S_2' = \{uv\}$, $S_2'' = \overline{S} = \{ux, vx, x\}$, $S_0' = \{\langle uv, \{u, v\}\rangle\}$, and $S_0'' = \{\langle ux, \emptyset\rangle, \langle vx, \emptyset\rangle, \langle x, \emptyset\rangle, \langle uvx, \emptyset\rangle, \langle uxz, \emptyset\rangle, \langle vxz, \emptyset\rangle, \langle xz, \emptyset\rangle, \langle uvxz, \emptyset\rangle\}$. The final result is

$$\mathsf{match}_{sl}([S_1, L_1, U_1], [S_2, L_2, U_2]) = [\{uv, uvx, ux, vx, x, uvxz, uxz, xz, vxz\}, \{y, z\}, U_1 \cup U_2]\ . \tag{15}$$

Considering the results for match_2 in Example 4.4, we have

$$\begin{aligned}&\alpha_{sl}(\mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2}))\\ &= \alpha_{sl}(\downarrow[uv, u^\infty v^\infty x^\infty, uxz, u^\infty x^\infty, v^\infty x^\infty, vxz, x^\infty, xz]_{u,v,x,y,z})\\ &= [\{uv, uvx, uxz, ux, vx, vxz, x, xz\}, \{y, z\}, \{u, v, x, y, z\}]\end{aligned}$$

and $\mathsf{match}_{sl}([S_1, L_1, U_1], [S_2, L_2, U_2]) > \alpha_{sl}(\mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2}))$. In particular, the sharing group $uvxz$ does not appear in $\alpha_{sl}(\mathsf{match}_2([T_1]_{U_1}, [T_2]_{U_2}))$ which proves that match_{sl} is not a complete abstraction of match_2. However, the next theorem shows that match_{sl} is optimal w.r.t. match_2 and by composition also w.r.t. match_ω and match.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

Theorem 5.4 (Optimality of match_{sl}*).*
The operator match_{sl} is correct and optimal w.r.t. match_2.

Proof
We prove that, given $[S_1, L_1, U_1]$ and $[S_2, L_2, U_2] \in \mathtt{ShLin}$, we have

$$\mathsf{match}_{sl}([S_1, L_1, U_1], [S_2, L_2, U_2]) = \alpha_{sl}(\downarrow[\mathsf{match}'_2(T_1, U_1, T_2, U_2)]_U)$$

where $T_i = \{B_{L_i} \mid B \in S_i\}$ is the set of maximal elements of $\gamma_{sl}([S_i, L_i, U_i])$ and $U = U_1 \cup U_2$.

Let us define the function γ_0 which maps a pair $\langle B, L\rangle$ with $B, L \in \mathscr{P}(\mathcal{V})$ to the 2-sharing group $B_{L\cup L_1}$. We show that, given $H \subseteq \mathscr{P}(\mathcal{V}) \times \mathscr{P}(\mathcal{V})$, we have $\alpha_{sl}([\downarrow\gamma_0(H)]_U) = [s(H), l(H), U]$. Assume $\alpha_{sl}([\downarrow\gamma_0(S)]_U]) = [R, V, U]$. We have $R = \{\!|o|\!| \mid o \in \downarrow\gamma_0(S)\} = \{\!|o|\!| \mid o \in \gamma_0(S)\} = \{\!|\gamma_0(\langle B, L\rangle)|\!| \mid \langle B, L\rangle \in S\} = \{B \mid \langle B, L\rangle \in S\}$ $= s(S)$ and $V = \{x \in U \mid \forall o \in \downarrow\gamma_0(S), o(x) \leq 1\} = \{x \in U \mid \forall o \in \gamma_0(S), o(x) \leq 1\} = \{x \in U \mid \forall \langle B, L\rangle \in S, B_{L\cup L_1}(x) \leq 1\}\} = \{x \in U \mid \forall \langle B, L\rangle \in S, x \in L_1 \cup L \cup (U \setminus B)\} = \bigcap\{L_1 \cup L \cup (U \setminus B) \mid \langle B, L\rangle \in S\} = l(S)$.

Therefore, if we prove that $\gamma_0(S'_0 \cup S''_0) = \mathsf{match}'_2(T_1, U_1, T_2, U_2)$, then we have $\mathsf{match}_{sl}([S_1, T_1, U_1], [S_2, T_2, U_2]) = [s(S'_0 \cup S''_0), l(S'_0 \cup S''_0), U] = \alpha_{sl}([\downarrow\gamma_0(S'_0 \cup S''_0)]_U]) = \alpha_{sl}(\downarrow[\mathsf{match}'_2(T_1, U_1, T_2, U_2)]_U))$.

Let us take $o \in \gamma_0(S'_0 \cup S''_0)$ and prove $o \in \mathsf{match}'_2(T_1, U_1, T_2, U_2)$. If $o = B_{L\cup L_1}$ for some $\langle B, L\rangle \in S'_0$, then $B \in S'_2$ and $L = L_2$. Therefore, $B_{L\cup L_1} = B_{L_2\cup L_1} = B_{L_2} \in T_2$ since $B \cap U_1 = \emptyset$. In particular, $B_{L_2} \in T'_2$, hence $B_{L_2} \in \mathsf{match}'_2(T_1, U_1, T_2, U_2)$. Otherwise, $o = \gamma_0(\langle C, L\rangle)$ for $\langle C, L\rangle \in S''_0$, where $C = B \cup \bigcup X$ and $L = L_2 \setminus nl(X) \setminus \bigcup(X \cap \overline{S})$ according to Definition 5.2. For each sharing group $B' \in X$, consider the 2-sharing group $B'_{L_2} \in T''_2$, and let Y be the set of all those 2-sharing groups. We want to prove that o is generated by $B_{L_1} \in T_1$ and $Y \subseteq T''_2$, according to Definition 4.6. First, note that $\mathrm{I}(\biguplus \mathrm{I}Y\mathrm{I})_{|U_1}\mathrm{I} = \mathrm{I}(\biguplus X)_{|U_1}\mathrm{I} = (\bigcup X) \cap U_1 = B \cap U_2 = \mathrm{I}(B_{L_1})_{|U_2}\mathrm{I}$. Now, assume $v \in \mathrm{I}(B_{L_1})_{|U_2}\mathrm{I}$. If v is linear in $(B_{L_1})_{|U_2}$, then $v \in L_1$, hence $v \notin nl(X)$ and v is linear in $(\biguplus X)_{|U_1} = (\biguplus \mathrm{I}Y\mathrm{I})_{|U_1}$. Therefore, Y is a valid choice for match'_2 and $\mathsf{match}'_2(B)$ generates $o' = (B_{L_1} \wedge \biguplus Y) \uplus \biguplus(Y \cap \overline{T})$. We prove $o' = o$.

First, we prove that $B' \in \overline{S}$ iff $B'_{L_2} \in \overline{T}$. We have $B'_{L_2} \in \overline{T}$ iff $B'_{L_2} \in T''_2$ and $\forall v \in \mathrm{I}B'_{L_2}\mathrm{I} \cap U_1, B'_{L_1}(v) = \infty$, iff $B' \in S''_2$ and $\forall v \in B' \cap U_1, v \notin L_1$ iff $B' \cap L_1 = \emptyset$ iff $B' \in S''_2$ and $B' \cap L_1 = \emptyset$ iff $B \in S$.

Now, it is immediate to check that, $\mathrm{I}o\mathrm{I} = B \cup \bigcup X = \mathrm{I}o'\mathrm{I}$. Then, consider $v \in \mathrm{I}o'\mathrm{I}$. We have that

$$\begin{aligned}
&o'(v) = 1 \Leftrightarrow \\
&(B_{L_1}(v) = 1 \vee \left(\biguplus Y\right)(v) = 1) \wedge \left(\biguplus(Y \cap \overline{T})\right) = 0 \Leftrightarrow \\
&(v \in L_1 \vee (v \in L_2 \setminus nl(X))) \wedge v \notin \bigcup(X \cap \overline{S}) \Leftrightarrow \\
&v \in L_1 \vee (v \in L_2 \setminus nl(X) \setminus \bigcup(X \cap \overline{S}) \Leftrightarrow \\
&v \in L_1 \vee v \in L \Leftrightarrow \\
&B_{L\cup L_1}(v) = o(v) = 1
\end{aligned}$$

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

It remains to prove that given $o \in \mathsf{match}'_2(T_1, U_1, T_2, U_2)$, we have $o \in \gamma_0(S'_0 \cup S''_0)$. If $o \in T'_2$, then $o = B_{L_2}$ with $B \in S'_2$. Therefore, $\langle B, L_2\rangle \in S'_0$ and $o = \gamma_0(\langle B, L_2\rangle)$. Otherwise, $o = (o' \wedge \biguplus Y) \uplus \biguplus(Y \cap \overline{T})$. We know $o' = B'_{L_1}$ with $B' \in S_1$ and let $X = \{\lfloor\!\lfloor o'' \rfloor\!\rfloor \mid o'' \in Y\}$. By Definition 4.6, $\bigcup X \cap U_1 = \lfloor\!\lfloor \uplus Y \rfloor\!\rfloor \cap U_1 = \lfloor\!\lfloor o' \rfloor\!\rfloor \cap U_2 = B' \cap U_2$. Moreover, if $v \in L_1$, then $o'(v) \leq 1$, and therefore, v cannot appear twice in Y, which means $v \notin nl(X)$; hence $L_1 \cap nl(X) = \emptyset$. Therefore, B' and X make a valid choice for match_{sl} and generate the pair

$$\langle B, L\rangle = \left\langle B' \cup \bigcup X, L_2 \setminus nl(X) \setminus \bigcup(X \cap \overline{S})\right\rangle .$$

Using the first half of the proof, it is easy to check that $B_{L \cup L'} = o$, which terminates the proof. □

6 Evaluation of matching in goal-dependent analysis

We now show two examples, in the context of goal-dependent analysis, where the newly introduced matching operators improve the precision w.r.t. what is attainable using only the mgu operators in Amato and Scozzari (2010).

6.1 Matching and backward unification

First, we show with a simple example how the matching operator strictly improves the result of a standard goal-dependent analysis using forward and backward unification. Consider the goal $p(x, f(x, z), z)$ with the (abstract) call substitution $[x, z]_{xz}$ and the trivial program with just one clause:
$p(u, v, w)$.

In order to analyze the goal, we first need to perform the forward unification between the call substitution $[x, z]_{xz}$, the goal $p(x, f(x, z), z)$, and the head of the clause $p(u, v, w)$, and then project the result on the variables of the clause. In order to keep the notation simple, we do not perform renaming, unless necessary. This amounts to computing:

$$\mathsf{mgu}_\omega([x, z]_{xz}, \{u/x, v/f(x, z), w/z\}) = [uvx, vwz]_{uvwxz} \ .$$

By projecting the result on the variables of the clause, we obtain the entry substitution $[uv, vw]_{uvw}$. Since the clause has no body, it is immediate to see that the exit substitution coincides with the entry substitution.

We now need to compute the backward unification of the exit substitution $[uv, vw]_{uvw}$, the call substitution $[x, z]_{xz}$ and $\theta = \{u/x, v/f(x, z), w/z\}$. If we implement this operator with the aid of matching, we may first unify $[x, z]_{xz}$ with θ, obtaining $[uvx, vwz]_{uvwxz}$ as above, and then apply the matching with the exit substitution $[uv, vw]_{uvw}$ obtaining

$$\mathsf{match}_\omega([uv, vw]_{uvw}, [uvx, vwz]_{uvwxz}) = [uvx, vwz]_{uvwxz} \ .$$

By projecting the result on the variables of the goal, we obtain the result $[x, z]_{xz}$, which proves that x and z do not share.

On the contrary, if we avoid matching, the backward-unification operator must unify $[x, z]_{xz}$ with $[uv, vw]_{uvw}$ and θ in any order it deems fit. However, this means that the operator must correctly approximate the mgu of any substitution θ_1 in the concretization

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

of the call substitution $[x, z]_{xz}$, with any substitution θ_2 in the concretization of the exit substitution $[uv, vw]_{uvw}$ and $\theta = \{u/x, v/f(x, z), w/z\}$. If we choose $\theta_1 = \epsilon$ and $\theta_2 = \{v/f(w, u)\}$, we obtain that the unification of θ_1, θ_2, and θ is

$$\{u/x, v/f(x, x), w/x, z/x\}$$

which is not in the concretization of $[uvx, vwz]_{uvwxz}$ since u, v, and w share a common variable. Moreover, in this substitution, x and z share. This means that, after the projection on the variables of the goals x and z, the result will always include the ω-sharing group xz. Note that this consideration holds for any correct abstract unification operator that avoids matching.

It is worth noting that the above example, with minimal changes, also works for $\texttt{ShLin}^2$ and ShLin: also in these cases, matching improves the precision of the analysis.

6.2 Example on a nontrivial program

Consider a program implementing the *member* predicate. Using the Prolog notation for lists, we have

$member(u, [u|v])$.

$member(u, [v|w]) \leftarrow member(u, w)$.

We want to analyze the goal $member(x, [y])$ in the domain $\texttt{ShLin}^2$ using the *call substitution* $[xy, xz]_{xyz}$.

We start by considering the first clause of *member*. The concrete unification of the goal $member(x, [y])$ and the head of the clause $member(u, [u|v])$ yields the most general unifier $\theta = \{x/u, y/u, v/[]\}$. Forward unification computes the entry substitution as the abstract mgu between the call substitution $[xy, xz]_{xyz}$ and θ. Proceeding one binding at a time, we have

- $\mathsf{mgu}_2([xy, xz]_{xyz}, \{x/u\}) = [uxy, uxz]_{uxyz}$;
- $\mathsf{mgu}_2([uxy, uxz]_{uxyz}, \{y/u\}) = {\downarrow}[u^\infty x^\infty y^\infty]_{uxyz}$;
- $\mathsf{mgu}_2({\downarrow}[u^\infty x^\infty y^\infty]_{uxyz}, \{v/[]\}) = {\downarrow}[u^\infty x^\infty y^\infty]_{uvxyz}$.

Projecting over the variables of the clause, we get the entry substitution ${\downarrow}[u^\infty]_{uv}$. Since this clause has no body, the entry substitution is equal to the exit substitution, and we may proceed to compute the answer substitution through backward unification.

First, we consider the case when the backward unification is performed using the standard mgu_2 operator. We need to unify the call substitution $[xy, xz]_{xyz}$, the exit substitution ${\downarrow}[u^\infty]_{uv}$, and the concrete substitution θ (the same as before). Unifying call and exit substitution is immediate since they are relative to disjoint variables of interest: the result is ${\downarrow}[u^\infty, xy, xz]_{uvxyz}$, obtained by collecting all sharing groups together. This should be unified with θ. Proceeding one binding at a time, and omitting the set of variables of interest since it does not change, we have

- $\mathsf{mgu}_2({\downarrow}[u^\infty, xy, xz], \{x/u\}) = {\downarrow}[u^\infty x^\infty y^\infty, u^\infty x^\infty y^\infty z^\infty, u^\infty x^\infty z^\infty]$;
- $\mathsf{mgu}_2({\downarrow}[u^\infty x^\infty y^\infty, u^\infty x^\infty y^\infty z^\infty, u^\infty x^\infty z^\infty], \{y/u\}) = {\downarrow}[u^\infty x^\infty y^\infty, u^\infty x^\infty y^\infty z^\infty]$;
- $\mathsf{mgu}_2({\downarrow}[u^\infty x^\infty y^\infty, u^\infty x^\infty y^\infty z^\infty], \{v/[]\}) = {\downarrow}[u^\infty x^\infty y^\infty, u^\infty x^\infty y^\infty z^\infty]$.

Projecting over the set of variables in the goal, we get ${\downarrow}[x^\infty y^\infty, x^\infty y^\infty z^\infty]_{xyz}$.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

On the contrary, if we perform backward unification using the matching operation, we need to compute the matching of the exit substitution $[u^\infty]_{uv}$ with the entry substitution before variable projection $[u^\infty x^\infty y^\infty]_{uvxyz}$, namely:

$$\mathsf{match}_2(\downarrow[u^\infty]_{uv}, \downarrow[u^\infty x^\infty y^\infty]_{uvxyz}) = \downarrow[u^\infty x^\infty y^\infty]_{uvxyz} \ .$$

Projecting over the variables of the goal, we get $\downarrow[x^\infty y^\infty]_{xyz}$: using matching we can prove that z is ground in the answer substitution.

However, we still need to check what happens when we analyze the second clause of the *member* predicate. In this case, the concrete unification between $member(x, [y])$ and $member(u, [v|w])$ gives the substitution $\theta = \{x/u, y/v, w/[]\}$. Then, forward unification between the call substitution and θ gives

- $\mathsf{mgu}_2([xy, xz]_{xyz}, \{x/u\}) = [uxy, uxz]_{uxyz}$;
- $\mathsf{mgu}_2([uxy, uxz]_{uxyz}, \{y/v\}) = [uvxy, uxz]_{uvxyz}$;
- $\mathsf{mgu}_2([uvxy, uxz]_{uvxyz}, \{w/[]\}) = [uvxy, uxz]_{uvwxyz}$.

Projecting over the variables of the clause, we get the entry substitution $[uv, v]_{uvw}$.

Now we should compute the answer substitution of the body $member(u, w)$ under the call substitution $[uv, u]_{uvw}$. We could proceed by showing all the details, but we try to be more concise. In the abstract substitution $[uv, w]_{uvw}$, the variable w is known to be ground. When *member* is called with its second argument ground, the first argument becomes ground too. This property is easily captured by `Sharing` and more precise domains, independently of the fact that we use matching or not for the backward unification. Therefore, we can conclude that the answer substitution for the goal $member(u, w)$ under the entry substitution $[uv, u]_{uvw}$ is $[\emptyset]_{uvw}$. Although we do not generally write the empty 2-sharing group $\emptyset$ in an element of $\mathtt{ShLin}^2$, in this case, it is important to write it in order to distinguish $[\emptyset]_{uvw}$, denoting those substitutions in which u, v, w are ground, from $[]_{uvw}$, denoting a non-succeeding derivation.

Performing the backward unification of $[\emptyset]_{uvw}$, we get the answer substitution $[\emptyset]_{xyz}$, independently of the use of matching.

Putting together the results we got for analyzing the goal $member(x, [y])$ according to the two clauses of the program, we have shown that using matching, we get $[x^\infty y^\infty]_{xyz}$, while using standard unification, we get $[x^\infty y^\infty, x^\infty y^\infty z^\infty]_{xyz}$, and we are not able to prove that z is ground after the goal returns.

7 Conclusion

In this paper, we have extended the domain $\mathtt{ShLin}^\omega$ (Amato and Scozzari, 2010) to goal-dependent analysis, by introducing a matching operator, and proved its optimality. From this operator, we have derived the optimal matching operators for the well-known $\mathtt{ShLin}^2$ (King, 1994) and $\mathtt{Sharing} \times \mathtt{Lin}$ (Muthukumar and Hermenegildo, 1992) abstract domains.

As far as we know, this is the first paper that shows matching optimality results for domains combining sharing and linearity information. In particular, the matching operators presented in Hans and Winkler (1992) and King (2000) for the domain *SFL*,

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

which combines set-sharing, linearity, and freeness information, are not optimal, as shown by Amato and Scozzari (2009).

Recently logic programming has been used as an intermediate representation for the analysis of imperative or object-oriented programs and services (see, e.g., Peralta et al. (1998); Henriksen and Gallagher (2006); Benton and Fischer (2007); Méndez-Lojo et al., (2008); Spoto et al. (2010); Albert et al. (2012); Ivanović et al., (2013); Gange et al. (2015); De Angelis et al. (2021)). Since many of these approaches use existing logic program analysis on the transformed program, we believe that they can benefit from more precise logic program analysis.

Acknowledgements

We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), Spoke 9 - Green-aware AI, under the NRRP MUR program funded by the NextGenerationEU.

Competing interests

The authors declare none.

References

Albert, E., Arenas, P., Genaim, S., Puebla, G. and Zanardini, D. 2012. Cost analysis of object-oriented bytecode programs. *Theoretical Computer Science* 413, 1, 142–159, Special Issue on Quantitative Aspects of Programming Languages (QAPL 2010).

Amato, G. and Scozzari, F. 2009. Optimality in goal-dependent analysis of sharing. *Theory and Practice of Logic Programming* 9, 5, 617–689.

Amato, G. and Scozzari, F. 2010. On the interaction between sharing and linearity. *Theory and Practice of Logic Programming* 10, 1, 49–112.

Amato, G. and Scozzari, F. 2011. Observational completeness on abstract interpretation. *Fundamenta Informaticae* 106, 2-4, 149–173.

Amato, G. and Scozzari, F. 2014. Optimal multibinding unification for sharing and linearity analysis. *Theory and Practice of Logic Programming* 14, 3, 379–400.

Armstrong, T., Marriott, K., Schachte, P. and Søndergaard, H. 1998. Two classes of boolean functions for dependency analysis. *Science of Computer Programming* 31, 1, 3–45.

Bagnara, R., Zaffanella, E. and Hill, P. M. 2005. Enhanced sharing analysis techniques: a comprehensive evaluation. *Theory and Practice of Logic Programming* 5, 1-2, 1–43.

Benton, W. C. and Fischer, C. N. 2007. Interactive, scalable, declarative program analysis: From prototype to implementation. Leuschel, M. and Podelski, A., Eds. In *PPDP '07: Proceedings of the 9th ACM SIGPLAN international conference on Principles and practice of declarative programming*, ACM, New York, NY, USA, 13–24.

Bruynooghe, M. 1991. A practical framework for the abstract interpretation of logic programs. *The Journal of Logic Programming* 10, 1/2/3 & 4, 91–124.

Codish, M., Mulkers, A., Bruynooghe, M., de la Banda, M. G. and Hermenegildo, M. 1995. Improving abstract interpretations by combining domains. *ACM Transactions on Programming Languages and Systems* 17, 1, 28–44.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press

Cortesi, A., Filé, G. and Winsborough, W. W. 1996. Optimal groundness analysis using propositional logic. *The Journal of Logic Programming* 27, 2, 137–167.

Cousot, P. and Cousot, R. 1979. Systematic design of program analysis frameworks. Aho, A. V. and Zilles, S. N., Eds. In *POPL '79: Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, ACM, New York, NY, USA, 269–282.

Cousot, P. and Cousot, R. 1992a. Abstract interpretation and applications to logic programs. *The Journal of Logic Programming* 13a, 2-3, 103–179.

Cousot, P. and Cousot, R. 1992b. Abstract interpretation frameworks. *Journal of Logic and Computation* 2b, 4, 511–549.

De Angelis, E., Fioravanti, F., Gallagher, J. P., Hermenegildo, M. V., Pettorossi, A. and Proietti, M. 2021. Analysis and transformation of constrained horn clauses for program verification. *Theory and Practice of Logic Programming* 22, 6, 974–1042.

de la Banda, M. G. and Hermenegildo, M. 1993. A practical approach to the global analysis of CLP programs. In *Proceedings of the 1993 International Symposium on Logic Programming*, The MIT Press, Cambridge, MA, USA, 437–455.

Gange, G., Navas, J. A., Schachte, P., Søndergaard, H. and Stuckey, P. J. 2015. Horn clauses as an intermediate representation for program analysis and transformation. *Theory and Practice of Logic Programming* 15, 4-5, 526–542.

Giacobazzi, R., Ranzato, F. and Scozzari, F. 2000. Making abstract interpretations complete. *Journal of the ACM* 47, 2, 361–416.

Hans, W. and Winkler, S. (1992). *Aliasing and groundness analysis of logic programs through abstract interpretation and its safety.* Technical University of Aachen (RWTH Aachen), http://sunsite.informatik.rwth-aachen.de/Publications/AIB. Last accessed Jan 10, 2024. Technical Report 92-27.

Henriksen, K. S. and Gallagher, J. P. (2006) Abstract interpretation of PIC programs through logic programming, In *2006 Sixth IEEE International Workshop on Source Code Analysis and Manipulation.* Los Alamitos, CA, USA, IEEE Computer Society Press, 184–196.

Hermenegildo, M. V. and Rossi, F. 1995. Strict and nonstrict independent and-parallelism in logic programs: Correctness, efficiency, and compile-time conditions. *The Journal of Logic Programming* 22, 1, 1–45.

Ivanović, D., Carro, M. and Hermenegildo, M. V. 2013. A sharing-based approach to supporting adaptation in service compositions. *Computing* 95, 6, 453–492.

Jacobs, D. and Langen, A. 1992. Static analysis of logic programs for independent AND parallelism. *The Journal of Logic Programming* 13, 2-3, 291–314.

King, A. 2000. Pair-sharing over rational trees. *The Journal of Logic Programming* 46, 1-2, 139–155.

King, A. 1994. A synergistic analysis for sharing and groundness which traces linearity. Sannella, D., Ed. In *Programming Languages and Systems – ESOP '94, 5th European Symposium on Programming*, April 11-13, Springer, Edinburg, U.K, Berlin Heidelberg, 788, 363–378, *Lecture Notes in Computer Science-Proceedings 1994*.

King, A. and Longley, M.1995. *Abstract matching can improve on abstract unification.* Canterbury, UK, University of Kent, Computing Laboratory. http://www.cs.ukc.ac.uk/pubs/1995/64. Last accessed Oct 10, 2022 Technical Report 4-95∗.

Langen, A. 1990. Static analysis for independent and-parallelism in logic programs. *PhD thesis*, University of Southern California. Los Angeles, California.

Le Charlier, B. and Van Hentenryck, P. 1994. Experimental evaluation of a generic abstract interpretation algorithm for PROLOG. *ACM Transactions on Programming Languages and Systems* 16, 1, 35–101.

Méndez-Lojo, M., Navas, J. and Hermenegildo, M. V. 2007. A flexible, (C)LP-based approach to the analysis of object-oriented programs, In *Logic Based Program Synthesis and Transformation 17th International Workshop, LOPSTR. 2007, Kongens Lyngby*, King, A., 4915, Denmark, Berlin Heidelberg, Springer, 154–168. Lecture Notes in Computer Science. Revised Selected Papers 2008.

Muthukumar, K. and Hermenegildo, M. V. 1992. Compile-time derivation of variable dependency using abstract interpretation. *The Journal of Logic Programming* 13, 2-3, 315–347.

Muthukumar, K. and Hermenegildo, M. V. Combined determination of sharing and freeness of program variables through abstract interpretation. Furukawa, K., Ed. In *Logic Programming, Proceedings of the Eighth International Conference 1991, Logic Programming*, The MIT Press, Cambridge, MA, USA, 49–63.

Peralta, J. C., Gallagher, J. P. and Sağlam, H. 1998. Analysis of imperative programs through analysis of constraint logic programs. Levi, G., Ed. In *Static Analysis. 5th International Symposium, SAS '98, Proceedings 1998*, September 14-16, Springer, Pisa, Italy, Berlin Heidelberg, 1503. 246–261. Lecture Notes in Computer Science.

Søndergaard, H. 1986, An application of abstract interpretation of logic programs: Occur check reduction. Robinet, B. and Wilhelm, R., Eds. In *ESOP 86, European Symposium on Programming, Saarbrücken, Federal Republic of Germany, Proceedings 1986*, March 17-19, Springer, Berlin Heidelberg, 213, 327–338, Lecture Notes in Computer Science.

Spoto, F., Mesnard, F. and Payet, E. 2010. A termination analyzer for Java bytecode based on path-length. *ACM Transactions on Programming Languages and Systems* 32, 3, 8:1–8:70.

https://doi.org/10.1017/S1471068424000152 Published online by Cambridge University Press