Abstract

Analyses for possible variable sharing and definite freeness are important both in the automatic parallelisation and in the optimisation of sequential logic programs. In this paper, a new efficient approach to analysis is described which can infer sharing and freeness information to an unusually high degree of accuracy. The analysis exploits a confluence property of the unification algorithm to split the analysis into two distinct phases. The two phase analysis improves efficiency by enabling each phase of the analysis to manipulate relatively simple data-structures. The precision follows from the combination of domains. The analysis propagates groundness with the accuracy of sharing groups and yet infers sharing and freeness to a precision which exceeds that of a normal freeness analysis. High precision compoundness information can be derived too. The usefulness of the analysis is demonstrated with worked examples. Correctness is formally proven.

1 Introduction

Abstract interpretation for sharing and freeness are important topics of logic programming. Sharing (or aliasing) analysis conventionally infers which program variables are definitely grounded and which variables can possibly be bound to terms containing a common variable. Freeness analysis usually infers which program variables are free, that is, which variables can never be bound to non-variable terms. Compoundness analysis [21, 9] is the dual of freeness analysis in that it detects which variables are guaranteed to be bound to non-variable (compound) terms. Compoundness analysis, as applied in [9], additionally traces the principal functor of variable bindings. Compoundness information can aid indexing. Applications of sharing and freeness information are numerous and include: the sound removal of the occurrence check [25]; optimisation of backtracking [5]; the specialisation of unification [27]; and the identification [28, 13] and efficient exploitation [23, 14, 24] of independent and-parallelism. Early proposals for sharing, freeness and compoundness analyses include [29, 12, 20], [23] and [21].

This paper is concerned with a semantic basis for sharing, freeness and compoundness analysis, and in particular, the justification of a high precision abstract unification algorithm. Following the approach of abstract interpretation [10], an abstract unification algorithm (the abstract operation) is constructed by mimicking the substitutions (the concrete data) which arise in a standard unification algorithm [17] (the concrete operation) with finite sharing, freeness and compoundness abstractions (the abstract data).
The accuracy of the analysis depends, in part, on the substitution properties that the sharing abstractions capture. The popular sharing and freeness domain $\text{Share} \times \text{Free}$ [23], for instance, captures possible sharing and definite groundness in its $\text{Share}$ component; and definite freeness in its $\text{Free}$ component. The structure of $\text{Share}$ is particularly rich, implicitly encoding covering information [9]. Covering, in short, permits groundness to interact nicely with sharing to remove redundant aliasing. For finiteness, $\text{Share} \times \text{Free}$ is parametrised by a finite set of program variables, $\text{Pvar}$, which typically equate to the variables of a clause. To be precise, $\text{Share}_{\text{Pvar}} = \psi(\psi(\text{Pvar}))$ and $\text{Free}_{\text{Pvar}} = \psi(\text{Pvar})$. If $\text{Pvar} = \{x, y, z\}$, for instance, the substitution $\phi = \{x \mapsto f(a, y, b), z \mapsto g(y)\}$ would be encoded by the pair $(\phi^S, \phi^F)$ where $\phi^S = \{(x, y, z), 0\}$ and $\phi^F = \{g\}$. The pair indicates that a variable occurs through $x$, $y$ and $z$, that is, they share; and that $y$ is free.

Tracking freeness often brings a two-fold win: first, it enlarges the class of ensuing optimisations [28, 13]; second, it improves the groundness and sharing [23]. Groundness and sharing is refined since freeness relates to the structural or type properties of a substitution. Precision is improved as a result of the synergy between sharing and type analysis. By keeping track of type information, it is possible to infer more accurate sharing information. Conversely, more accurate type information can be deduced if sharing is traced. Specifically, by tracking freeness [23, 9, 26] (or alternatively a type property called linearity [25, 7, 15]), a sharing analysis does not always have to assume that aliasing is transitive [7]. If variables can be inferred to be free, worst case aliasing need not be assumed in an analysis.

$\text{Share}_{\text{Pvar}} \times \text{Free}_{\text{Pvar}}$, however, only captures shallow type information: it traces the freeness of terms to which a variable can be bound but not the freeness of sub-terms. The usefulness of tracing sharing and freeness to the level of sub-terms has been reported before [16] but the analysis proposed in [16] is difficult to implement efficiently. This paper remedies this deficiency by augmenting the domain $\text{Share}_{\text{Pvar}} \times \text{Free}_{\text{Pvar}}$ with a simple but powerful type component $\text{Sub}_{\text{Pvar}}^T$, and by adopting a new, modular approach to analysis. The domain $\text{Sub}_{\text{Pvar}}^T$ consists of a set of canonical substitutions on $\text{Pvar}$ which encode structure. The composite domain is a subset of $\text{Sub}_{\text{Pvar}}^T \times \text{Share}_{\text{Pvar}} \times \text{Free}_{\text{Pvar}}$, denoted $\text{Type}_{\text{Pvar}}$, and might typically represent $\phi$ by

$$\phi^C = \{x \mapsto f(a, x^2, b), y \mapsto y^\lambda, z \mapsto g(z^1)\},$$

$(\phi^C, \phi^S, \phi^F)$ where $\phi^S = \{(x^2, y^\lambda, z^1), 0\}$,

$\phi^F = \{x^2, y^1\}$

The triple indicates sharing between the second argument of term bound to $x$ ($x^2$), $y$ ($y^\lambda$), and the first argument of the term bound to $z$ ($z^1$). The triple also records the freeness of both $x^2$ and $y^\lambda$. It thus represents sharing and freeness to the precision of sub-terms. In addition, it captures useful compoundness information too, for instance, that $x$ is bound to a term with a principal functor $f$ and an arity of 3.

In contrast to other approaches [16], high precision does not come at the expense of gross inefficiency. The analysis exploits a confluence property of the unification algorithm (that all unifiers are equal up to renaming [17]) to split the analysis into two distinct phases. In the first phase compoundness information is tracked. In the second phase sharing and freeness is traced. The compoundness phase only operates on the compoundness component of the domain. Similarly, the sharing and freeness phase only operates on the sharing and freeness component of the domain. Since each phase of the analysis need only manipulate its own (relatively simple) data-structure, efficiency is maintained without sacrificing precision. The modularity also leads to a well-structured proof of correctness.
The exposition is structured as follows. Section 2 describes the notation and preliminary definitions which will be used throughout. In section 3, the focus is on abstracting data. An abstraction for substitutions is constructed which expressively captures sharing, freeness and compoundness properties of substitutions. In section 4, the emphasis changes to abstracting operations. Abstract analogs for renaming, unification, composition and restriction are defined in terms of an abstract unify operator [14]. An abstract unification algorithm is defined which, in turn, describes an abstract analog of unify. (Once an abstract unify operator is specified and proved safe, a complete and correct abstract interpreter is practically defined by virtue of existing abstract interpretation frameworks [2, 18, 24].) Correctness is also proved. For reasons of brevity and continuity, however, the proofs are relegated to an appendix, section 7. Finally, sections 5 and 6 present the related work and the concluding discussion.

2 Notation and preliminaries

To introduce the analysis some notation and preliminary definitions are required. The reader is assumed to be familiar with the standard constructs used in logic programming [19] such as a universe of (possibly sup- and sub-scripted) variables (u, v ∈ Uvar; the set of terms (t ∈)Term formed from Uvar and the set of functors (f, g, h ∈ Func (of the first-order language underlying the program), and the set of program atoms Atom. Func is considered to include the set of constants Const. It is sometimes convenient to abbreviate f(t1, ..., tn) to f(t).[1]

Let Pvar denote a finite set of program variables – the variables that are in the text of the program; and let var(o) denote the set of variables in a syntactic object o. Also suppose that the set of finite sequences of positive integers is denoted by S = {1, 2, ...}*. S is considered to include the empty sequence λ. If denotes concatenation of sequences, the application of a term t to the sequence s, t[s], can be defined by the partial mapping t[λ] = t and f(t1, ..., tn)[i·s] = t[i][s]. Hence, f(u, g(v))[2-1·λ] = v, f(u, g(v))[1·λ] = u, and f(u, g(v))[λ] = f(u, g(v)).

2.1 Substitutions

A substitution φ is a total mapping φ : Uvar → Term such that its domain dom(φ) = {u ∈ Uvar | φ(u) ≠ u} is finite. The application of a substitution φ to a variable u is denoted by φ(u). Thus the codomain is give by cod(φ) = ∪u∈dom(φ)var(φ(u)). A substitution φ is sometimes represented as a finite set of variable and term pairs {u ↦ φ(u) | u ∈ dom(φ)}. The identity mapping on Uvar is called the empty substitution and is denoted by ε. Substitutions, sets of substitutions, and the set of substitutions on Uvar are denoted by lower-case Greek letters, upper-case Greek letters, and Sub. Substitutions are extended in the usual way from variables to functions, from functions to terms, and from terms to atoms. The restriction of a substitution φ to a set of variables U ⊆ Uvar and the composition of two substitutions φ and ϕ, are denoted by φ|U and φ ◦ ϕ respectively, and defined so that (φ ◦ ϕ)(u) = φ(ϕ(u)). Restriction lifts to sets of substitutions by: Φ|U = {φ|U | φ ∈ Φ}. The preorder Sub (⊆), is more general than ϕ, is defined by: φ ⊆ ϕ if and only if there exists a substitution ψ ∈ Sub such that ϕ = ψ ◦ ϕ. The preorder induces an equivalence relation ≈ on Sub, that is: φ ≈ ϕ if and only if φ ⊆ ϕ and ϕ ⊆ φ.
A useful related preorder $Sub (\subseteq P_{var})$ is defined by: $\phi \subseteq P_{var} \varphi$ if and only if there exists a substitution $\psi \in Sub$ such that $\varphi \uparrow P_{var} = (\psi \circ \phi) \uparrow P_{var}$.

### 2.2 Equations and most general unifiers

An equation is an equality constraint of the form $a = b$ where $a$ and $b$ are terms or atoms. Let $(e \in) Eqn$ denote the set of finite sets of equations. The equation set $\{e\} \cup E$, following [7], is abbreviated by $e : E$. There is a natural mapping from substitutions to equations, that is, $eqn(\phi) = \{u = t_u \mid u \mapsto t_u \in \phi\}$. Thus, when unambiguous, substitutions will be expressed as equations. The set of most general unifiers of $E$, $mgu(E)$, is defined operationally [1-4] in terms of a predicate $mgu$. The predicate $mgu(E, \phi)$ which is true if $\phi$ is a most general unifier of $E$.

**Definition 2.1** ($mgu$) The set of most general unifiers $mgu(E) \in \phi(Sub)$ is defined by:

$$mgu(E) = \{\phi \mid mgu(E, \phi)\}$$

where

$$mgu(\emptyset, e) = \emptyset$$

$$mgu(v = v : E, \zeta) = \{v \mapsto e\} : E, \zeta$$

$$mgu(t = v : E, \zeta) = \{v \mapsto t\} : E, \zeta$$

$$mgu(v = t : E, \zeta \circ \eta) = \{v \mapsto t\} : E, \zeta \circ \eta \cup \{v \mapsto \eta\}$$

By induction it follows that $dom(\phi) \cap cod(\phi) = \emptyset$ if $\phi \in mgu(E)$, or put another way, that the most general unifiers are idempotent [17].

Following [14], the semantics of a logic program is formulated in terms of a single $unify$ operator. To construct $unify$, and specifically to rename apart program variables, an invertible substitution [17], $\Upsilon$, is introduced. It is convenient to let $R_{var}$ denote a universe of renaming variables distinct from $U_{var}, U_{var} \cap R_{var} = \emptyset$, and suppose that $\Upsilon : U_{var} \rightarrow R_{var}$. $\Upsilon$ is assumed to consistently rename super-scripted variables, that is, $\Upsilon^*(u*) = \Upsilon(u^*)$.

**Definition 2.2** ($unify$) The partial mapping $unify : Atom \times Sub \times Atom \times Sub \rightarrow Sub$ is defined by:

$$unify(a, \phi, b, \psi) = (\varphi \circ \phi) \uparrow P_{var} \text{ where } \varphi \in mgu(\{\phi(a) = \Upsilon(\psi(b))\})$$

To approximate the $unify$ operation it is convenient to introduce a collecting semantics, concerned with sets of substitutions, to record the substitutions that occur at various program points. In the collecting semantics interpretation, $unify$ is extended to $unify^f$, which manipulates (possibly infinite) sets of substitutions.

**Definition 2.3** ($unify^f$) The mapping $unify^f : Atom \times \phi(Sub) \times Atom \times \phi(Sub) \rightarrow \phi(Sub)$ is defined by:

$$unify^f(a, \Phi, b, \Psi) = \{\theta \mid \phi \in \Phi \land \psi \in \Psi \land \theta = unify(a, \phi, b, \psi)\}$$
2.3 Linearity

Linearity relates to the number of times a variable occurs in a term [25, 7, 15]. A term is linear if it definitely does not contain multiple occurrences of a variable; otherwise it is non-linear. The significance of linearity is that the unification of linear terms only yields restricted forms of aliasing. This is exploited (in proof 7.10) to simplify the proof of correctness. To be more precise about linearity, it is necessary to introduce the variable multiplicity of a term $t$, denoted $\chi(t)$.

**Definition 2.4 (variable multiplicity, $\chi$ [7])** The variable multiplicity operator $\chi : \text{Term} \to \{0, 1, 2\}$ is defined by:

$$
\chi(t) = \max\{\chi_u(t) \mid u \in \text{var}\} \quad \text{where} \quad \chi_u(t) = \begin{cases} 0 & \text{if } u \text{ does not occur in } t \\ 1 & \text{if } u \text{ occurs only once in } t \\ 2 & \text{if } u \text{ occurs many times in } t \end{cases}
$$

Lemma 2.1 states one restriction on a most general unifier which follow from unification with a linear term.

**Lemma 2.1** $\chi(b) \neq 2 \land \text{var}(a) \cap \text{var}(b) = \emptyset \land \phi \in \text{mg}(\{a = b\}) \Rightarrow$

1. $\forall u, u' \in \text{var}. u \neq u' \land \text{var}(\phi(u)) \cap \text{var}(\phi(u')) \neq \emptyset \Rightarrow u \not\in \text{var}(a) \lor u' \not\in \text{var}(a)$.

Lemma 2.1 represents one case of a three part result which is formally established in [15]. The lemma differs from the corresponding lemma in [7] (lemma 2.2) because lemma 2.1 requires that $a$ and $b$ do not share variables. This is essentially a work-around for a subtle mistake in lemma 2.2 [11].

3 Abstracting substitutions

Abstract interpretation clarifies how data is represented in the abstract by requiring the relationship between the data and the abstract data to be made explicit. Because of the compositional nature of $\text{Type}_{\text{Pvar}}$ this relationship is detailed in three steps. In the first step, section 3.1, the focus is on the $\text{Sub}_{\text{Pvar}}$ component and its abstraction and concretisation mappings. Section 3.2, the second step, is concerned with the $\text{Share}_{\text{Pvar}} \times \text{Free}_{\text{Pvar}}$ component. Finally the third step, section 3.3, explains how the $\text{Sub}_{\text{Pvar}}^C$, $\text{Share}_{\text{Pvar}}$ and $\text{Free}_{\text{Pvar}}$ components fit together to produce $\text{Type}_{\text{Pvar}}$.

3.1 Abstracting substitutions with $\text{Sub}_{\text{Pvar}}^C$

The $\phi$ example of section 1 captures the compoundness of $\phi$ in $\phi^C = \{x \mapsto f(a, x^2, b), y \mapsto y^3, z \mapsto g(z^3)\}$ and the sharing and freeness of the variables $\text{cod}(\phi^C) = \{x^2, y^3, z^3\}$ by $\phi^S$ and $\phi^F$. The intuition behind the compoundness abstraction $\phi^C$ is that it represents substitutions $\phi$ for which: $\phi^C$ is more general than any $\phi$. One advance of this approach is that it permits compoundness to be expressed in simple terms. One disadvantage, however, is that since $\text{Sub}$ is a pre-order (rather than a poset), the compoundness of $\phi$ could equally be represented by $\{x \mapsto f(a, u, b), y \mapsto v, z \mapsto g(w)\}$. Thus, in general, there is not a unique best substitution for representing the compoundness of $\phi$. 

5
3.1.1 On the domain $Sub^C_{\text{Pvar}}$

Representation problems can be straightforwardly avoided, however, by defining the compoundness domain to coincide with a poset of substitutions. One suitable and particularly simple poset can be constructed from substitutions with codomain variables super-scripted by integer sequences drawn from $S$.

To be more precise, if $Uvar^S$ denotes the variables of $Uvar$ which are super-scripted by $S$, then the substitutions of the poset are defined to map variables of $Uvar$ into a set of terms with super-scripted variables dubbed $Term^S$. $Term^S$ is formally defined by: $Term^S = \{ t \in Term | \text{var}(t) \subseteq Uvar^S \}$. In fact, to conform to a poset, the variables must be consistently super-scripted. The particular notion of consistency is formalised by the term set $Term^\lambda_u$.

**Definition 3.1** ($Term^\lambda_u$) $Term^\lambda_u$ is the least set defined by:

$$Term^\lambda_u = \{ u^\lambda \} \cup \left\{ f(t_1, \ldots, t_n) \in Term^\lambda_u \mid \begin{array}{l} f \in \text{Func} \land \hphantom{\text{dom}((\phi^C)) = \text{Pvar} \land \forall u \mapsto t_u \in \phi^C \cdot t_u \in Term^\lambda_u} \\
\quad t_1 \in Term^{\lambda_1}_u \land \ldots \land t_n \in Term^{\lambda_n}_u \end{array} \right\}$$

In $Term^\lambda_u$ variables are consistently super-scripted in the sense that the position $p$ of a variable in a term $t$ is given by its super-script, that is, $t[p] = u^{\lambda_p}$.

**Example 3.1** Returning to the example of section 1, observe that the terms $\phi^C(x), \phi^C(y)$ and $\phi^C(z)$ are consistently super-scripted in that

$$f(a, x^2, b) \in Term^\lambda_z, \quad y^\lambda \in Term^\lambda_y, \quad g(z^1) \in Term^\lambda_z$$

To flag the domain and codomain of a substitution, it is helpful to identify three classes of substitution. Specifically, let $\phi^D \in Sub^D$ denote substitutions for which $\phi^D : Uvar^S \rightarrow Term$; let $\phi^C \in Sub^C$ denote substitutions such that $\phi^C : Uvar \rightarrow Term^S$; and finally let $\phi^B \in Sub^B$ denote substitutions with $\phi^B : Uvar^S \rightarrow Term^S$. Informally, the $D, C$ and $B$ indicate whether or not the domain, codomain or both annotated with sequence information.

The domain $Sub^C_{\text{Pvar}}$ is finally fleshed out by restricting the variable term bindings $u \mapsto t_u$ of $Sub^C$ to a particular form where $u \in Pvar$ and $t_u \in Term^\lambda_u$.

**Definition 3.2** ($Sub^C_{\text{Pvar}}$) The domain $Sub^C_{\text{Pvar}}$ is defined by:

$$Sub^C_{\text{Pvar}} = \left\{ \phi^C \in Sub^C \mid \begin{array}{l} \text{dom}(\phi^C) = Pvar \land \hphantom{\text{dom}(\phi^C) = Pvar \land \forall u \mapsto t_u \in \phi^C \cdot t_u \in Term^\lambda_u} \hphantom{\forall u \mapsto t_u \in \phi^C \cdot t_u \in Term^\lambda_u} \\
\quad \forall u \mapsto t_u \in \phi^C \cdot t_u \in Term^\lambda_u \end{array} \right\}$$

The significance of the construction is that $Sub^C_{\text{Pvar}} (\sqsubseteq_{\text{Pvar}})$ is a poset rather than a pre-order. Moreover, when equipped with a top element $\text{fail}^C$, a lub $\sqcup^C$ (corresponding to unification) and a glb $\sqcap^C$ (corresponding to anti-unification [17]), $Sub^C_{\text{Pvar}} \cup \{ \text{fail}^C \} (\sqsubseteq_{\text{Pvar}})$ is a complete lattice.

3.1.2 On the mappings $\alpha^C_{\text{Pvar}}$ and $\gamma^C_{\text{Pvar}}$

The mappings $\alpha^C_{\text{Pvar}}$ and $\gamma^C_{\text{Pvar}}$ are introduced to explain how compoundness abstractions connect with substitutions.
Definition 3.3 (αC_Pvar and γC_Pvar) The abstraction and concretisation mappings αC_Pvar : ϕ(Sub) → S ubC_Pvar and γC_Pvar : S ubC_Pvar → ϕ(Sub) are defined by:

\[ αC_Pvar(ϕ) = \bigcup^C \{ φC ∈ S ubC_Pvar \mid φC ⊆ Pvar \land ϕ ∈ Φ \} \]

\[ γC_Pvar(φC) = \{ φ ∈ S ub \mid φC ⊆ Pvar \land φ \} \]

Notice that αC_Pvar(\{e\}) = eC where eC = \{u ↦ u^λ \mid u ∈ Pvar\} and more generally u ↦ t u ∈ αC_Pvar(Φ) for all u ∈ Pvar.

Example 3.2 If μ = \{x ↦ f(a, y, b), z ↦ g(y)\} and ν = \{x ↦ f(y, z, c)\} with Pvar = \{x, y, z\} then

\[ αC_Pvar(\{μ\}) = \{x ↦ f(a, x^2, b), y ↦ y^λ, z ↦ g(z^1)\} \]

\[ αC_Pvar(\{ν\}) = \{x ↦ f(x^1, x^2, c), y ↦ y^λ, z ↦ z^λ\} \]

\[ αC_Pvar(\{μ, ν\}) = \{x ↦ f(x^1, x^2, x^3), y ↦ y^λ, z ↦ z^λ\} \]

3.1.3 On finiteness

For a given program Func is finite and therefore termination can be enforced by representing compoundness information to a predetermined depth bound k. The notion of depth is formalised by a mapping depth : Term → \{1, 2, \ldots\} defined by:

\[ depth(t) = \begin{cases} 
1 & \text{if } t ∈ U var \lor t ∈ Const \\
1 + \max(\{depth(t_i)\}_{i=1}^n) & \text{if } t ∈ Func \land t = f(t_1, \ldots, t_n)
\end{cases} \]

A natural depth-k widening can be defined by extending the depth mapping to substitutions:

\[ depth(t) = \max(\{depth(t_u) \mid u ↦ t_u ∈ ϕ\}) \]

Definition 3.4 (∇k C) The widening ∇k C : S ubC_Pvar → S ubC_Pvar is defined by:

\[ ∇k C(φC) = \bigcup^C \{ φC’ ∈ S ubC_Pvar \mid φC’ ⊆ Pvar \land depth(φC’) ≤ k \} \]

Observe that γC_Pvar(φC) ⊆ γC_Pvar(∇k C(φC)) as required for correctness.

Example 3.3 Adopting the μ of example 3.2

\[ ∇1 C(αC_Pvar(\{μ\})) = \{x ↦ x^λ, y ↦ y^λ, z ↦ z^λ\} \]

\[ ∇2 C(αC_Pvar(\{μ\})) = \{x ↦ f(a, x^2, b), y ↦ y^λ, z ↦ g(z^1)\} \]

3.2 Abstracting substitutions with SharePvar × FreePvar

To keep the paper self-contained the SharePvar × FreePvar domain and its mappings will be briefly reviewed.
3.2.1 On the domain $Share_{Pvar} \times Free_{Pvar}$

$Share_{Pvar}$ is formulated in terms of sharing groups [14, 24] which record which program variables potentially share variables. A sharing group is a (possibly empty) set of program variables. $Free_{Pvar}$, on the other hand, represents the free program variables as a set.

**Definition 3.5** ($Share_{Pvar}$ and $Free_{Pvar}$) The domains $Share_{Pvar}$ and $Free_{Pvar}$ are defined by:

$$Share_{Pvar} = \emptyset(\emptyset(Pvar)), \quad Free_{Pvar} = \emptyset(Pvar)$$

The intuition is that a sharing group records which program variables are bound to terms that share a variable. $Share_{Pvar} \times Free_{Pvar}$ is finite since Pvar is finite.

3.2.2 On the mappings $\alpha_{Pvar}^{SF}$ and $\gamma_{Pvar}^{SF}$

In the spirit of [24], the abstraction and concretisation mappings are constructed by lifting two mappings, $sh_{Pvar}$ and $fr_{Pvar}$, to sets of substitutions. The mappings $sh_{Pvar}$ and $fr_{Pvar}$ detail how a single substitution is abstracted.

**Definition 3.6** ($sh_{Pvar}$ and $fr_{Pvar}$) The abstraction mappings $sh_{Pvar} : Sub \rightarrow Share_{Pvar}$ and $fr_{Pvar} : Sub \rightarrow Free_{Pvar}$ are defined by:

$$sh_{Pvar}(\phi) = \{occ_{Pvar}(u, \phi) | u \in Uvar\}, \quad occ_{Pvar}(u, \phi) = \{v \in Pvar | u \in var(\phi(v))\}$$

$$fr_{Pvar}(\phi) = \{v \in Pvar \mid var(\phi(v)) \in Uvar\}$$

**Example 3.4** Using the $\mu$, $\nu$ and Pvar = $\{x, y, z\}$ of example 3.2

$$sh_{Pvar}(\mu) = \{occ_{Pvar}(y, \mu), \emptyset\} = \{\{x, y, z\}, \emptyset\}$$

$$sh_{Pvar}(\nu) = \{occ_{Pvar}(y, \nu), occ_{Pvar}(z, \nu), \emptyset\} = \{\{x, y\}, \{x, z\}, \emptyset\}$$

$$fr_{Pvar}(\mu) = \{y\}, \quad fr_{Pvar}(\nu) = \{y, z\}$$

The abstraction $sh_{Pvar}$ is analogous to the abstraction $\mathcal{A}$ used in [24]. Observe that for $\phi \approx \varphi$, $sh_{Pvar}(\phi) = sh_{Pvar}(\varphi)$ and $fr_{Pvar}(\phi) = fr_{Pvar}(\varphi)$. The mapping $\alpha_{Pvar}^{SF}$ and $\gamma_{Pvar}^{SF}$ follow directly from $sh_{Pvar}$ and $fr_{Pvar}$.

**Definition 3.7** ($\alpha_{Pvar}^{SF}$ and $\gamma_{Pvar}^{SF}$) The abstraction and concretisation mappings $\alpha_{Pvar}^{SF} : \emptyset(Sub) \rightarrow Share_{Pvar} \times Free_{Pvar}$ and $\gamma_{Pvar}^{SF} : Share_{Pvar} \times Free_{Pvar} \rightarrow \emptyset(Sub)$ are defined by:

$$\alpha_{Pvar}^{SF}(\Phi) = (\alpha_{Pvar}^{S}(\Phi), \alpha_{Pvar}^{F}(\Phi))$$

$$\gamma_{Pvar}^{SF}(\phi^{SF}) = \gamma_{Pvar}^{S}(\phi^{S}) \cap \gamma_{Pvar}^{F}(\phi^{F})$$

where

$$\alpha_{Pvar}^{S}(\Phi) = \cup_{\phi \subseteq \Phi} sh_{Pvar}(\phi), \quad \gamma_{Pvar}^{S}(\phi^{S}) = \{\phi \mid sh_{Pvar}(\phi) \subseteq \phi^{S}\}$$

$$\alpha_{Pvar}^{F}(\Phi) = \cap_{\phi \subseteq \Phi} fr_{Pvar}(\phi), \quad \gamma_{Pvar}^{F}(\phi^{F}) = \{\phi \mid \phi^{F} \subseteq fr_{Pvar}(\phi)\}$$

Note that $\alpha_{Pvar}^{S}(\emptyset) = \emptyset$ whereas $\alpha_{Pvar}^{S}(\Phi) = \{\emptyset\}$ if $\Phi$ is a set of substitutions which all ground Pvar. This distinction is preserved in both $Share_{Pvar} \times Free_{Pvar}$ and $Type_{Pvar}$.

**Example 3.5** Continuing with example 3.4

$$\alpha_{Pvar}^{SF}(\{\mu, \nu\}) = \{\{x, y, z\}, \{x, y\}, \{x, z\}, \emptyset\}, \{y\}$$
3.3 Abstracting substitutions with TypePvar

The domain TypePvar is a strict subset of SubPvar \times SharePvar \times FreePvar, crafted so that TypePvar is a complete lattice. The construction proceeds by first specifying the structure of TypePvar in section 3.3.1. Second, in section 3.3.2, the concretisation mapping \( \gamma_{Pvar}^{\mathcal{CSF}} \) is introduced to formalise the relationship between TypePvar and \( \varphi(Sub) \). Third, in section 3.3.3, the lattice \( \varphi(Sub) (\subseteq) \) is used to induce a poset TypePvar (\( \subseteq^{\mathcal{CSF}} \)) which, when equipped with a lub \( \cup^{\mathcal{CSF}} \) and a glb \( \cap^{\mathcal{CSF}} \), is elevated to a complete lattice. Fourth, in section 3.3.4, the glb \( \cup^{\mathcal{CSF}} \) and the \( \gamma_{Pvar}^{\mathcal{CSF}} \) are used to finally formulate the abstraction mapping \( \sigma_{Pvar}^{\mathcal{CSF}} \), and thus complete the construction of a Galois connection [10].

3.3.1 On the domain TypePvar

The domain TypePvar is formulated so that \( \gamma_{Pvar}^{\mathcal{CSF}} \) is injective. An injective concretisation mapping is useful since it can be used to straightforwardly induce a poset (rather an a pre-order) on TypePvar from Sub (\( \subseteq \)). In short, the domain is constructed so that different domain elements (in the abstract) represent different sets of substitutions (in the concrete).

**Definition 3.8 (TypePvar)** The abstract domain, TypePvar, is defined by:

\[
\text{TypePvar} = \{ \langle \text{fail}, \emptyset, \emptyset \rangle \cup \langle \phi^C, \mathcal{U}, \mathcal{U} \rangle \mid \phi^C \in \text{SubPvar}^\wedge \\
\mathcal{U} \in \text{Share}_c^{\text{cod}(\phi^C)} \wedge \mathcal{U} \neq \emptyset \wedge \\
\mathcal{U} \in \text{Free}_c^{\text{cod}(\phi^C)} \}
\]

3.3.2 On the mapping \( \gamma_{Pvar}^{\mathcal{CSF}} \)

The relationship between TypePvar and \( \varphi(Sub) \) is made explicit by the mapping \( \gamma_{Pvar}^{\mathcal{CSF}} \).

**Definition 3.9 (\( \gamma_{Pvar}^{\mathcal{CSF}} \))** The concretisation mapping \( \gamma_{Pvar}^{\mathcal{CSF}} : \text{TypePvar} \rightarrow \varphi(Sub) \) is defined by:

\[
\gamma_{Pvar}^{\mathcal{CSF}}(\langle \phi^C, \phi^S, \phi^F \rangle) = \\
\left\{ \phi \mid \phi \uparrow Pvar = \phi^D \circ \phi^C \uparrow Pvar \wedge \\
\text{sh}_{\text{cod}(\phi^C)}(\phi^D) \subseteq \phi^S \wedge \phi^F \subseteq \text{fr}_{\text{cod}(\phi^C)}(\phi^D) \right\}
\]

It is assumed that \( \gamma_{Pvar}^{\mathcal{CSF}}(\langle \text{fail}, \emptyset, \emptyset \rangle) = \emptyset \). Notice how the definitions of \( \sigma_{Pvar}^{\mathcal{CSF}} \) requires the \( \text{sh}_{Pvar} \) and \( \text{fr}_{Pvar} \) mappings to be parametrised by \( Pvar \). The parameter is required because the Share and Free components of TypePvar record the sharing and freeness of the variables of \( \text{cod}(\phi^F) \) which is not fixed. Note also that for \( \phi \approx \varphi \), \( \phi \in \gamma_{Pvar}^{\mathcal{CSF}}(\phi^{\mathcal{CSF}}) \) if and only if \( \varphi \in \gamma_{Pvar}^{\mathcal{CSF}}(\phi^{\mathcal{CSF}}) \). This follows because \( \text{sh}_{Pvar}(\emptyset) = \text{sh}_{Pvar}(\emptyset) \) and \( \text{fr}_{Pvar}(\emptyset) = \text{fr}_{Pvar}(\emptyset) \) if \( \emptyset \approx \emptyset \).

**Example 3.6** To illustrate the expressiveness of TypePvar, consider the properties of an arbitrary substitution \( \phi \) taken from \( \gamma_{Pvar}^{\mathcal{CSF}}(\phi^{\mathcal{CSF}}) \) where

\[
\phi^{\mathcal{CSF}} = \{ \{ x \mapsto f(x^1, x^2, x^3), y \mapsto y^\lambda, z \mapsto z^\lambda \}, \\
\{ x^1, y^\lambda \}, \{ x^2, y^\lambda, z^\lambda \}, \{ x^2, z^\lambda \}, \emptyset, \{ x^2, y^\lambda \} \}
\]
sharing: The first argument of \( \phi(x) \) and \( \phi(z) \) do not share. This follows because \( x^1 \) and \( z^3 \) do not occur in a sharing group of \( \phi^S \). Likewise, \( x \) never has any internal aliasing under \( \phi \). Put another way, \( x \) is linear [25, 7, 15], which in this case means that the first, second and third argument of \( x \) are independent.

groundness: The third argument of \( \phi(x) \) is ground. This follows since the variables of \( \text{cod}(\phi^F) \) which \( \phi \) ground, do not appear in \( \phi^S \).

freeness: The variable \( y \) is free under \( \phi \) and the second argument of \( \phi(x) \) is also free. This follows immediately from \( \phi^F \).

compoundness: Compoundness is captured in that \( \phi^C \) shows that \( \phi(x) \) is compound with a principal functor of \( f \) and an arity of 3.

covering: Covering is also implicitly captured by virtue of \( \phi^S \). For instance, \( x \) covers both \( y \) or more exactly, the first and second arguments of \( x \) cover \( y \). Thus grounding \( x \) grounds \( y \); or more precisely, grounding the first and second arguments of \( x \) grounds \( y \).

3.3.3 On the domain Type_{Pvar} (continued)

A natural ordering \( \text{Type}_{Pvar} (\sqsubseteq^{CSF}) \) can be induced from \( \text{Sub} (\subseteq) \) by \( \gamma_{Pvar}^{CSF} \).

Definition 3.10 The preorder \( \text{Type}_{Pvar} (\sqsubseteq^{CSF}) \) is defined by: \( \phi^{CSF} \sqsubseteq^{CSF} \varphi^{CSF} \) if and only if \( \gamma_{Pvar}^{CSF} (\phi^{CSF}) \subseteq \gamma_{Pvar}^{CSF} (\varphi^{CSF}) \).

Note that \( \gamma_{Pvar}^{CSF} \) is monotonically increasing by definition. \( \text{Type}_{Pvar} \) is carefully engineered so that the preorder \( \text{Type}_{Pvar} (\sqsubseteq^{CSF}) \) is also a poset. This is formally stated as lemma 3.1 and established by proof 3.1.

Lemma 3.1 \( \text{Type}_{Pvar} (\sqsubseteq^{CSF}) \) is a poset.

The poset has a top element \( \top^{CSF} = (e, \phi(\text{cod}(e))), 0) \) and a bottom element \( \bot^{CSF} = (\text{fail}, 0, 0) \) since \( \gamma_{Pvar}^{CSF} (\top^{CSF}) = \text{Sub} \) and \( \gamma_{Pvar}^{CSF} (\bot^{CSF}) = 0 \). The bottom element is meaningful and, in fact, represents goal failure. Specifically, in a top-down abstract interpretation framework [2, 18, 24], it is possible for a goal to fail if it is called with some abstract substitutions. Returning \( \bot^{CSF} \) for the success abstract substitution indicates that the goal can never be satisfied under any calling substitution which the calling abstract substitution abstracts.

For the purposes of abstract interpretation [10], \( \text{Type}_{Pvar} (\sqsubseteq^{CSF}) \) is required to be a complete lattice, preferably equipped with a lub \( \sqcup^{CSF} \) which is straightforward to compute. To succinctly define \( \sqcup^{CSF} \) (and later a widening \( \nu^{CSF} \)) it is convenient to introduce an auxiliary mapping. The mapping is formulated as a substitution. More exactly, if \( \phi^B = \{ x^1 \mapsto a, x^3 \mapsto b, z^3 \mapsto g(z^2) \} \), it is helpful to define \( \psi^B = \{ z^1 \mapsto z^3 \} \). Informally, \( \psi^B \) is constructed to map the codomain variables of \( \phi^B \) to the domain variables of \( \phi^S \). More formally, \( \psi^B = \{ u_t \mapsto u^t | u^t \mapsto t_{u^t} \in \psi^B \land u_t \in \text{var}(t_{u^t}) \} \). A sufficient condition for \( \psi^B \) to be well-defined is that \( t_{u^t} \in \text{Term}^a_u \) for all \( u_t \mapsto t_{u^t} \in \psi^B \). This, in fact, will always be guaranteed by the context.
Definition 3.11 ($\sqcup^{CSF}$) The mapping $\sqcup^{CSF} : \psi(Type_{Pvar}) \to Type_{Pvar}$ is defined by:

$$\sqcup^{CSF}(\Phi^{CSF}) = \langle \varphi^C, \varphi^S, \varphi^F \rangle$$

where

$$\varphi^C = \cap \ell (\{ \phi^C | \phi^{CSF} \in \Phi^{CSF} \})$$

$$\varphi^S = \cup (\{ \phi^B(\phi^S) | \phi^{CSF} \in \Phi^{CSF} \land \phi^B \circ \varphi^C \uparrow Pvar = \phi^C \})$$

$$\varphi^F = \cap (\{ \phi^F \setminus \text{cod}(\phi^B) | \phi^{CSF} \in \Phi^{CSF} \land \phi^B \circ \varphi^C \uparrow Pvar = \phi^C \})$$

It is assumed that $\cap \ell (\emptyset) = f\!a\!i\!l$ so that $\sqcup^{CSF}(\emptyset) = \perp^{CSF}$. Computationally, the lub $\sqcup^{CSF}$ inherits much of its simplicity from the pair-wise union and intersection lub of $Share_{Pvar} \times Free_{Pvar}$. The main novelty is that anti-unification [17] is required to compute the glb for the $Sub_{Pvar}^C$ component.

Lemma 3.2 $Type_{Pvar}(\sqcap^{CSF})$ has a lub $\sqcup^{CSF}$.

Example 3.7 To illustrate the calculation of $\sqcup^{CSF}$ consider $\sqcup^{CSF}(\{\mu^{CSF}, \nu^{CSF}\})$ where $Pvar = \{x, y, z\}$ and

$$\mu^{CSF} = \langle \{x \mapsto f(a, x^2, b), y \mapsto y^\lambda, z \mapsto g(z^1)\}, \{\{x^2, y^\lambda, z^1\}, \emptyset, \{x^2, y^\lambda, z^1\}\}$$

$$\nu^{CSF} = \langle \{x \mapsto f(x^1, x^2, c), y \mapsto y^\lambda, z \mapsto z^\lambda\}, \{\{x^1, y^\lambda\}, \{x^2, z^\lambda\}, \emptyset, \{x^1, x^2, y^\lambda, z^\lambda\}\}$$

For conciseness put $\varphi^{CSF} = \sqcup^{CSF}(\{\mu^{CSF}, \nu^{CSF}\})$ and thus

$$\varphi^C = \mu^C \cap \nu^C = \{x \mapsto f(x^1, x^2, x^3), y \mapsto y^\lambda, z \mapsto z^\lambda\}$$

$$\varphi^S = \mu^S(\mu^S) \cup \nu^B(\nu^S) = \{\{x^1, y^\lambda\}, \{x^2, y^\lambda, z^\lambda\}, \{x^2, z^\lambda\}, \emptyset\}$$

$$\varphi^F = (\mu^F \setminus \text{cod}(\mu^B)) \cap (\nu^F \setminus \text{cod}(\nu^B)) = \{x^2, y^\lambda\}$$

since

$$\mu^B = \{x^1 \mapsto a, x^3 \mapsto b, z^\lambda \mapsto g(z^1)\}, \quad \nu^B = \{x^3 \mapsto c\}$$

$$\mu^B = \{z^1 \mapsto z^\lambda\}, \quad \nu^B = \epsilon$$

with $\mu^B \circ \varphi^C \uparrow Pvar = \mu^C$ and $\nu^B \circ \varphi^C \uparrow Pvar = \nu^C$.

The glb $\cap^{CSF}$ can be defined (in the standard way [1]) in terms of the lub $\sqcup^{CSF}$. To be more precise, $\cap^{CSF}(\Phi^{CSF}) = \langle \{\varphi^{CSF} | \{\cap^{CSF} \neq \perp^{CSF} \} \in \Phi^{CSF} \} \cap \{\varphi^{CSF} \uparrow Pvar \neq \varphi^{CSF} \} \rangle$. Corollary 3.1 and proof 7.3 document that $Type_{Pvar}(\sqcap^{CSF})$ is a complete lattice as a consequence.

Corollary 3.1 $Type_{Pvar}(\sqcap^{CSF})$ is a complete lattice.

3.3.4 On the mapping $\alpha_{Pvar}^{CSF}$

Once the glb $\cap^{CSF}$ is defined, the concretisation mapping $\gamma_{Pvar}^{CSF}$ determines the abstraction mapping $\alpha_{Pvar}^{CSF}$. This is spelt out in definition 3.12. The reason for explicitly introducing $\alpha_{Pvar}^{CSF}$ (rather than leaving its definition implicit) is to aid the synthesis of a constructive version of $\alpha_{Pvar}^{CSF}$. 
Definition 3.12 \((α^{CSF}_P)\) The abstraction mapping \(α^{CSF}_P : ψ(Sub) → Type_P\) is defined by:
\[
α^{CSF}_P(Φ) = \bigcap^{CSF}(\{\phi^{CSF}_P | Φ ⊆ γ^{CSF}_P(\phi^{CSF}_P)\})
\]

Note that \(α^{CSF}_P\) is monotonically increasing since \(γ^{CSF}_P\) is monotonically decreasing. Furthermore \(Φ ⊆ γ^{CSF}_P(α^{CSF}_P(Φ))\) for all \(Φ ∈ ψ(Sub)\) and \(α^{CSF}_P(γ^{CSF}_P(\phi^{CSF}_P)) ⊆^{CSF} \phi^{CSF}_P\) for all \(\phi^{CSF}_P ∈ Type_P\).

Following [10], the relationship between \(Type_P\) and \(ψ(Sub)\) can thus be stated as a Galois connection. This gives lemma 3.3. The corresponding proof, numbered 7.5, is almost immediate.

Lemma 3.3 (Galois connection) \(α^{CSF}_P\) and \(γ^{CSF}_P\) form a Galois connection between \(Type_P\) and \(ψ(Sub)\) (⊆).

In fact, to be precise, \(α^{CSF}_P\) and \(γ^{CSF}_P\) form a Galois insertion [22] since \(γ^{CSF}_P\) is injective. In more pragmatic terms, an insertion means that the abstract domain does not contain any redundant elements.

Returning to the abstraction mapping, lemma 3.4 presents one constructive version of \(α^{CSF}_P\). Interestingly, the reformulation not only details how \(α^{CSF}_P\) can be computed, but it also emphasizes the connection between \(α^{CSF}_P\) and \(α^{CSF}_P\). Correctness is established in proof 7.4 and example 3.8 demonstrates one application of the new, constructive abstraction mapping.

Lemma 3.4
\[
α^{CSF}_P(Φ) = \bigcap^{CSF}(\{\phi^{CSF}_P | Φ ⊆ γ^{CSF}_P(\phi^{CSF}_P)\})
\]

where \(\phi^{CSF}_P = α^{CSF}_P(Φ)\).

Example 3.8 Building on example 3.2, let \(\phi^{CSF} = α^{CSF}_P(\{μ, ν\})\) and consider the calculation of \(α^{CSF}_P(Φ)\) where \(Φ = \{μ, ν\}\) and \(Pvar = \{x, y, z\}\). Thus
\[
α^{CSF}_P(Φ) = \langle \phi^{CSF}, sh_{\text{cod}(\phi^{CSF})}(μ^P) \cup sh_{\text{cod}(\phi^{CSF})}(ν^P), f_{\text{cod}(\phi^{CSF})}(μ^P) \cap f_{\text{cod}(\phi^{CSF})}(ν^P) \rangle
\]

where

\[
μ^P = \left\{\begin{array}{l}
x^1 \mapsto a, \\
x^2 \mapsto y, \\
x^3 \mapsto b, \\
y^λ \mapsto y, \\
z^λ \mapsto g(y)
\end{array}\right\}, \quad ν^P = \left\{\begin{array}{l}
x^1 \mapsto y, \\
x^2 \mapsto z, \\
x^3 \mapsto c, \\
y^λ \mapsto y, \\
z^λ \mapsto z
\end{array}\right\}
\]

Hence
\[
α^{CSF}_P(Φ) = \langle \{x \mapsto f(x^1, x^2, x^3), y \mapsto y^λ, z \mapsto z^λ\}, \{x^1, y^λ\}, \{x^2, y^λ, z^λ\}, \{x^3, z^λ\}, \{\emptyset\}, \{x^2, y^λ\}\rangle
\]

Notice that \(α^{CSF}_P(Φ)\) tallies with the \(ϕ^{CSF}\) of example 3.6. A comparison of the \(Φ\) (of example 3.8) to the \(γ^{CSF}_P(α^{CSF}_P(Φ))\) (of example 3.6) demonstrates the precision with which \(Type_P\) captures sharing, groundness, freeness, compoundness and covering information.
3.3.5 On finiteness

Since widening is required to induce termination on $Sub_{Pvar}$, widening is also necessary to obtain finiteness on $Type_{Pvar}$. A suitable widening can be constructed from $\nabla^k_{\mathcal{CSF}}$, however, by borrowing some machinery from $\mathcal{CSF}$. Specifically, if $\nabla^k_{\mathcal{CSF}}$ details how the $Sub_{Pvar}$ domain can be widened, then $\mathcal{CSF}$ explains how the $Share$ and $Free$ components can be amended to fit with the widened $Sub_{Pvar}$ component.

**Definition 3.13** ($\nabla^k_{\mathcal{CSF}}$) The widening $\nabla^k_{\mathcal{CSF}} : Type_{Pvar} \rightarrow Type_{Pvar}$ is defined by:

$$\nabla^k_{\mathcal{CSF}}(\phi_{\mathcal{CSF}}) = \langle \nabla^k_{\mathcal{CSF}}(\phi^B), \nabla^k_{\mathcal{CSF}}(\phi^S), \phi^F \setminus \text{cod}(\phi^B) \rangle$$

where $\phi^B \circ \nabla^k_{\mathcal{CSF}}(\phi^C) \upharpoonright Pvar = \phi^C$.

Note that the $k$ does not necessarily need to be fixed but can be varied during analysis – just so long as termination is achieved. The correctness of the widening is stated as lemma 3.5 and proven in proof 7.6.

**Lemma 3.5** $\phi_{\mathcal{CSF}} \subseteq_{\mathcal{CSF}} \nabla^k_{\mathcal{CSF}}(\phi_{\mathcal{CSF}})$

4 Abstracting unification

Abstract interpretation can help to focus the development of an analysis by illuminating the connection between an operation (like unification) and its abstract counterpart. In this case, the abstract counterpart for unification is divided into two distinct algorithmic phases. This is a consequence of exploiting confluence. The first phase, detailed in section 4.2, traces compoundness information; whereas the second phase, documented in section 4.1, infers sharing and freeness information. The order of presentation reflects the construction of the analysis: first, an efficient sharing and freeness analysis is synthesised in section 4.1; second, the analysis is extended to trace compoundness to bounded depth in section 4.2.

4.1 Abstracting unification with $Share_{Pvar} \times Free_{Pvar}$

The sharing and freeness component of the analysis is, in fact, interesting within its own right. It is interesting for a number of reasons. First, the sharing and freeness analysis can be used by itself, stand-alone. Second, although the analysis applies many of the ideas of [23], it is both simple and efficient. Third, the analysis has been proved correct. In short, the analysis can be regarded as a systematic and efficient reformulation of the abstract unification algorithm of [23].

Unification is abstracted by tracing the steps of a standard unification algorithm [17]. To trace unification, the abstract algorithm mimics the recursive simplification steps of $mgu$, relegating the solution of simplified equations of the form $v = t$ to a mapping $mgu_{\mathcal{CSF}}$. Similar simplification steps, dubbed pre-unification in [7], are applied in other abstract unification algorithms [23, 7]. The mapping $mgu_{\mathcal{CSF}}$ is defined to abstract a slight variant of $mgu$. Specifically, if $\phi \in mgu\{\{v \bar{=} t\}\}$ and $\phi \in \gamma^{\mathcal{CSF}}_{Pvar}(\phi_{\mathcal{CSF}})$ then $mgu_{\mathcal{CSF}}(u, t, \phi_{\mathcal{CSF}})$ abstracts the composition $\phi \circ \phi$ (rather than $\phi$), that is, $\phi \circ \phi \in \gamma^{\mathcal{CSF}}_{Pvar}(mgu_{\mathcal{CSF}}(u, t, \phi_{\mathcal{CSF}}))$. This spares the need to define an extra (composition) operator.
Definition 4.1 (mgu_{SF})  The relation \( mgu_{SF} : Eqn \times (Share_{P_{var}} \times Free_{P_{var}}) \times (Share_{P_{var}} \times Free_{P_{var}}) \) is defined by:

\[
mgu_{SF}(0, \phi_{SF}, \phi_{SF}) = mgu_{SF}(u = u : E, \phi_{SF}, \phi_{SF}) = mgu_{SF}(t = u : E, \phi_{SF}, \phi_{SF}) = mgu_{SF}(u = t : E, \phi_{SF}, \phi_{SF}) = mgu_{SF}(f(t_i) = f(t'_i) : E, \phi_{SF}, \phi_{SF})
\]

where \( \tau^S = mgu^S(u, t, \phi_{SF}) \) and \( \tau^F = mgu^F(u, t, \phi_{SF}) \).

To define the mappings \( mgu^S \) and \( mgu^F \) (and thus the relation \( mgu_{SF} \)) a number of standard auxiliary operators are required [14, 24]. First, \( rel(t, \phi^S) \) represents the sharing groups of \( \phi^S \) which are relevant to the term \( t \), that is, those sharing groups of \( \phi_{SF} \) which share variables with \( t \). Second, in the absence of useful freeness information worst-case aliasing is assumed. Thus, as in [14, 24], a closure under union operator, \( \ast \), is employed to enumerate all the possible sharing groups that can possibly arise in unification. Third, to succinctly define \( mgu_{SF} \), it is convenient to lift \( \cup \) to sets of sharing groups with a pair-wise union operator, denoted \( \Box \).

Definition 4.2 (rel, \( \ast \) [14, 24] and \( \Box \))

\[
rel(t, \phi^S) = \{ U \in \phi^S \mid U \cap \text{var}(t) \neq \emptyset \}
\]

\[
\phi^{S*} = \phi^S \cup \{ U \cup U' \mid U, U' \in \phi^{S*} \}, \quad \phi^S \Box \phi^S = \{ U \cup U' \mid U \in \phi^S \land U' \in \phi^S \}
\]

The mappings \( mgu^S \) and \( mgu^F \) apply different analysis strategies according to the freeness of \( \phi(v) \) and \( \phi(t) \) for \( \phi \in \gamma_{P_{var}}(\phi_{SF}) \). The default strategy of \( mgu^S \) corresponds to the standard treatment in the abstract solver \( amgu \) of [14].

Definition 4.3 (mgu^S and mgu^F)

\[
mgu^S(u, t, \phi_{SF}) = \begin{cases} 
\psi^S \cup (rel(u, \phi^S) \Box rel(t, \phi^S)) \text{ if } u \in \phi^F \land t \in \phi^F \\
\psi^S \cup (rel(u, \phi^S)^* \Box rel(t, \phi^S)^*) \text{ otherwise}
\end{cases}
\]

where \( \psi^S = \phi^S \setminus (rel(u, \phi^S) \cup rel(t, \phi^S)) \).

\[
mgu^F(u, t, \phi_{SF}) = \begin{cases} 
\phi^F \setminus \text{var}(rel(u, \phi^S)) \quad \text{if } u \in \phi^F \land t \in \phi^F \\
\phi^F \setminus \text{var}(rel(u, \phi^S)) \setminus \text{var}(rel(t, \phi^S)) \quad \text{else if } u \in \phi^F \\
\phi^F \setminus \text{var}(rel(u, \phi^S) \cup rel(t, \phi^S)) \quad \text{else if } t \in \phi^F \\
\phi^F \setminus \text{var}(rel(u, \phi^S) \cup rel(t, \phi^S)) \quad \text{otherwise}
\end{cases}
\]

Note that \( rel(u, \phi^S) \Box rel(t, \phi^S) = \emptyset \) and \( rel(u, \phi^S)^* \Box rel(t, \phi^S)^* = \emptyset \) if \( rel(u, \phi^S) = \emptyset \). Thus, in case one of \( mgu^S \), \( rel(t, \phi^S) \) need not be calculated if \( rel(u, \phi^S) = \emptyset \) and similarly in case two, \( rel(t, \phi^S) \) need not be computed or closed under union if \( rel(u, \phi^S) = \emptyset \). Analogous refinements follow if \( rel(t, \phi^S) = \emptyset \). In addition, observe that \( mgu^S \) applies the refinement suggested in [23], that is, if either \( u \) or \( t \) are free, then the calculation of a closure can be avoided. This contrasts with other freeness algorithms, for example [26], which always calculate a closure unless both \( u \) and \( t \) are free. The correctness of the mappings \( mgu^S \) and \( mgu^F \) is asserted in lemma 4.1 and 4.2. The corresponding proofs are numbered 7.7 and 7.8.
Example 4.1

\[
\phi \in \gamma^{S}_{Pvar}(\phi^{S}) \land \varphi \in mgu(\{\phi(u) = \phi(t)\}) \land \\
\{u\} \cup \text{var}(t) \subseteq \text{Pvar} \land u \notin \text{var}(t) \Rightarrow \varphi \circ \phi \in \gamma^{S}_{Pvar}(mgu^{S}(u, t, \phi^{SF}))
\]

Lemma 4.2

\[
\phi \in \gamma^{F}_{Pvar}(\phi^{F}) \land \varphi \in mgu(\{\phi(u) = \phi(t)\}) \land \\
\{u\} \cup \text{var}(t) \subseteq \text{Pvar} \land u \notin \text{var}(t) \Rightarrow \varphi \circ \phi \in \gamma^{F}_{Pvar}(mgu^{F}(u, t, \phi^{SF}))
\]

The correctness of the relation \(mgu^{SF}\) follows from lemma 4.1 and 4.2 and is stated as theorem 4.1. The corresponding proof is numbered 7.9.

Theorem 4.1

\[
\phi \in \gamma^{S}_{Pvar}(\phi^{SF}) \land \varphi \in mgu(\phi(E)) \land \\
mgu^{SF}(E, \phi^{SF}, \mu^{SF}) \land \text{var}(E) \subseteq \text{Pvar} \Rightarrow \varphi \circ \phi \in \gamma^{S}_{Pvar}(\mu^{SF})
\]

It is convenient shorthand to regard \(mgu^{SF}\) as a mapping, that is, \(mgu^{SF}(E, \phi^{SF}) = \psi^{SF}\) if \(mgu^{SF}(E, \phi^{SF}, \psi^{SF})\). Strictly, it is necessary to show that \(mgu^{SF}(E, \phi^{SF}, \psi^{SF})\) is deterministic for \(mgu^{SF}(E, \phi^{SF})\) to be well-defined. Like in [7], the conjecture is that \(mgu^{SF}\) yields a unique abstract substitution \(\psi^{SF}\) for \(\phi^{SF}\) regardless of the order in which \(E\) is solved (though, in practice, any \(\psi^{SF}\) is safe).

Example 4.1 illustrates that the simplicity of the analysis is not gained at the expense of precision. Indeed the analysis seems to possess much of the power of the original sharing and freeness analysis of [23].

Example 4.1

Adapting an example from [23], consider the computation of \(mgu^{SF}(E, \phi^{SF})\) where

\[
E = \begin{cases}
x_1 = f(y_1, y_2), \\
x_1 = f(y_3, y_4), \\
y_3 = a, \\
y_5 = x_2, \\
y_6 = x_2, \\
y_6 = f(x_1, x_3)
\end{cases}, \quad \phi^{SF} = \langle \{0, \{x_1\}, \{x_2\}, \{x_3\}, \\
\{y_1, y_2\}, \{y_2\}, \{y_3\}, \{y_5\}, \{y_6\}\rangle, \\
\{x_1, x_2, x_3, y_1, y_3, y_5, y_6\}\rangle
\]

Thus, putting \(\phi^{SF}_i = \phi^{SF}\) and considering each equation of \(E = \{u_i = t_i\}_{i=1}^6\) in turn, then \(\phi^{SF}_{i+1} = (mgu^{S}(u_i, t_i, \phi^{SF}_i), mgu^{F}(u_i, t_i, \phi^{SF}_i))\) where

\[
\phi^{SF}_1 = \langle \{0, \{x_1\}, \{x_2\}, \{x_3\}, \{y_1, y_2\}, \{y_2\}, \{y_3\}, \{y_5\}, \{y_6\}\rangle, \\
\{x_1, x_2, x_3, y_1, y_3, y_5, y_6\}\rangle
\]

\[
\phi^{SF}_2 = \langle mgu^{S}(x_1, f(y_1, y_2), \phi^{SF}_1), mgu^{F}(x_1, f(y_1, y_2), \phi^{SF}_1) \rangle \\
= \langle \{0, \{x_1, y_1, y_2\}, \{x_1, y_2\}, \{x_2\}, \{x_3\}, \{y_3\}, \{y_5\}, \{y_6\}\rangle, \\
\{x_2, x_3, y_1, y_3, y_5, y_6\}\rangle
\]

\[
\phi^{SF}_3 = \langle \{0, \{x_1, y_1, y_2, y_3\}, \{x_1, y_2, y_3\}, \{x_2\}, \{x_3\}, \{y_5\}, \{y_6\}\rangle, \\
\{x_2, x_3, y_5, y_6\}\rangle
\]

\[
\phi^{SF}_4 = \langle \{0, \{x_2\}, \{x_3\}, \{y_5\}, \{y_6\}\}, \{x_2, x_3, y_5, y_6\}\rangle
\]

\[
\phi^{SF}_5 = \langle \{0, \{x_2, y_5\}, \{x_3\}, \{y_6\}\}, \{x_2, x_3, y_5, y_6\}\rangle
\]

\[
\phi^{SF}_6 = \langle \{0, \{x_2, y_5, y_6\}, \{x_3\}, \{x_2, x_3, y_5, y_6\}\rangle
\]

\[
\phi^{SF}_7 = \langle \{0, \{x_2, x_3, y_5, y_6\}, \{x_3\}\rangle
\]

15
Therefore \( mgu^{\text{CSF}}(E, \phi^{\text{CSF}}) = \langle \{ \emptyset, \{ x_2, x_3, y_5, y_6 \} \}, \{ x_3 \} \rangle \). The freeness analysis of [23] similarly infers (modulo a projection operation [23]) that only \( x_2, x_3, y_5 \) and \( y_6 \) are possibly aliased and that \( x_3 \) is free.

### 4.2 Abstracting unification with Type_{Pvar}

The task of extending abstract unification from \( \text{Share}_{Pvar} \times \text{Free}_{Pvar} \) to \( \text{Type}_{Pvar} \) boils down to defining a mapping \( mgu^{\text{CSF}} \). The specific requirement is for a mapping \( mgu^{\text{CSF}} \) such that if \( \varphi \in mgu(\phi(E)) \), \( \phi \in \gamma_{Pvar}^{\text{CSF}}(\phi^{\text{CSF}}) \) and \( \mu^{\text{CSF}} = mgu^{\text{CSF}}(E, \phi^{\text{CSF}}) \) then \( \varphi \circ \phi \in \gamma_{Pvar}^{\text{CSF}}(\mu^{\text{CSF}}) \).

As with \( mgu^{\text{CSF}} \), the problem is cast in a way so as to avoid a composition operation.

The mapping \( mgu^{\text{CSF}} \) builds on \( mgu^{\text{SF}} \) by detailing how \( \mu^{\text{CSF}} \) can be computed. In addition, \( mgu^{\text{CSF}} \) explains how \( \mu^{\text{CSF}} \) can be calculated with \( mgu^{\text{SF}} \) from \( \mu^{\text{C}} \), \( \phi^{\text{CSF}} \) and \( E \). Interestingly, \( \mu^{\text{C}} \) is calculated by applying (concrete) unification to solve the equation set \( \phi^{\text{C}}(E) \). The intuition is that if \( \varphi^B \) is a unifier of \( mgu(\phi(E)) \) then \( \varphi^B \circ \phi^C \uparrow Pvar \) is a likely to be a good candidate for \( \mu^C \). One technical point, however, is that the substitution \( \varphi^B \circ \phi^C \uparrow Pvar \) is not always an element of \( \text{Sub}_{Pvar}^B \).

**Example 4.2** Consider, for example, the abstraction of \( mgu(E, \phi) \) where \( \phi \) is represented by \( \phi^{\text{CSF}}, E \) is defined in example 4.1 and \( \phi^{\text{C}} = \{ x_1 \mapsto x_1^A, x_2 \mapsto x_2^A, x_3 \mapsto x_3^A, y_1 \mapsto y_1^A, \ldots, y_6 \mapsto y_6^A \} \). Thus, calculating \( \varphi^B \in mgu(\phi^{\text{C}}(E)) \) and \( \varphi^B \circ \phi^C \uparrow Pvar \) yields

\[
\varphi^B = \begin{cases}
  x_1^A & \mapsto f(a, y_2^B), \\
  x_2^A & \mapsto f(a, y_2^B, x_3^A), \\
  y_1^A & \mapsto a, \\
  y_2^A & \mapsto y_2^A, \\
  y_3^A & \mapsto f(a, y_2^B, x_3^A), \\
  y_6^A & \mapsto f(a, y_2^B, x_3^A)
\end{cases}, \quad \varphi^B \circ \phi^C \uparrow Pvar = \begin{cases}
  x_1^A & \mapsto f(a, y_2^B), \\
  x_2^A & \mapsto f(f(a, y_2^B, x_3^A)), \\
  y_1^A & \mapsto a, \\
  y_3^A & \mapsto a, \\
  y_4^A & \mapsto y_2^A, \\
  y_5^A & \mapsto f(f(a, y_2^B, x_3^A)), \\
  y_6^A & \mapsto f(f(a, y_2^B, x_3^A))
\end{cases}
\]

Therefore, in general, \( \psi \circ \phi^C \uparrow Pvar \not\in \text{Sub}_{Pvar}^B \).

One way to capture the compoundedness of \( \varphi^B \circ \phi^C \uparrow Pvar \) in terms of \( \text{Sub}_{Pvar}^B \) is to find a substitution \( \psi^B \) such that \( \psi^B \subseteq \varphi^B \) with the required \( \psi^B \circ \phi^C \uparrow Pvar \in \text{Sub}_{Pvar}^B \). Safety follows since \( \psi^B \subseteq \varphi^B \). To maximise the precision, however, it is important to choose the least general \( \psi^B \) with the \( \psi^B \subseteq \varphi^B \) property. This observation on precision and ordering leads to the definition of the poset \( \text{Sub}_{Pvar}^B (\subseteq Pvar) \).

**Definition 4.4 (\text{Sub}_{Pvar}^B)** The poset \( \text{Sub}_{Pvar}^B (\subseteq Pvar) \) is defined by:

\[
\text{Sub}_{Pvar}^B = \left\{ \psi^B \in \text{Sub}_{Pvar}^B \mid \text{dom}(\psi^B) \subseteq Pvar \land \forall u^* \mapsto t_u \in \phi^B, t_u \in \text{Term}_u^* \right\}
\]

\( \text{Sub}_{Pvar}^B (\subseteq Pvar) \) is a poset (rather than a preorder) by virtue of its restricted variable term bindings. Also, when equipped with a top element \( \text{fail}^B \), a lub and \( \text{glb}^B \) (corresponding to unification) and a glb \( \cap^B \) (corresponding to anti-unification [17]), \( \text{Sub}_{Pvar}^B \cup \{ \text{fail}^B \} (\subseteq Pvar) \) is a complete lattice.

With the aid of the lub, a mapping \( \text{Sub}_{Pvar}^B \) can be formulated which calculates the most accurate, safe substitution \( \varphi^B \) such that \( \psi^B \circ \phi^C \uparrow Pvar \in \text{Sub}_{Pvar}^B \).
Definition 4.5 (sub\textsubscript{Pvar}) The mapping sub\textsubscript{Pvar} : Sub\textsubscript{Pvar} → Sub\textsubscript{Pvar} is defined by:

\[
\text{sub}\textsubscript{Pvar}(\phi^B) = \bigcup^B \{ \phi^B \in \text{Sub}\textsubscript{Pvar} \mid \phi^B \subseteq \text{Pvar} \phi^B \}
\]

Although the mapping sub\textsubscript{Pvar} is defined in terms of the hub of a possibly large set, sub\textsubscript{Pvar} can be implemented efficiently.

Example 4.3 Continuing with example 4.2, put \(\psi^B = \text{sub}\textsubscript{Pvar}(\phi^B)\) and thus

\[
\psi^B = \begin{cases}
  x_1 &\mapsto f(a, x_1), \\
  x_2 &\mapsto f(f(a, x_2^1), x_2^2), \\
  y_1 &\mapsto a, \\
  y_3 &\mapsto a, \\
  y_4 &\mapsto f(f(a, y_3^1), y_3^2), \\
  y_5 &\mapsto f(f(a, y_3^1), y_3^2) \\
\end{cases}, \quad \psi^B \circ \phi^C \upharpoonright \text{Pvar} = \begin{cases}
  x_1 &\mapsto f(a, x_1), \\
  x_2 &\mapsto f(f(a, x_2^1), x_2^2), \\
  y_1 &\mapsto a, \\
  y_3 &\mapsto a, \\
  y_4 &\mapsto f(f(a, y_3^1), y_3^2), \\
  y_5 &\mapsto f(f(a, y_3^1), y_3^2) \\
\end{cases}
\]

Observe that \(\psi^B \circ \phi^C \upharpoonright \text{Pvar} \in \text{Sub}\textsubscript{Pvar} \) as required.

With the compoundness component \(\mu^F = \psi^B \circ \phi^C \upharpoonright \text{Pvar} \) computed, the problem reduces to calculating the sharing and freeness component \(\mu^{SF}\). The sharing and freeness component is computed by transforming the Type\textsubscript{Pvar} problem into a Share\textsubscript{Pvar} × Free\textsubscript{Pvar} problem. The idea is that if \(mgu^{SF}\) is efficient and fast, then \(mgu^{SF}\) should be used wherever possible. The transformation is underpinned by lemma 4.3 and theorem 4.2. In theorem 4.2 confluence is used to show that \([\psi^D \circ (\psi^B \circ \phi^C \upharpoonright \text{Pvar})] \upharpoonright \text{Pvar} = \phi^D\) where \(\psi^D = mgu(\theta \circ \phi^B \circ \phi^C (E \cup \theta)\) and \(\theta = mgu(\theta \cup \psi^B)\). This closes the gap between Type\textsubscript{Pvar} and Share\textsubscript{Pvar} × Free\textsubscript{Pvar} since if it is possible to find a \(\psi^{SF}\) such that \(\theta \in \tau^P\) then it is also possible to compute \(\mu^{SF}\) with \(mgu^{SF}\). More exactly, \(mgu^{SF}(\psi^B \circ \phi^C (E), \phi^{SF}) = \mu^{SF}\). This is the first result. The second result, lemma 4.3, asserts that an auxiliary operation \(mgu^{BSF}\) can provide the required \(\psi^{SF}\). To be precise, if \(mgu^{BSF}(\psi^B, \phi^{SF}, \phi^{SF})\) then \(\psi^B \circ \phi^{SF} \upharpoonright \text{Pvar} \in \text{Sub}\textsubscript{Pvar} \) as required.

Thus both results, when applied together, define a procedure for computing \(\mu^{SF}\). In short, \(\mu^{SF} = mgu^{SF}(\psi^B \circ \phi^C (E), \phi^{SF})\) where \(mgu^{BSF}(\psi^B, \phi^{SF}, \phi^{SF})\).

The operation \(mgu^{BSF}\) is designed to calculate an abstraction for \(\theta = mgu(\theta \cup \psi^B)\). This, in fact, can be calculated relatively straightforwardly because of the simple structure of \(\psi^B\). First, the domain variables of \(\psi^B\) do not share in \(\psi^B\): \(\text{var}(\psi^B(u)) \cap \text{var}(\psi^B(v)) = \emptyset\) for any pair of distinct variables \(u, v \in \text{dom}(\psi^B)\). Second, the domain variables of \(\psi^B\) are guaranteed to be linear: \(\chi(\psi^B(u)) = 1\) for all \(u \in \text{dom}(\psi^B)\). This means that \(mgu^{BSF}\) can calculate an accurate abstraction for \(\theta\).

Definition 4.6 (mgu\textsubscript{BSF}) The relation mgu\textsubscript{BSF} : Sub\textsubscript{Pvar} × (Share\textsubscript{Pvar} × Free\textsubscript{Pvar}) × (Share\textsubscript{Pvar} × Free\textsubscript{Pvar}) is defined by:

\[
\text{mgu}\textsubscript{BSF}(\emptyset, \phi^{SF}, \phi^{SF}) \quad \text{mgu}\textsubscript{BSF}(u = t : E, \phi^{SF}, \phi^{SF}) \text{ if } \text{mgu}\textsubscript{BSF}(E, \tau^{SF}, \phi^{SF})
\]

where \(\tau^S = \text{mgu}\textsubscript{BS}(u, t, \phi^{SF})\) and \(\tau^F = \text{mgu}\textsubscript{BF}(u, t, \phi^{SF})\).

Definition 4.7 (mgu\textsubscript{BS} and mgu\textsubscript{BF})

\[
\text{mgu}\textsubscript{BS}(u, t, \phi^{SF}) = \begin{cases}
  \tau^S \cup (U_u \square U_t) & \text{if } u \in \phi^T \\
  \tau^T \cup (U_u \square U_t) & \text{otherwise}
\end{cases}
\]
Theorem 4.3

\[ \tau^S = \phi^S \setminus \text{rel}(u, \phi^S) \]
\[ U_u = \{ U_u \setminus \{ u \} \mid U_u \in \text{rel}(u, \phi^S) \} \]
\[ U_t = \{ \{ u \} \mid u \in \text{var}(t) \} \]
\[ U_t = \{ U_t \subseteq \text{var}(t) \mid U_t \neq \emptyset \} \]

\[ mg_u^{\text{BS}}(u, t, \phi^S) = \begin{cases} (\phi^F \setminus \text{var}(\text{rel}(u, \phi^S))) \cup \text{var}(t) & \text{if } u \in \phi^F \\ (\phi^F \setminus \text{var}(\text{rel}(u, \phi^S))) & \text{otherwise} \end{cases} \]

The irregular structure of \( mg_u^{\text{BS}} \) and \( mg_u^{\text{SF}} \) stems from the requirement that \( \theta \in \gamma_{\text{cod}(\psi^{B} \circ \phi^{C} \uparrow \text{Pvar})}^{\phi^S}(\phi^S) \) whereas \( \phi^D \in \gamma_{\text{cod}(\phi^{C})}^{\phi^D}(\phi^S) \) (referring to the \( \phi^D \) of lemma 4.3). Put another way, the \( \phi^S \) is expressed in terms of the variables \( \text{cod}(\psi^{B} \circ \phi^{C} \uparrow \text{Pvar}) \) whereas \( \phi^D \) is represented in terms of the variables \( \text{cod}(\phi^{C}) \). This means that \( mg_u^{\text{BS}, \text{SF}} \) has to incrementally extend from \( \text{Share}_{\text{cod}(\phi^{C})} \times \text{Free}_{\text{cod}(\phi^{C})} \) to \( \text{Share}_{\text{cod}(\psi^{B} \circ \phi^{C} \uparrow \text{Pvar})} \times \text{Free}_{\text{cod}(\psi^{B} \circ \phi^{C} \uparrow \text{Pvar})} \). This is the main technicality involved in establishing lemma 4.3 in proof 4.3.

As with \( mg_u^{\text{SF}} \), it is a convenient shorthand to regard \( mg_u^{\text{BS}, \text{SF}} \) as a mapping. Like before, \( mg_u^{\text{BS}, \text{SF}}(\psi^{B}, \phi^S, \phi^S) \) yields a safe \( \theta^{\text{SF}} \) regardless of the order in which \( \psi^{B} \) is solved.

Lemma 4.3

\[ \phi \uparrow \text{Pvar} = \phi^D \circ \phi^C \uparrow \text{Pvar} \]
\[ \phi^B \in mg_u^{\phi^C}(E) \quad \land 
\[ \psi^{B} = \text{sub}_{\text{Pvar}}(\phi^B) \quad \land 
\[ \theta \in mg_u^{\phi^B \cup \phi^D} \quad \land 
\[ mg_u^{\text{BS}, \text{SF}}(\psi^{B}, \phi^S, \phi^S) \Rightarrow
\[ \theta \in \gamma_{\text{cod}(\psi^{B} \circ \phi^{C} \uparrow \text{Pvar})}^{\phi^S}(\phi^S) \]

Definition 4.8 and theorem 4.2 summarise how \( mg_u^{\text{BS}, \text{SF}} \) and \( mg_u^{\text{SF}} \) fit together to form \( mg_u^{\text{CS}, \text{SF}} \). The corresponding proof is labelled 7.11.

Definition 4.8 The mapping \( mg_u^{\text{CS}, \text{SF}} : \text{Eqn} \times \text{Type}_{\text{Pvar}} \rightarrow \text{Type}_{\text{Pvar}} \) is defined by:

\[ mg_u^{\text{CS}, \text{SF}}(E, \phi^{\text{CS}, \text{SF}}) = (\psi^{B} \circ \phi^{C} \uparrow \text{Pvar}, \mu^{S}, \mu^{F}) \text{ where }
\[ \phi^B \in mg_u^{\phi^C}(E) \land 
\[ \psi^{B} = \text{sub}_{\text{Pvar}}(\phi^B) \land 
\[ \mu^{SF} = mg_u^{\phi^S}(\psi^{B} \circ \phi^{C}(E), mg_u^{\text{BS}, \text{SF}}(\psi^{B}, \phi^S, \phi^S)) \]

Theorem 4.2

\[ \phi \in \gamma_{\text{Pvar}}^{\phi^{\text{CS}, \text{SF}}}(\phi^{\text{CS}, \text{SF}}) \land \phi \in mg_u^{\phi}(E) \land \text{var}(E) \subseteq \text{Pvar} \Rightarrow \phi \circ \phi \in \gamma_{\text{Pvar}}^{\phi^{\text{CS}, \text{SF}}}(mg_u^{\text{CS}, \text{SF}}(E, \phi^{\text{CS}, \text{SF}})) \]

Example 4.4 Consider the calculation of \( mg_u^{\text{CS}, \text{SF}}(E, \phi^{\text{CS}, \text{SF}}) = \mu^{\text{CS}, \text{SF}} \) adopting the E of example 4.1 and where

\[ \phi^C = \{ x_1 \leftarrow x_1, x_2 \leftarrow x_2, x_3 \leftarrow x_3, y_1 \leftarrow y_1, \ldots, y_6 \leftarrow y_6 \} 
\[ \phi^S = \emptyset, \{ x_1 \}, \{ x_2 \}, \{ x_3 \}, \{ y_1 \}, \{ y_2 \}, \{ y_3 \}, \{ y_4 \}, \{ y_5 \}, \{ y_6 \} \}
\[ \phi^F = \{ x_1, x_2, x_3, y_1, y_2, y_3, y_4, y_5, y_6 \} \]
Thus, using the $\varphi^B \in mgu(\phi^C(E))$ and $\psi^B = sub_{\text{var}}(\varphi^B)$ of example 4.3, then if $\psi^B \circ \phi^C(\psi^B)$ is simplified to $E'$ through pre-unification

$$
\psi^B \circ \phi^C(\psi^B) = \begin{cases}
    f(a, x^1_1) = f(a, y^1_1), \\
    f(a, x^1_2) = f(a, y^1_2), \\
    f(a, x^2_1) = f(a, x^2_2), \\
    f(a, y^2_1) = f(a, x^2_1), \\
    f(a, y^2_2) = f(a, y^2_1),
\end{cases}
\quad E' = \begin{cases}
    x^1_1 = y^1_2, \\
    x^1_2 = y^1_4, \\
    y^2_1 = x^2_2, \\
    y^2_2 = x^2_1,
\end{cases}
$$

Moving onto the calculation of $mgu^{\text{RSF}}(\psi^B, \phi^S)$, putting $\psi^S = \phi^{\text{SF}}$, and considering each equation of $\phi^B = \{ a_i \,= \,t^B_{i1} \}_{i=1}^5$ in turn.

$$
\begin{align*}
\psi^1_1 &= \langle \{0, \{x^1_1\}, \{x^1_2\}, \{x^2_1\}, \{y^1_1, y^2_1\}, \{y^2_2\}, \{y^3\}, \{y^3_1\}, \{y^3_2\} \rangle, \\
&\quad \langle x^1_1, x^1_2, x^2_1, y^1_1, y^2_2, y^1_3, y^1_4 \rangle \\
\psi^1_2 &= \langle mgu^{\text{BSF}}(x^1_1, f(a, x^1_1), \psi^1_1), mgu^{\text{BSF}}(x^1_2, f(a, x^1_2), \psi^1_1) \rangle \\
&= \langle \{0, \{x^1_1\}, \{x^1_2\}, \{x^2_1\}, \{y^1_1, y^2_1\}, \{y^2_2\}, \{y^3\}, \{y^3_1\}, \{y^3_2\} \rangle, \\
&\quad \langle x^1_1, x^1_2, x^2_1, y^1_1, y^2_2, y^1_3, y^1_4 \rangle \\
\psi^1_3 &= \langle \{0, \{x^1_1\}, \{x^1_2\}, \{x^2_1\}, \{y^1_1, y^2_1\}, \{y^2_2\}, \{y^3\}, \{y^3_1\}, \{y^3_2\} \rangle, \\
&\quad \langle x^1_1, x^1_2, x^2_1, y^1_1, y^2_2, y^1_3, y^1_4 \rangle \\
\psi^1_4 &= \langle \{0, \{x^1_1\}, \{x^1_2\}, \{x^2_1\}, \{y^1_1, y^2_1\}, \{y^2_2\}, \{y^3\}, \{y^3_1\}, \{y^3_2\} \rangle, \\
&\quad \langle x^1_1, x^1_2, x^2_1, y^1_1, y^2_2, y^1_3, y^1_4 \rangle \\
\psi^1_5 &= \langle \{0, \{x^1_1\}, \{x^1_2\}, \{x^2_1\}, \{y^1_1, y^2_1\}, \{y^2_2\}, \{y^3\}, \{y^3_1\}, \{y^3_2\} \rangle, \\
&\quad \langle x^1_1, x^1_2, x^2_1, y^1_1, y^2_2, y^1_3, y^1_4 \rangle \\
\psi^1_6 &= \langle \{0, \{x^1_1\}, \{x^1_2\}, \{y^1_1, y^2_1\}, \{y^2_2\}, \{y^3\}, \{y^3_1\}, \{y^3_2\} \rangle, \\
&\quad \langle x^1_1, x^1_2, x^2_1, y^1_1, y^2_2, y^1_3, y^1_4 \rangle \\
\psi^1_7 &= \langle \{0, \{x^1_1\}, \{x^1_2\}, \{y^1_1, y^2_1\}, \{y^2_2\}, \{y^3\}, \{y^3_1\}, \{y^3_2\} \rangle, \\
&\quad \langle x^1_1, x^1_2, x^2_1, y^1_1, y^2_2, y^1_3, y^1_4 \rangle \\
\end{align*}
$$

With $\psi^{\text{SF}} = \psi^S$, $mgu^{\text{SF}}(E', \psi^{\text{SF}}) = \mu^{\text{SF}} = \{0, \{x^2_1, x^2_3, y^3_2, y^3_6\}\}$. Thus $\mu^{\text{SF}}$ is given by

$$
\mu^C = \begin{cases}
    x^1_1 \mapsto f(a, x^1_1), \\
    x^2_2 \mapsto f(a, x^2_2), \\
    x^2_3 \mapsto x^2_3, \\
    y_1 \mapsto a, \\
    y_2 \mapsto y_2^1, \\
    y_3 \mapsto a, \\
    y_4 \mapsto y_4^1, \\
    y_5 \mapsto f(a, y_5^1), \\
    y_6 \mapsto f(a, y_6^1),
\end{cases}
\quad \mu^S = \{0, \{x^2_1, x^2_3, y^3_2, y^3_6\}\},
\quad \mu^F = \{x^2_1, x^2_3, y^3_2, y^3_6\}
$$

Clearly $\mu^{\text{SF}}$ is a refinement of the $\{0, \{x_2, x_3, y_5, y_6\}\}$ derived in example 4.1. In addition to recording the sharing between $x_2, x_3, y_5$ and $y_6$ and freeness of $x_3$, $\mu^{\text{SF}}$ records term structure in $\mu^C$, details sharing to the precision of sub-terms in $\mu^S$, and infers which sub-terms are free in $\mu^F$. 
Example 4.5 By widening $\psi^S$, finiteness can be enforced and the precision of the analysis can be adjusted to suit the application. Thus, returning to the calculation of $\text{mgw}^{S\cdot SF}(\{a = b\}, \phi^{S\cdot SF}) = \mu^{S\cdot SF}$ in example 4.4, consider the effect of throttling $\psi^S$ to depth-0 and depth-1 to obtain

$$\psi_0^S = \{ y_1^a \mapsto a, y_3^a \mapsto a \}, \quad \psi_1^S = \left\{ \begin{array}{l} x_1^a \mapsto f(a, x_1^a), \\ x_2^a \mapsto f(x_2^a, x_3^a), \\ y_1^a \mapsto a, \\ y_3^a \mapsto a, \\ y_5^a \mapsto f(y_5^a, y_6^a), \\ y_6^a \mapsto f(y_6^a, y_7^a) \end{array} \right\}$$

Like before, $\psi_i^S \circ \phi^C(E)$ can be simplified via pre-unification to obtain $E'_i$

$$E'_0 = \left\{ \begin{array}{l} x_1^a = f(a, y_2^a), \\ x_2^a = f(a, y_3^a), \\ y_3^a = x_2^a, \\ y_5^a = x_2^a, \\ y_6^a = f(x_1^a, x_3^a) \end{array} \right\}, \quad E'_1 = \left\{ \begin{array}{l} x_1^a = y_2^a, \\ x_1^a = y_4^a, \\ y_2^a = x_2^a, \\ y_5^a = x_2^a, \\ y_6^a = f(a, x_1^a) \end{array} \right\}$$

Calculating $\text{mgw}^{B\cdot SF}(\psi_i^S, \phi^{S\cdot SF}) = \phi_i^{S\cdot SF}$ gives

$$\phi_i^{CS} = \langle \{0, \{x_1^a\}, \{x_2^a\}, \{x_3^a\}, \{y_2^a\}, \{y_5^a\}, \{y_6^a\}\}, \{x_1^a, x_2^a, x_3^a, y_5^a, y_6^a\} \rangle$$

$$\phi_i^{CS} = \langle \{0, \{x_1^a\}, \{x_2^a\}, \{x_3^a\}, \{y_2^a\}, \{y_5^a\}, \{y_6^a\}\}, \{x_1^a, x_2^a, x_3^a, y_5^a, y_6^a\} \rangle$$

Finally $\text{mgw}^{S\cdot SF}(E'_i, \phi_i^{S\cdot SF})$ defines $\mu_i^{S\cdot SF}$ and $\psi_i^S \circ \phi^C$ defines $\mu_i^C$

$$\mu_0^{S\cdot SF} = \langle \{0, \{x_2^a, x_3^a, y_5^a, y_6^a\}\}, \{x_3^a\} \rangle$$

$$\mu_1^{S\cdot SF} = \langle \{0, \{x_2^a, x_3^a, y_5^a, y_6^a\}\}, \{x_2^a, x_3^a, y_5^a, y_6^a\} \rangle$$

$$\mu_0^C = \left\{ \begin{array}{l} x_1 \mapsto x_1^a, \\ x_2 \mapsto x_2^a, \\ x_3 \mapsto x_3^a, \\ y_1 \mapsto a, \\ y_2 \mapsto y_2^a, \\ y_3 \mapsto a, \\ y_4 \mapsto y_4^a, \\ y_5 \mapsto y_5^a, \\ y_6 \mapsto y_6^a \end{array} \right\}, \quad \mu_1^C = \left\{ \begin{array}{l} x_1 \mapsto f(a, x_1^a), \\ x_2 \mapsto f(x_2^a, x_3^a), \\ x_3 \mapsto x_3^a, \\ y_1 \mapsto a, \\ y_2 \mapsto y_2^a, \\ y_3 \mapsto a, \\ y_4 \mapsto y_4^a, \\ y_5 \mapsto f(y_5^a, y_6^a), \\ y_6 \mapsto f(y_6^a, y_7^a) \end{array} \right\}$$

Note that even at depth-0, the analysis captures compoundness information that cannot be derived by a conventional Share x Free analysis. Note too that since variables only occur in $\mu$ at level 1, then in terms of sharing and freeness, depth-1 analysis is just as accurate as depth-2 analysis. Finally observe that depth-0 analysis fails to infer the sharing and freeness of sub-terms.
Note that \( mg_u^{CSF} \) and each of its constituent parts are independent of \( k \). Thus \( mg_u^{CSF} \) is an abstract equation solver for depth-\( k \) abstractions of arbitrary \( k \). However, arbitrary \( k \) can lead to non-terminating computations and therefore, in general, some method for enforcing convergence and finiteness is required. One simple way of ensuring termination is to widening at the level of \( mg_u^{CSF} \). This approach requires just one additional construction, \( mg_u^{CSF}_k \), which thresholds the abstract unifier to depth-\( k \). This approach compares very favourably with the finiteness machinery which is detailed in [16].

**Definition 4.9 (\( mg_u^{CSF}_k \))** The mapping \( mg_u^{CSF}_k : \text{Eqn} \times \text{Sub}_{Pvar}^{CSF} \rightarrow \text{Sub}_{Pvar}^{CSF} \) is defined by:

\[
mg_u^{CSF}_k(E, \phi^{CSF}) = \nabla_k^{CSF}(mg_u^{CSF}(E, \phi^{CSF}))
\]

Then, with the addition of some renaming machinery, \( mg_u^{CSF}_k \) defines a depth-\( k \) version of \( unifyf, unifyy^{CSF}_k \). To define \( unifyy^{CSF}_k \) and prove safety it is necessary to introduce an abstract restriction operator.

**Definition 4.10 (abstract restriction)** The abstract restriction operator, \( \gamma^{CSF} \), is defined by:

\[
\phi^{CSF} \mid CSF U = (\phi^C \mid U, \phi^S \mid U, \phi^F \mid U) \quad \phi^C \mid U = \phi^C \mid \nabla U \\
\phi^S \mid U = \{U \cup U' \mid U' \in \phi^S\} \\
\phi^F \mid U = U \cap \phi^F
\]

The definition of \( unifyy^{CSF}_k \) is given below with its safety stated as theorem 4.3. Theorem 4.3 assumes \( \text{var}(a) \cup \text{var}(b) \subseteq \text{Pvar} \) and is established by proof 7.12.

**Definition 4.11 (\( unifyy^{CSF}_k \))** The mapping \( unifyy^{CSF}_k : \text{Atom} \times \text{Type}_{Pvar} \times \text{Atom} \times \text{Type}_{Pvar} \rightarrow \text{Type}_{Pvar} \) is defined by:

\[
unifyy^{CSF}_k(a, \phi^{CSF}, b, \psi^{CSF}) = mg_u^{CSF}_k(\{a = \Upsilon(b)\}, \phi^{CSF} \cup \Upsilon(\psi^{CSF})) \mid CSF \text{ Pvar}
\]

**Theorem 4.3 (local safety of \( unifyy^{CSF}_k \))**

\[
\Phi \subseteq \gamma^{CSF}_{Pvar}(\phi^{CSF}) \land \Psi \subseteq \gamma^{CSF}_{Pvar}(\psi^{CSF}) \Rightarrow \\
unifyy^{CSF}_k(a, \Phi, b, \Psi) \subseteq \gamma^{CSF}_{Pvar}(unifyy^{CSF}_k(a, \phi^{CSF}, b, \psi^{CSF}))
\]

**Example 4.6** For the sake of comparison with the freeness analysis of [23], consider the calculation of \( unifyy^{CSF}_k(a, \phi^{CSF}_a, b, \phi^{CSF}_b) \) for \( k = 1, 2 \) and 3 where \( a = p(x_1, x_1, a, x_2, x_1, x_3, x_6, x_3), b = p(f(x_1, x_2), f(x_3, x_4), x_3, x_5, x_6, x_3) \) and

\[
\begin{align*}
\phi^a &= \{x_1 \mapsto x_1^\lambda, x_2 \mapsto x_1^\lambda, x_3 \mapsto x_3^\lambda, x_6 \mapsto x_6^\lambda\}, \\
\phi^b &= \{x_1 \mapsto x_1^\lambda, \ldots, x_6 \mapsto x_6^\lambda\}, \\
\phi^a_S &= \{0, \{x_1^\lambda\}, \{x_2^\lambda\}, \{x_3^\lambda\}\}, \\
\phi^b_S &= \{0, \{x_1^\lambda, x_3^\lambda\}, \{x_3^\lambda\}\}, \\
\phi^a_F &= \{x_1^\lambda, x_2^\lambda, x_3^\lambda\}, \\
\phi^b_F &= \{x_1^\lambda, x_3^\lambda, x_5^\lambda, x_6^\lambda\}
\end{align*}
\]

Thus, supposing \( \Upsilon(x_i) = y_i \) where \( x_i \in \text{Pvar} \) and \( y_i \in \text{Rvar} \), then the computation reduces to \( \text{unifyy}^{CSF}_k(\{a = \Upsilon(b)\}, \phi^{CSF} \cup \Upsilon(\phi^{CSF}_b)) \mid CSF \text{ Pvar} \) which, in turn, simplifies to \( \nabla_k^{CSF}(mg_u^{CSF}(E, \phi^{CSF})) \mid CSF \text{ Pvar} \). An example of 4.4. Thus, by example 4.4, it follows that \( unifyy^{CSF}_k(a, \phi^{CSF}_a, b, \phi^{CSF}_b) = \varphi^{CSF}_k \) where

\[
\begin{align*}
\varphi^C &= \begin{cases} 
    x_1 \mapsto x_1^\lambda, \\
    x_2 \mapsto x_1^\lambda, \\
    x_3 \mapsto x_3^\lambda,
\end{cases} \\
\varphi^S &= \{0, \{x_2^\lambda, x_3^\lambda\}\}, \\
\varphi^F &= \{x_2^\lambda, x_3^\lambda\}
\end{align*}
\]
By comparison, the freeness analysis of [23] likewise infers that \(\varphi_3\) is free and that \(x_2\) and \(x_3\) share. However, the analysis of [23] cannot infer the compoundness of \(x_1\) and \(x_2\), nor which sub-terms of \(x_2\) and \(x_3\) share, nor which sub-terms of \(x_2\) are free.

5 Related and future work

Recently, four relevant proposals for computing freeness information have been put forward in the literature. In the first proposal [8], multiple domains and analyses are run in lock step. At each step, the abstract substitutions derived by the different analyses are compared and refined in order to improve the precision. This paper follows the trend for simultaneously tracing different properties (namely sharing, freeness and compoundness), explaining how accuracy and efficiency can be further improved by exploiting confluence. In particular the paper reports a depth-\(k\) analysis which cannot be synthesised in terms of the combined domain approach.

In the second proposal [6], the correctness of a sharing and freeness analysis is considered. An abstract unification algorithm is proposed as a basis for constructing accurate freeness analyses with a domain formulated in terms of a system of abstract equations. Safety follows because the abstract algorithm mimics the unification algorithm in an intuitive way. Correctness is established likewise here. One important distinction between the two works is that this paper uses the domain \(\text{Sub}_{\text{Pvar}}\) to potentially encode more accurate sharing and freeness information than the abstract equations of [6]. Consequently, a depth-\(k\) analysis cannot be derived from the abstract equations of [6]. Also, as pointed out in [3], “it is doubtful whether it (the abstract unification algorithm of [6]) can be the basis for a very efficient analysis”.

Third, in [16], the format of sharing groups is revised to capture structural properties of substitutions. An abstract substitution is represented as a set of sharing groups where a sharing group is a (possibly empty) set of program variable and path pairs. The paths permit sharing groups to record the positions of shared variables within a binding, that is, where the shared variable occurs in the terms to which the program variables are bound. Correctness is proved. The usefulness of the approach, however, is compromised by its potential inefficiency. The essential problem is that paths are required to be concatenated, compared and truncated at almost every stage of abstract unification algorithm. This can be expensive. Moreover, because of the way paths are widened, much of the formal analysis machinery has to be duplicated: first, a depth-\(\infty\) analysis is formulated; second, a depth-\(k\) analysis is constructed. In contrast, the \(\text{Sub}_{\text{Pvar}}\) domain of this paper was chosen carefully to simplify widening and ease the construction and presentation of the analysis. Also, in terms of implementation and practicality, the analysis presented in this paper applies confluence in a novel way to split the analysis into small, simple and efficient units.

Very recently, in a fourth proposal [4], a sharing and freeness analysis is formulated in terms of a transition system which reduces a set of abstract equations to an abstract solved...
form. Sharing is represented in a sharing group fashion with variables enriched with linearity and freeness information by an annotation mapping. Depth-\(k\) sharing, groundness, freeness and compoundness information can be represented to a bounded depth by virtue of the abstract equations. To be precise, the domain is formulated as a set of equivalence classes of abstract equations. The domain is similar in spirit to \(T_{type_{Pvar}}\): sharing groups and freeness sets record the aliasing between variable place markers that are introduced in the structural component of the domain. One distinction, however, is the \(S_{ulb}^k\) domain is engineered to be a poset whereas the abstract equations of \[4\] lead to a preorder which makes termination more subtle. A second difference is that in this paper the frequently used operations like projection \(\tau^{C_{SF}}\), lub \(\sqcup^{C_{SF}}\), and widening \(\sqcap_k^{C_{SF}}\) are designed to be efficient. In \[4\], however, these frequently used operators are not discussed. A third distinction is the emphasis this paper puts on modularity. Modularity follows by using confluence to split an analysis into its constituent parts. Modularity is advantageous since it simplifies both the presentation and the implementation. On a software engineering perspective it also permits an existing \(Share_{Pvar} \times Free_{Pvar}\) unification code, for instance \[23\], to be plugged into the implementation to reuse valuable code. To be fair, however, the analysis of \[4\] does trace linearity and capture compoundness and definite sharing between the variables of the abstract equations. This might be useful. Of these three differences, linearity is probably the most significant, and the confluence approach (and in particular lemma 4.3) can extended to accommodate linearity. This is not difficult. The principal reason why linearity has not been directly addressed in this paper is that it is simply not (yet) clear that the extra complication is worthwhile if structure can be traced to depth-\(k\). Future work will focus on implementation and benchmarking (will is a non-trivial study within itself) to suggest suitable \(k\) and to determine whether or not complexity of tracing linearity is worthwhile.

6 Conclusions

A powerful and formally justified analysis has been presented for inferring definite groundness, freeness and compoundness, and possible sharing to a bounded depth \(k\). The analysis exploits confluence to split the analysis into its constituent components and introduce modularity. Modularity simplifies the implementation, aids the presentation, and leads to a well-structured proof of correctness. High precision follows from the combination of domains. The analysis propagates groundness with the accuracy of sharing groups and yet infers sharing and freeness to a precision which exceeds that of a normal freeness analysis. The analysis is significant because it can underpin many optimisations in logic programming. It is likely to be particularly useful in the detection of independent and-parallelism.

Acknowledgements

Thanks are due to Manuel Hermenegildo, Francisco Bueno and Bob Kemp for useful discussions on sharing and freeness. This work was supported, in part, by ESPRIT project (6707) “ParForce”.

References


7 Appendix

Proof 7.1 (for lemma 3.1) Let $\phi^{\text{CSF}}, \varphi^{\text{CSF}} \in \text{Type}_\text{Pvar}$.

1. Suppose $\phi^S = \emptyset$ and $\varphi^S = \emptyset$. Thus $\phi^{\text{CSF}} = \varphi^{\text{CSF}}$ and $\gamma^{\text{CSF}}(\phi^{\text{CSF}}) = \gamma^{\text{CSF}}(\varphi^{\text{CSF}})$.

2. Suppose $\phi^S = \emptyset$ and $\varphi^S \neq \emptyset$. Then $\gamma^{\text{CSF}}(\phi^{\text{CSF}}) = \emptyset$ whereas $\gamma(\varphi^{\text{CSF}}) \neq \emptyset$.

3. Suppose $\phi^S \neq \emptyset$ and $\varphi^S = \emptyset$. Thus $\gamma^{\text{CSF}}(\phi^{\text{CSF}}) \neq \emptyset$ whereas $\gamma(\varphi^{\text{CSF}}) = \emptyset$.

4. Suppose $\phi^S \neq \emptyset$ and $\varphi^S \neq \emptyset$.

   (a) Suppose $\phi^c \neq \varphi^c$.

      i. Suppose $U\phi \in \phi^S$ and $U\varphi \notin \varphi^S$. Thus $\phi \in \gamma^{\text{CSF}}(\phi^{\text{CSF}})$ such that $\phi \notin \gamma^{\text{CSF}}(\varphi^{\text{CSF}})$.

      ii. Suppose $U\varphi \in \varphi^S$ and $U\phi \notin \phi^S$. Thus $\varphi \in \gamma^{\text{CSF}}(\varphi^{\text{CSF}})$ such that $\varphi \notin \gamma^{\text{CSF}}(\phi^{\text{CSF}})$.

      iii. Suppose $\phi^S = \varphi^S$.

         A. Suppose $u^* \in \phi^S$ and $u^* \notin \varphi^S$. Thus $\varphi \in \gamma^{\text{CSF}}(\varphi^{\text{CSF}})$ such that $\varphi \notin \gamma^{\text{CSF}}(\phi^{\text{CSF}})$.

         B. Suppose $u^* \in \varphi^S$ and $u^* \notin \phi^S$. Thus $\phi \in \gamma^{\text{CSF}}(\phi^{\text{CSF}})$ such that $\phi \notin \gamma^{\text{CSF}}(\varphi^{\text{CSF}})$.

         C. Suppose $\phi^S = \varphi^S$. Thus $\phi^{\text{CSF}} = \varphi^{\text{CSF}}$ and $\gamma^{\text{CSF}}(\phi^{\text{CSF}}) = \gamma(\varphi^{\text{CSF}})$.

   (b) Suppose $\phi^c = \varphi^c$.

      i. Suppose $\varphi^c(\bar{u})[s] = f(t_1, \ldots, t_n)$ and $\varphi^c(\bar{u'})[s'] = f'(t'_1, \ldots, t'_n)$ with $f \neq f'$ or $n \neq n'$. Thus $\gamma^{\text{CSF}}(\phi^{\text{CSF}}) \cap \gamma(\varphi^{\text{CSF}}) = \emptyset$.

      ii. Suppose $\varphi^c(\bar{u})[s] \in \text{Uvar}$ and $\varphi^c(\bar{u'})[s'] = f(t_1, \ldots, t_n)$.

      iii. Suppose $\phi^c(\bar{u})[s] \in \text{Uvar}$. Thus $\varphi \in \gamma^{\text{CSF}}(\phi^{\text{CSF}})$ such that $\varphi \notin \gamma^{\text{CSF}}(\phi^{\text{CSF}})$.

Proof 7.2 (for lemma 3.2) For brevity put $\varphi^{\text{CSF}} = \bigcup^{\text{CSF}}(\phi^{\text{CSF}})$.

1. Suppose $\Phi^{\text{CSF}} = \emptyset$.

   (a) To show $\phi^{\text{CSF}} \subseteq^{\text{CSF}} \varphi^{\text{CSF}}$ for all $\phi^{\text{CSF}} \in \Phi^{\text{CSF}}$. Immediate since $\Phi^{\text{CSF}} = \emptyset$.

   (b) Suppose $\phi^{\text{CSF}} \subseteq^{\text{CSF}} \varphi^{\text{CSF}}$ for all $\phi^{\text{CSF}} \in \Phi^{\text{CSF}}$. To show $\phi^{\text{CSF}} \subseteq^{\text{CSF}} \varphi^{\text{CSF}}$. Since $\varphi^{\text{CSF}} = \bigcup^{\text{CSF}} \Phi^{\text{CSF}}(\Phi^{\text{CSF}}) = 0$ and thus $\phi^{\text{CSF}} \subseteq^{\text{CSF}} \varphi^{\text{CSF}}$.

2. Suppose $\Phi^{\text{CSF}} \neq \emptyset$.

   (a) To show $\phi^{\text{CSF}} \subseteq^{\text{CSF}} \varphi^{\text{CSF}}$ for all $\phi^{\text{CSF}} \in \Phi^{\text{CSF}}$. Let $\phi \in \Phi^{\text{CSF}}$ such that $\phi \in \text{Pvar}$, $\phi^d \supseteq \phi^c \uparrow \text{Pvar}$. Note that $\phi^c \subseteq \phi^d$ as required.

      i. Since $\varphi^c \subseteq \phi^d$ there exists $\phi^d$ such that $\varphi^d \circ \phi^c \uparrow \text{Pvar} = \phi \uparrow \text{Pvar}$. Let $u \in \Lambda \text{var}$. To show $\text{occ}_{\text{cod}}(\varphi^c)(u, \phi^d) \in \varphi^S$. Now $\{v \in \text{cod}(\phi^d) \mid u \in \var\phi(v)[s]\}$ $= \text{occ}_{\text{cod}}(\varphi^c)(u, \phi^d) \in \varphi^S$. Hence $\phi^d(\{v \in \text{cod}(\phi^d) \mid u \in \var\phi(v)[s]\})) \in \varphi^S$ where $\phi^d \circ \phi^c \uparrow \text{Pvar} = \phi^d$. Therefore $\{v \in \text{cod}(\phi^c) \mid u \in \var\phi(v)[s]\} = \text{occ}_{\text{cod}}(\varphi^c)(u, \phi^d) \in \varphi^S$. 26
ii. Suppose \( u^* \in \varphi^F \). But \( u^* \in \phi^\mathcal{S} \) and thus \( \phi(u)[s] \in U \text{var} \) as required.

Hence \( \phi \in \gamma_{\text{Pvar}}(\varphi_{\mathcal{S} \mathcal{F}}) \).

(b) Suppose \( \phi_{\mathcal{S} \mathcal{F}} \sqsubseteq \phi_{\mathcal{S} \mathcal{F}} \) for all \( \phi_{\mathcal{S} \mathcal{F}} \in \Phi_{\mathcal{S} \mathcal{F}} \) and \( \psi_{\mathcal{S} \mathcal{F}} \sqsubseteq \psi_{\mathcal{S} \mathcal{F}} \). To show \( \phi_{\mathcal{S} \mathcal{F}} = \psi_{\mathcal{S} \mathcal{F}} \).

Hence \( \theta^c \sqsubseteq \phi^c \). But since \( \gamma_{\text{Pvar}}(\theta^c_{\mathcal{S} \mathcal{F}}) \sqsubseteq \gamma_{\text{Pvar}}(\phi^c_{\mathcal{S} \mathcal{F}}) \), \( \phi^c \sqsubseteq \theta^c \). Therefore \( \theta^c = \phi^c \). Thus \( \theta^c \sqsubseteq \phi^c \) and \( \phi^c \sqsubseteq \theta^c \) since \( \gamma_{\text{Pvar}}(\theta^c_{\mathcal{S} \mathcal{F}}) \sqsubseteq \gamma_{\text{Pvar}}(\phi^c_{\mathcal{S} \mathcal{F}}) \).

Proof 7.3 (for corollary 3.1) Put \( \cap_{\mathcal{S} \mathcal{F}}(\emptyset) = \cap_{\mathcal{S} \mathcal{F}}. \) Since \( Type_{\text{Pvar}}(\cap_{\mathcal{S} \mathcal{F}}) \) is a poset and \( \sqcup_{\mathcal{S} \mathcal{F}} \) is defined for every subset of \( Type_{\text{Pvar}} \), then, by theorem 3 of chapter 5 of [1], it follows that \( Type_{\text{Pvar}}(\cap_{\mathcal{S} \mathcal{F}}) \) is a complete lattice.

Proof 7.4 (for lemma 3.4) Suppose \( \Phi \subseteq \gamma_{\text{Pvar}}(\phi_{\mathcal{S} \mathcal{F}}) \). For conciseness let \( \psi_{\mathcal{S} \mathcal{F}} \) and \( \phi_{\mathcal{S} \mathcal{F}} \) respectively denote the right-hand-sides of definition 3.12 and lemma 3.4. Observe that \( \psi_{\mathcal{S} \mathcal{F}} \sqsubseteq \phi_{\mathcal{S} \mathcal{F}} \). Note that \( \phi \sqsubseteq \phi^c \) since \( \phi^c = \gamma_{\text{Pvar}}(\Phi) \). Moreover if \( \phi^c = \phi^c \) then \( \phi^c \sqsubseteq \phi^c \) and \( \phi^c \sqsubseteq \phi^c \). Thus \( \phi_{\mathcal{S} \mathcal{F}} \sqsubseteq \phi_{\mathcal{S} \mathcal{F}} \).
ii. Suppose \( v \in \text{var}(\phi(u)) \) and \( v \not\in \text{var}(\phi(t)) \). Since \( \varphi \in \text{mgv}\{\phi(u) = \phi(t)\} \), \( v \in \text{dom}(\varphi) \) or \( v \in \text{cod}(\varphi) \). Since \( \varphi(v) = v \), \( v \not\in \text{dom}(\varphi) \) and thus \( v \in \text{cod}(\varphi) \). Thus \( v \in \text{var}(\varphi \circ \phi(u)) \) and therefore \( v \in \text{var}(\varphi \circ \phi(t)) \). Since \( v \not\in \text{var}(\phi(t)) \), there exists \( w \in \text{var}(\phi(t)) \) such that \( v \in \text{var}(\varphi(w)) \). Thus \( v \in \text{var}(\varphi \circ \phi(t)) \) and since \( v \not\in \text{cod}(\varphi \circ \phi) \), \( v = t \) so that \( \phi(t) = v \) which is a contradiction.

iii. Suppose \( v \not\in \text{var}(\phi(u)) \) and \( v \in \text{var}(\phi(t)) \). Like case 1(a)ii.

iv. Suppose \( v \in \text{var}(\phi(u)) \) and \( v \in \text{var}(\phi(t)) \). Since \( \varphi(v) = v \) and \( v \not\in \text{cod}(\varphi \circ \phi) \), \( v \not\in \text{cod}(\varphi) \). Thus \( \phi(v) = v \) and therefore \( \varphi \in \text{mgv}\{v = \phi(t)\} \) with \( v \in \text{var}(\phi(t)) \) which is a contradiction.

(b) Suppose \( v \in \text{dom}(\varphi \circ \phi) \). Since \( v \not\in \text{cod}(\varphi \circ \phi) \), \( \text{occ}_\text{Pvar}(v, \varphi \circ \phi) = \emptyset \in \text{mgv}^S(u, t, \varphi^S) \).

2. Suppose \( v \in \text{cod}(\varphi \circ \phi) \setminus \text{var}(\varphi \circ \phi(u)) \). Let \( w \in \text{Pvar} \). Suppose \( v \in \text{var}(\varphi \circ \phi(u)) \) but \( v \not\in \text{var}(\phi(w)) \). Then \( v \in \text{cod}(\varphi) \) and hence \( v \in \text{var}(\varphi \circ \phi(w)) \) which is a contradiction. Suppose \( v \in \text{var}(\phi(w)) \) but \( v \not\in \text{var}(\varphi \circ \phi(w)) \). Thus \( v \in \text{dom}(\varphi) \) and \( v \not\in \text{cod}(\varphi) \) so that \( v \not\in \text{cod}(\varphi \circ \phi) \) which is a contradiction. Hence \( \text{occ}_\text{Pvar}(v, \varphi \circ \phi) = \text{occ}_\text{Pvar}(v, \phi) \in \varphi^S \).

3. Suppose \( v \in \text{cod}(\varphi \circ \phi) \cap \text{var}(\varphi \circ \phi(u)) \). Note that \( \text{occ}_\text{Pvar}(v, \varphi \circ \phi) = \bigcup_{v \in \text{var}(\varphi(u))} \text{occ}_\text{Pvar}(w, \phi) \).

(a) Suppose \( u \in \varphi^S \) with \( \phi(t) = u_\text{e} \).

i. Suppose \( \varphi = \{v_u \mapsto \phi(t)\} \). Since \( v \in \text{var}(\varphi \circ \phi(u)) \), \( v \in \text{var}(\phi(t)) \). Thus \( \{w \mid v \in \varphi(u)\} = \{u_v, v\} \). Hence \( \text{occ}_\text{Pvar}(v, \varphi \circ \phi) \in \text{rel}(u, \phi^S) \cap \text{rel}(t, \phi^S) \subseteq \text{mgv}^S(u, t, \phi^S) \).

ii. Suppose \( \varphi = \{v_1 \mapsto v_u\} \) with \( \phi(t) = v_1 \). Since \( v \in \text{var}(\varphi \circ \phi(u)) \), \( v = v_u \). Thus \( \{w \mid v \in \varphi(u)\} = \{v_u, v_1\} \). Hence \( \text{occ}_\text{Pvar}(v, \varphi \circ \phi) \in \text{rel}(u, \phi^S) \cap \text{rel}(t, \phi^S) \subseteq \text{mgv}^S(u, t, \phi^S) \).

(b) Suppose \( t \in \varphi^S \). Like case 3a.

c. Suppose \( u \not\in \varphi^S \) and \( t \not\in \varphi^S \). There exists \( W_u \subseteq \text{var}(\varphi(u)) \) and \( W_t \subseteq \text{var}(\phi(t)) \) such that \( \text{occ}_\text{Pvar}(v, \varphi \circ \phi) = \bigcup_{w \in W_u \cup W_t} \text{occ}_\text{Pvar}(w, \phi) \). Since \( v \in \text{var}(\varphi \circ \phi(u)) \), \( W_u \not= \emptyset \) and thus \( W_t \not= \emptyset \). Thus \( \text{occ}_\text{Pvar}(v, \varphi \circ \phi) \in \text{rel}(u, \phi^S) \cap \text{rel}(t, \phi^S) \subseteq \text{mgv}^S(u, t, \phi^S) \).

Proof 7.8 (for lemma 4.2) Let \( \phi \in \gamma_\text{Pvar}(\phi^F) \), \( \varphi \in \text{mgv}\{\phi(u) = \phi(t)\} \), \{u\} \cup \text{var}(t) \subseteq \text{Pvar} \) and \( u \not\in \text{var}(t) \). To show \( \text{fr}_\text{Pvar}(\varphi \circ \phi) \supseteq \text{mgv}^F(u, t, \phi^S) \). Let \( v \in \text{mgv}^F(u, t, \phi^S) \).

1. Suppose \( u \in \varphi^F \) and \( t \in \varphi^F \) where \( \phi(u) = u_\text{e} \) and \( \phi(t) = v_\text{e} \).

(a) If \( \varphi = \{v_u \mapsto v_\text{e}\} \) then \( v \in \text{fr}_\text{Pvar}(\varphi \circ \phi) \) since \( v \in \text{fr}_\text{Pvar}(\phi) \).

(b) If \( \varphi = \{v_\text{e} \mapsto v_u\} \) then \( v \in \text{fr}_\text{Pvar}(\varphi \circ \phi) \) since \( v \in \text{fr}_\text{Pvar}(\phi) \).

2. Suppose \( u \not\in \varphi^F \).

(a) Suppose \( \varphi = \{v_u \mapsto \phi(t)\} \) where \( \phi(u) = v_u \). Since \( v \not\in \text{var}(\text{rel}(u, \phi^S)) \), \( v_u \not\in \text{var}(\phi(v)) \). Hence \( \varphi \circ \phi(v) = \phi(v) \). Thus, since \( v \in \text{fr}_\text{Pvar}(\phi) \), \( v \in \text{fr}_\text{Pvar}(\varphi \circ \phi) \).
(b) Suppose $\varphi = \{ u \mapsto v_u \}$ where $\varphi(u) = v_u$ and $\varphi(t) = v_t$. Thus $v \in \text{fr}_P(\varphi \circ \phi)$ since $v \in \text{fr}_P(\phi)$.

3. Suppose $t \in \phi^\mathcal{F}$. Like case 2.

4. Suppose $u \notin \phi^\mathcal{F}$ and $t \notin \phi^\mathcal{F}$. Since $v \notin \text{var}(\text{rel}(u, \phi^\mathcal{F}))$ and $v \notin \text{var}(\text{rel}(t, \phi^\mathcal{F}))$, $\text{var}(\varphi(v)) \cap \text{var}(\varphi(u)) = \emptyset$ and $\text{var}(\varphi(v)) \cap \text{var}(\varphi(t)) = \emptyset$ and hence $\varphi \circ \phi(v) = \phi(v) \in \text{fr}_P(\phi)$.

**Proof 7.9 (for theorem 4.1)** Let $\phi \in \gamma_{P\text{var}}(\psi_{\mathcal{SF}}^\phi)$, $\varphi \in \text{mgu}(\phi(E))$ and $\text{mgu}(\psi_{\mathcal{SF}}(E, \phi^\mathcal{F}, \mu^\mathcal{SF}))$ with $\text{var}(E) \subseteq \text{Pvar}$. By induction on the steps of $\text{mgu}^\mathcal{SF}$ and by lemmas 4.1 and 4.2, there exists $\theta \in \text{mgu}(\phi(E))$ such that $\theta \circ \phi \in \gamma_{P\text{var}}(\psi_{\mathcal{SF}}^\phi)$. But $\theta \approx \varphi$ [17] and thus $\theta \circ \phi \approx \varphi \circ \phi$. Hence $\varphi \circ \phi \in \gamma_{P\text{var}}(\psi_{\mathcal{SF}}^\phi)$.

**Proof 7.10 (for lemma 4.3)** Proof by induction. Suppose $\phi \uparrow \text{Pvar} = \phi^O \cup \phi^C \uparrow \text{Pvar}$; $\phi^B \in \text{mgu}(\phi(E))$; $\psi^B = \text{sub}_P(\phi^B)$; $\psi^B = \{ u \rightarrow t_i \}_{i=1}^{n} \text{ so that } \psi^B = \psi^B \; \gamma_{\text{cod}^{\psi_{\mathcal{SF}}^\phi \circ \phi^C \uparrow \text{Pvar}}}^{\phi^S_{n+1}}(\delta_{n+1})$. Put $\delta^S_{n+1} = \text{mgu}^{\psi_{\mathcal{SF}}^\phi}(a_{n+1}, t_{n+1}, \delta^S_n)$ and $\delta^O_{n+1} = \text{mgu}^{\psi_{\mathcal{SF}}^\phi}(a_{n+1}, t_{n+1}, \delta^O_n)$. To show $\forall \text{cod}^{\psi_{\mathcal{SF}}^\phi} \psi_{\mathcal{SF}}^\phi \circ \phi^C \uparrow \text{Pvar} \setminus \{ \psi^B_{n+1} \circ \phi^C \uparrow \text{Pvar} \} \setminus (\psi^B_{n+1} \circ \phi^C \uparrow \text{Pvar}) \subseteq \delta^O_{n+1}$. Let $w \in \text{Uvar}$.

1. Suppose $w = u_{n+1}^s$. Now $\delta_{n+1} = \text{mgu}(\{ u_{n+1}^s \rightarrow t_{n+1} \}) \circ \{ u_{n+1}^s \rightarrow t_{n+1} \}$ and thus $w \in \text{dom}(\delta_{n+1})$ so that $\text{occ} = \text{cod}^{\psi_{\mathcal{SF}}^\phi \circ \phi^C \uparrow \text{Pvar}}(w, \delta_{n+1}) = \emptyset \in \delta_{n+1}$. If $w \in \phi^O_{n+1}$ then $w \in \text{var}(\text{rel}(u_{n+1}^s, \phi_{n+1}))$ and thus $w \notin \phi^O_{n+1}$.

2. Suppose $w \in \text{var}(t_{n+1})$.

   (a) Suppose $u_{n+1}^s \in \phi^O_{n+1}$. Thus $\delta_{n+1} = \text{mgu}(\{ u_{n+1}^s \rightarrow t_{n+1} \}) \circ \{ u_{n+1}^s \rightarrow t_{n+1} \}$. Hence $\delta_{n+1} = \text{mgu}(\{ u_{n+1}^s \rightarrow t_{n+1} \}) \circ \{ u_{n+1}^s \rightarrow t_{n+1} \}$.

   (b) Suppose $u_{n+1}^s \notin \phi^O_{n+1}$. Note that $\delta_{n+1} = \text{mgu}(\{ \delta_{n}(u_{n+1}^s) = t_{n+1} \}) \circ \delta_{n}$ since $\text{dom}(\delta_{n}) \cap t_{n+1} = \emptyset$. Let $\chi = \text{mgu}(\{ \delta_{n}(u_{n+1}^s) = t_{n+1} \})$. Now $\text{occ} = \text{cod}^{\psi_{\mathcal{SF}}^\phi \circ \phi^C \uparrow \text{Pvar}}(w, \chi \circ \delta_{n}) = \emptyset \in \text{Uvar}(\chi \circ \delta_{n})$ and $\text{occ} = \text{cod}^{\psi_{\mathcal{SF}}^\phi \circ \phi^C \uparrow \text{Pvar}}(w, \delta_{n+1}) = \emptyset \in \text{Uvar}(\delta_{n+1})$. Since $\delta_{n+1} \subseteq \text{Uvar}(t_{n+1})$ there exists $Y_u \subseteq \text{var}(\delta_{n+1})$ and $Y_t \subseteq \text{var}(t_{n+1})$ such that $Y_u \cup Y_t = Y_{t+1}$. By induction $\chi(t_{n+1}) = 1$, by lemma 2.1, $Y_u \subseteq \{ y \}$.
i. If $Y_u = \emptyset$ then $w \not\in \var(y(\delta_u(w_{n+1}))))$ and thus $w \not\in \var(y(t_{n+1}))$ so that $Y_t = \emptyset$.

Thus $\text{occ}_{\psi_n^{u+1}, \phi^c}(\var(\psi_n^{u+1}, \phi^c)) \subseteq \emptyset \subseteq \emptyset^S$.

ii. If $Y_u \neq \emptyset$, then $Y_t \neq \emptyset$. Because $u_{n+1} \in \text{dom}(\psi^n)$, $u_{n+1} \in \text{dom}(\psi^n \circ \phi^c)$, and thus $\text{occ}_{\psi_n^{u+1}, \phi^c}(\psi_n^{u+1}, \phi^c) \subseteq \text{rel}(u_{n+1}, \phi^c)$. Now $y_u \in \var(y(u_{n+1})) \subseteq \{u_{n+1}\} \cup \text{dom}(\delta_u)$. But $\{u_{n+1}\} \cup \text{dom}(\delta_u) \cap \var(t_{n+1}) = \emptyset$ and thus $\var(t_{n+1}) = \var(y(t_{n+1}))$. Thus $y_u \not\in \var(y(t_{n+1}))$. Hence $\text{occ}_{\psi_n^{u+1}, \phi^c}(\var(\psi_n^{u+1}, \phi^c)) \subseteq \emptyset \subseteq \emptyset^S$.

If $y_t \in Y_t$, $y_t \not\in \text{dom}(\delta_t)$ and thus $\text{occ}_{\psi_n^{u+1}, \phi^c}(\var(\psi_n^{u+1}, \phi^c)) \subseteq \emptyset \subseteq \emptyset^S$.

Note that $w \not\in \emptyset^T$.

3. Suppose $w \not\in \{u_{n+1}\} \cup \var(t_{n+1})$. Let $\zeta = \text{mgv}(\{\delta_u(u_{n+1}) = \delta_u(t_{n+1})\})$ so that $\delta_u = \zeta \circ \delta_u$. Hence $\text{occ}_{\psi_n^{u+1}, \phi^c}(\var(\psi_n^{u+1}, \phi^c)) \subseteq \emptyset \subseteq \emptyset^S$.

Since $w \not\in \{u_{n+1}\} \cup \var(t_{n+1})$ and $w \not\in \text{dom}(\delta_t)$, $\zeta(w) = w$ and hence $\text{occ}_{\psi_n^{u+1}, \phi^c}(\var(\psi_n^{u+1}, \phi^c)) \subseteq \emptyset \subseteq \emptyset^S$.

Because $\text{rel}(u_{n+1}, \phi^c)$ and $\text{occ}_{\psi_n^{u+1}, \phi^c}(\var(\psi_n^{u+1}, \phi^c)) \subseteq \emptyset \subseteq \emptyset^S$.

Thus $\var(\psi_n^{u+1}, \phi^c) \subseteq \emptyset^S$ and $\var(\psi_n^{u+1}, \phi^c) \subseteq \emptyset^S$.

Proof 7.11 (for theorem 4.2) Let $\phi \in \gamma_{\psi_n^{u+1}, \phi^c}(\var(\psi_n^{u+1}, \phi^c))$ and $\text{mgv}(\phi(\psi^n))$, $\phi \in \text{mgv}(\phi(E))$ and $\text{mgv}(\phi(E))$, $\phi \in \psi_n^{u+1}, \phi^c$ with $\var(\psi^n) \subseteq \emptyset \subseteq \emptyset^S$.

Thus $\text{occ}_{\psi_n^{u+1}, \phi^c}(\var(\psi_n^{u+1}, \phi^c)) \subseteq \emptyset \subseteq \emptyset^S$.

Since $\phi \in \gamma_{\psi_n^{u+1}, \phi^c}(\var(\psi_n^{u+1}, \phi^c))$ there exists $\phi^c$ such that $\phi = \phi^c \circ \phi^c$. Now $\phi \circ \phi = \text{mgv}(\phi(\psi^n))$, $\phi \in \phi(E) \cup \phi(\psi^n)$ and $\text{mgv}(\phi(\psi^n))$, $\phi \in \phi(E) \cup \phi(\psi^n)$ and $\text{mgv}(\phi(\psi^n))$.

Thus $\phi = \phi^c \circ \phi^c$. Now $\phi \circ \phi = \text{mgv}(\phi(\psi^n))$, $\phi \in \phi(E) \cup \phi(\psi^n)$ and $\text{mgv}(\phi(\psi^n))$.
Proof 7.12 (for theorem 4.3) Let $\Phi \subseteq \gamma^{\text{CSF}}(\phi^{\text{CSF}})$, $\Psi \subseteq \gamma^{\text{CSF}}(\psi^{\text{CSF}})$ and $\theta \in \text{unify}(a, \Phi, b, \Psi)$. Thus $\theta = (\varphi \circ \phi) \uparrow \text{Pvar}$ where $\varphi \in \text{mgw}(\{\phi(a) = Y(\psi(b))\})$, $\phi \in \Phi$ and $\psi \in \Psi$. Observe that $\varphi \in \text{mgw}(\{\phi(a) = Y(\psi(T^{-1}(Y(b))))\})$ and thus putting $\sigma = \phi \cup (Y \circ \psi \circ T^{-1})$, $\varphi \in \text{mgw}(\sigma\{a = Y(b)\})$. Note that $\phi \in \gamma^{\text{CSF}}_{\text{Pvar}}(\phi^{\text{CSF}})$ and $\psi \circ Y^{-1} \in \gamma^{\text{CSF}}_{\text{Pvar}}(Y(\psi^{\text{CSF}}))$ and hence $Y \circ \psi \circ Y^{-1} \in \gamma^{\text{CSF}}_{\text{Pvar}}(Y(\psi^{\text{CSF}}))$. Since $\text{var}(\phi) \cap \text{var}(Y \circ \psi \circ Y^{-1}) = \emptyset$, $\sigma \in \gamma^{\text{CSF}}_{\text{Pvar} \cup \text{Pvar}}(\phi^{\text{CSF}} \cup Y(\psi^{\text{CSF}}))$. Thus, by theorem 4.2, since $\text{var}(a) \cup \text{var}(Y(b)) \subseteq \text{Pvar} \cup Y(\text{Pvar})$, $\varphi \circ \sigma \in \gamma^{\text{CSF}}(\text{mgw}(\{a = Y(b)\}), \phi^{\text{CSF}} \cup Y(\psi^{\text{CSF}}))$. But $(\varphi \circ \sigma) \uparrow \text{Pvar} = (\varphi \circ \sigma \uparrow \text{Pvar}) \uparrow \text{Pvar} = (\varphi \circ \phi) \uparrow \text{Pvar}$ and $(\varphi \circ \sigma) \uparrow \text{Pvar} \in \gamma^{\text{CSF}}_{\text{Pvar} \cup \text{Pvar}}(\text{mgw}(\{a = Y(b)\}), \phi^{\text{CSF}} \cup Y(\psi^{\text{CSF}})) \gamma^{\text{CSF}} \text{Pvar}$ and therefore $\theta \in \gamma^{\text{CSF}}_{\text{Pvar}}(\text{unify}^{\text{CSF}}(a, \phi^{\text{CSF}}, b, \psi^{\text{CSF}}))$. 

31