Finite-Tree Analysis for Constraint Logic-Based Languages: The Complete Unabridged Version ⋆

Roberto Bagnara[a], Roberta Gori[b], Patricia M. Hill[c], Enea Zaffanella[a]

[a]*Department of Mathematics, University of Parma, Italy*

[b]*Department of Computer Science, University of Pisa, Italy*

[c]*School of Computing, University of Leeds, UK*

Abstract

Logic languages based on the theory of rational, possibly infinite, trees have much appeal in that rational trees allow for faster unification (due to the safe omission of the occurs-check) and increased expressivity (cyclic terms can provide very efficient representations of grammars and other useful objects). Unfortunately, the use of infinite rational trees has problems. For instance, many of the built-in and library predicates are ill-defined for such trees and need to be supplemented by run-time checks whose cost may be significant. Moreover, some widely-used program analysis and manipulation techniques are correct only for those parts of programs working over finite trees. It is thus important to obtain, automatically, a knowledge of the program variables (the *finite variables*) that, at the program points of interest, will always be bound to finite terms. For these reasons, we propose here a new data-flow analysis, based on abstract interpretation, that captures such information. We present a parametric domain where a simple component for recording finite variables is coupled, in the style of the *open product* construction of Cortesi et al., with a generic domain (the parameter of the construction) providing sharing information. The sharing domain is abstractly specified so as to guarantee the correctness of the combined domain and the generality of the approach. This finite-tree analysis domain is further enhanced by coupling it with a domain of Boolean functions, called *finite-tree dependencies*, that precisely captures how the finiteness of some variables influences the finiteness of other variables. We also summarize our experimental results showing how finite-tree analysis, enhanced with finite-tree dependencies, is a practical means of obtaining precise finiteness information.

Key words: static analysis, abstract interpretation, rational unification, occurs-check

1 Introduction

The intended computation domain of most logic-based languages [1] includes the algebra (or structure) of *finite trees*. Other (constraint) logic-based languages, such as Prolog II and its successors [1,2], SICStus Prolog [3], and Oz [4], refer to a computation domain of *rational trees*. [2] A rational tree is a possibly infinite tree with a finite number of distinct subtrees and where each node has a finite number of immediate descendants. These properties ensure that rational trees, even though infinite in the sense that they admit paths of infinite length, can be finitely represented. One possible representation makes use of connected, rooted, directed and possibly cyclic graphs where nodes are labeled with variable and function symbols as is the case of finite trees.

Applications of rational trees in logic programming include graphics [6], parser generation and grammar manipulation [1,7], and computing with finite-state automata [1]. Rational trees also constitute the basis of the abstract domain of *rigid type graphs*, which is used for type analysis of logic programs [8–10]. Other applications are described in [11] and [12]. Very recently, Manuel Carro has described a nice application of rational trees where they are used to represent imperative programs within interpreters. Taking a continuation-passing style approach, each instruction is coupled with a data structure representing the remaining part of the program to be executed so that sequences of instructions for realizing (backward) jumps, iterations and recursive calls give rise to cyclic structures in the form of rational trees. Compared to a naive interpreter for the same language, this threaded interpreter is faster and uses less memory, at the cost of a simple preliminary "compilation pass" to generate the rational tree representation for the program [13].

Going from Prolog to CLP, in [14] K. Mukai has combined constraints on rational trees and record structures, while the logic-based language *Oz* allows constraints over rational and feature trees [4]. The expressive power of rational trees is put to use, for instance, in several areas of natural language processing.

⋆ This work has been partly supported by MURST projects "Automatic Program Certification by Abstract Interpretation", "Abstract Interpretation, Type Systems and Control-Flow Analysis", and "Constraint Based Verification of Reactive Systems." Some of this work was done during visits of the fourth author to Leeds, funded by EPSRC under grant M05645.

Email addresses: `bagnara@cs.unipr.it` (Roberto Bagnara), `gori@di.unipi.it` (Roberta Gori), `hill@comp.leeds.ac.uk` (Patricia M. Hill), `zaffanella@cs.unipr.it` (Enea Zaffanella).

[1] That is, ordinary logic languages, (concurrent) constraint logic languages, functional logic languages and variations of the above.

[2] Support for rational trees is also provided as an option by the YAP Prolog system [5].

Rational trees are used in implementations of the HPSG formalism (Head-driven Phrase Structure Grammar) [15], in the ALE system (Attribute Logic Engine) [16], and in the ProFIT system (Prolog with Features, Inheritance and Templates) [17].

While rational trees allow for increased expressivity, they also come equipped with a surprising number of problems. As we will see, some of these problems are so serious that rational trees must be used in a very controlled way, disallowing them in any context where they are "dangerous." This, in turn, causes a secondary problem: in order to disallow rational trees in selected contexts one must first detect them, an operation that may be expensive.

The first thing to be aware of is that almost any semantics-based program manipulation technique developed in the field of logic programming —whether it be an analysis, a transformation, or an optimization— assumes a computation domain of *finite trees*. Some of these techniques might work with rational trees but their correctness has only been proved in the case of finite trees. Others are clearly inapplicable. Let us consider a very simple Prolog program:

```
list([]).
list([_|T]) :- list(T).
```

Most automatic and semi-automatic tools for proving program termination [3] and for complexity analysis [4] agree on the fact that `list/1` will terminate when invoked with a ground argument. Consider now the query

```
?- X = [a|X], list(X).
```

and note that, after the execution of the first rational unification, the variable `X` will be bound to a rational term containing no variables, i.e., the predicate `list/1` will be invoked with `X` ground. However, if such a query is given to, say, SICStus Prolog, then the only way to get the prompt back is by interrupting the program. The problem stems from the fact that the analysis techniques employed by these tools are only sound for finite trees: as soon as they are applied to a system where the creation of cyclic terms is possible, their results are inapplicable. The situation can be improved by combining these termination and/or complexity analyses with a finiteness analysis providing the precondition for the applicability of the other techniques.

The implementation of built-in predicates is another problematic issue. Indeed, it is widely acknowledged that, for the implementation of a system that provides real support for rational trees, the biggest effort concerns proper han-

[3] Such as TerminWeb [18,19], TermiLog [20], cTI [21], and LPTP [22,23].
[4] Systems like GAIA [24], CASLOG [25], and the Ciao-Prolog preprocessor [26].

dling of built-ins. Of course, the meaning of 'proper' depends on the actual built-in. Built-ins such as `copy_term/2` and `==/2` maintain a clear semantics when passing from finite to rational trees. For others, like `sort/2`, the extension can be questionable:[5] failing, raising an exception, answering `Y = [a]` (if duplicates are deleted) and answering `Y = [a|Y]` (if duplicates are kept) can all be argued to be "the right reaction" to the query

```
?- X = [a|X], sort(X, Y).
```

Other built-ins do not tolerate infinite trees in some argument positions. A good implementation should check for finiteness of the corresponding arguments and make sure "the right thing" —failing or raising an appropriate exception— always happens. However, such behavior appears to be uncommon. A small experiment we conducted on six Prolog implementations with queries like

```
?- X = 1+X, Y is X.
?- X = [97|X], name(Y, X).
?- X = [X|X], Y =.. [f|X].
```

resulted in infinite loops, memory exhaustion and/or system thrashing, segmentation faults or other fatal errors. One of the implementations tested, SICStus Prolog, is a professional one and implements run-time checks to avoid most cases where built-ins can have catastrophic effects.[6] The remaining systems are a bit more than research prototypes, but will clearly have to do the same if they evolve to the stage of production tools. Again, a data-flow analysis aimed at the detection of those variables that are definitely bound to finite terms could be used to avoid a (possibly significant) fraction of the useless run-time checks. Note that what has been said for built-in predicates applies to libraries as well. Even though it may be argued that it is enough for programmers to know that they should not use a particular library predicate with infinite terms, it is clear that the use of a "safe" library, including automatic checks ensuring that such a predicate is never called with an illegal argument, will result in a robuster system. With the appropriate data-flow analyses, safe libraries do not have to be inefficient libraries.

Another serious problem is the following: the standard term ordering dictated by ISO Prolog [27] cannot be extended to rational trees [M. Carlsson, Personal communication, October 2000]. Consider the rational trees defined by `A = f(B, a)` and `B = f(A, b)`. Clearly, `A == B` does not hold. Since the stan-

[5] Even though `sort/2` is not required to be a built-in by the ISO Prolog standard, it is offered as such by several implementations.

[6] SICStus 3.11 still loops on `?- X = [97|X], name(Y, X).`

dard term ordering is total, we must have either `A @< B` or `B @< A`. Assume `A @< B`. Then `f(A, b) @< f(B, a)`, since the ordering of terms having the same principal functor is inherited by the ordering of subterms considered in a left-to-right fashion. Thus `B @< A` must hold, which is a contradiction. A dual contradiction is obtained by assuming `B @< A`. As a consequence, applying any Prolog term-ordering predicate to terms where one or both of them is infinite may cause inconsistent results, giving rise to bugs that are exceptionally difficult to diagnose. For this reason, any system that extends ISO Prolog with rational trees ought to detect such situations and make sure they are not ignored (e.g., by throwing an exception or aborting execution with a meaningful message). However, predicates such as the term-ordering ones are likely to be called a significant number of times, since they are often used to maintain structures implementing ordered collections of terms. This is another instance of the efficiency issue mentioned above.

Still on efficiency, it is worth noting that even for built-ins whose definition on rational trees is not problematic, there is often a performance penalty in catering for the possibility of infinite trees. Thus, for such predicates, which include rational unification provided by `=/2`, a compile-time knowledge of term finiteness can be beneficial. For instance, rational-tree implementations of the built-ins `ground/1`, `term_variables/2`, `copy_term/2`, `subsumes/2`, `variant/2` and `numbervars/3` need more expensive marking techniques to ensure they do not enter an infinite loop. With finiteness information it is possible to avoid this overhead.

In this paper, we present a parametric abstract domain for finite-tree analysis, denoted by $H \times P$. This domain combines a simple component H (written with the initial of *Herbrand* and called the *finiteness* component) recording the set of definitely finite variables, with a generic domain P (the parameter of the construction) providing sharing information. The term "sharing information" is to be understood in its broader meaning, which includes variable aliasing, groundness, linearity, freeness and any other kind of information that can improve the precision on these components, such as explicit structural information. Several domain combinations and abstract operators, characterized by different precision/complexity trade-offs, have been proposed to capture these properties (see [28,29] for an account of some of them). By giving a generic specification for this parameter component, in the style of the *open product* construct proposed in [30], it is possible to define and establish the correctness of abstract operators on the finite-tree domain independently from any particular domain for sharing analysis.

The information encoded by H is *attribute independent* [31], which means that each variable is considered in isolation. What this lacks is information about how finiteness of one variable affects the finiteness of other variables. This kind of information, usually called *relational information*, is not captured at all by H and is only partially captured by the composite domain $H \times P$. Moreover, $H \times P$ is designed to capture the "negative" aspect of term-finiteness, that is, the circumstances under which finiteness can be lost. However, term-finiteness has also a "positive" aspect: there are cases where a variable is granted to be bound to a finite term and this knowledge can be propagated to other variables. Guarantees of finiteness are provided by several built-ins like `unify_with_occurs_check/2`, `var/1`, `name/2`, all the arithmetic predicates, besides those explicitly provided to test for term-finiteness such as the `acyclic_term/1` predicate of SICStus Prolog. For these reasons $H \times P$ is coupled with a domain of Boolean functions that precisely captures how the finiteness of some variables influences the finiteness of other variables. This domain of *finite-tree dependencies* provides relational information that is important for the precision of the overall finite-tree analysis. It also combines obvious similarities, interesting differences and somewhat unexpected connections with classical domains for *groundness dependencies*. Finite-tree and groundness dependencies are similar in that they both track *covering* information (a term s covers t if all the variables in t also occur in s) and share several abstract operations. However, they are different because covering does not tell the whole story. Suppose x and y are free variables before either the unification $x = f(y)$ or the unification $x = f(x, y)$ are executed. In both cases, x will be ground if and only if y will be so. However, when $x = f(y)$ is the performed unification, this equivalence will also carry over to finiteness. In contrast, when the unification is $x = f(x, y)$, x will never be finite and will be totally independent, as far as finiteness is concerned, from y. Among the unexpected connections is the fact that finite-tree dependencies can improve the groundness information obtained by the usual approaches to groundness analysis.

The paper is structured as follows. The required notations and preliminary concepts are given in Section 2. The concrete domain for the analysis is presented in Section 3. The finite-tree domain is then introduced in Section 4: Section 4.1 provides the specification of the parameter domain P; Section 4.2 defines some computable operators that extract, from substitutions in rational solved form, properties of the denoted rational trees; Section 4.3 defines the abstraction function for the finiteness component H; Section 4.4 defines the abstract unification operator for $H \times P$. Section 5 introduces the use of Boolean functions for tracking finite-tree dependencies, whereas Section 6 illustrates the interaction between groundness and finite-tree dependencies. Our experimental results are presented in Section 7. We conclude the main body of the paper in Section 8.

Appendix A specifies the sharing domain SFL defined in [32,33] as a possible instance of the parameter P. All the results are then proved in Appendix B.

This paper is a combined and improved version of [34] and [35].

2 Preliminaries

2.1 Infinite Terms and Substitutions

The cardinality of a set S is denoted by $\# S$; $\wp(S)$ is the powerset of S, whereas $\wp_{\mathrm{f}}(S)$ is the set of all the *finite* subsets of S. Let Sig denote a possibly infinite set of function symbols, ranked over the set of natural numbers. It is assumed that Sig contains at least one function symbol having rank 0 and one having rank greater than 0. Let Vars denote a denumerable set of variables disjoint from Sig and Terms denote the free algebra of all (possibly infinite) terms in the signature Sig having variables in Vars. Thus a term can be seen as an ordered labeled tree, possibly having some infinite paths and possibly containing variables: every non-leaf node is labeled with a function symbol in Sig with a rank matching the number of the node's immediate descendants, whereas every leaf is labeled by either a variable in Vars or a function symbol in Sig having rank 0 (a constant).

If $t \in \mathrm{Terms}$ then $\mathrm{vars}(t)$ and $\mathrm{mvars}(t)$ denote the set and the multiset of variables occurring in t, respectively. We will also write $\mathrm{vars}(o)$ to denote the set of variables occurring in an arbitrary syntactic object o.

Suppose $s, t \in \mathrm{Terms}$: s and t are *independent* if $\mathrm{vars}(s) \cap \mathrm{vars}(t) = \varnothing$; t is said to be *ground* if $\mathrm{vars}(t) = \varnothing$; t is *free* if $t \in \mathrm{Vars}$; if $y \in \mathrm{vars}(t)$ occurs exactly once in t, then we say that variable *y occurs linearly in t*, more briefly written using the predication $\mathrm{occ\_lin}(y, t)$; t is *linear* if we have $\mathrm{occ\_lin}(y, t)$ for all $y \in \mathrm{vars}(t)$; finally, t is a *finite term* (or *Herbrand term*) if it contains a finite number of occurrences of function symbols. The sets of all ground, linear and finite terms are denoted by GTerms, LTerms and HTerms, respectively. As we have specified that Sig contains function symbols of rank 0 and rank greater than 0, $\mathrm{GTerms} \cap \mathrm{HTerms} \neq \varnothing$ and $\mathrm{GTerms} \setminus \mathrm{HTerms} \neq \varnothing$.

A *substitution* is a total function $\sigma \colon \mathrm{Vars} \to \mathrm{HTerms}$ that is the identity almost everywhere; in other words, the *domain* of σ,

$$\mathrm{dom}(\sigma) \stackrel{\mathrm{def}}{=} \bigl\{\, x \in \mathrm{Vars} \bigm| \sigma(x) \neq x \,\bigr\},$$

is finite. Given a substitution $\sigma \colon \mathrm{Vars} \to \mathrm{HTerms}$, we overload the symbol 'σ' so as to denote also the function $\sigma \colon \mathrm{HTerms} \to \mathrm{HTerms}$ defined as follows, for each term $t \in \mathrm{HTerms}$:

$$\sigma(t) \stackrel{\mathrm{def}}{=} \begin{cases} t, & \text{if } t \text{ is a constant symbol;} \\ \sigma(t), & \text{if } t \in \mathrm{Vars;} \\ f\bigl(\sigma(t_1), \ldots, \sigma(t_n)\bigr), & \text{if } t = f(t_1, \ldots, t_n). \end{cases}$$

If $t \in \mathrm{HTerms}$, we write $t\sigma$ to denote $\sigma(t)$ and $t\sigma\tau$ to denote $(t\sigma)\tau$.

If $x \in \mathrm{Vars}$ and $t \in \mathrm{HTerms} \setminus \{x\}$, then $x \mapsto t$ is called a *binding*. The set of all bindings is denoted by Bind. Substitutions are denoted by the set of their bindings, thus a substitution σ is identified with the (finite) set

$$\left\{ x \mapsto x\sigma \;\middle|\; x \in \mathrm{dom}(\sigma) \right\}.$$

We denote by $\mathrm{vars}(\sigma)$ the set of variables occurring in the bindings of σ.

A substitution is said to be *circular* if, for $n > 1$, it has the form

$$\{x_1 \mapsto x_2, \ldots, x_{n-1} \mapsto x_n, x_n \mapsto x_1\},$$

where $x_1, \ldots, x_n$ are distinct variables. A substitution is in *rational solved form* if it has no circular subset. The set of all substitutions in rational solved form is denoted by RSubst.

The composition of substitutions is defined in the usual way. Thus $\tau \circ \sigma$ is the substitution such that, for all terms $t \in \mathrm{HTerms}$,

$$(\tau \circ \sigma)(t) = \tau\bigl(\sigma(t)\bigr) = t\sigma\tau$$

and has the formulation

$$\tau \circ \sigma = \left\{ x \mapsto x\sigma\tau \;\middle|\; x \in \mathrm{dom}(\sigma) \cup \mathrm{dom}(\tau), x \neq x\sigma\tau \right\}.$$

As usual, σ^0 denotes the identity function (i.e., the empty substitution) and, when $i > 0$, σ^i denotes the substitution $(\sigma \circ \sigma^{i-1})$.

Consider an infinite sequence of terms $t_0, t_1, t_2, \ldots$ with $t_i \in \mathrm{HTerms}$ for each $i \in \mathbb{N}$. Suppose there exists $t \in \mathrm{Terms}$ such that, for each $n \in \mathbb{N}$, there exists $m_0 \in \mathbb{N}$ such that, for each $m \in \mathbb{N}$ with $m \geq m_0$, the trees corresponding to the terms t and t_m coincide up to the first n levels. Then we say that *the sequence* $t_0, t_1, t_2, \ldots$ *converges to* t and we write $t = \lim_{i \to \infty} t_i$ [36].

For each $\sigma \in \mathrm{RSubst}$ and $t \in \mathrm{HTerms}$, the sequence of finite terms

$$\sigma^0(t), \sigma^1(t), \sigma^2(t), \ldots$$

converges [36,37]. Therefore, the function $\mathrm{rt} \colon \mathrm{HTerms} \times \mathrm{RSubst} \to \mathrm{Terms}$ such that

$$\mathrm{rt}(t, \sigma) \stackrel{\mathrm{def}}{=} \lim_{i \to \infty} \sigma^i(t)$$

is well defined.

An *equation* is a statement of the form $s = t$ where $s, t \in \mathrm{HTerms}$. Eqs denotes the set of all equations. As usual, a system of equations (i.e., a conjuction of elements in Eqs) will be denoted by a subset of Eqs. A substitution σ may be regarded as a finite set of equations, that is, as the set $\{\, x = t \mid x \mapsto t \in \sigma \,\}$. A set of equations e is in *rational solved form* if $\bigl\{\, s \mapsto t \bigm| (s = t) \in e \,\bigr\} \in \mathrm{RSubst}$. In the rest of the paper, we will often write a substitution $\sigma \in \mathrm{RSubst}$ to denote a set of equations in rational solved form (and vice versa).

Languages such as Prolog II, SICStus and Oz are based on $\mathcal{RT}$, the theory of rational trees [1,38]. This is a syntactic equality theory (i.e., a theory where the function symbols are uninterpreted), augmented with a *uniqueness axiom* for each substitution in rational solved form. Informally speaking these axioms state that, after assigning a ground rational tree to each non-domain variable, the substitution uniquely defines a ground rational tree for each of its domain variables. Thus, any set of equations in rational solved form is, by definition, satisfiable in $\mathcal{RT}$. Equality theories and, in particular, $\mathcal{RT}$ are presented in more detail in Appendix B.1.1. Note that being in rational solved form is a very weak property. Indeed, unification algorithms returning a set of equations in rational solved form are allowed to be much more "lazy" than one would usually expect. For instance, $\{x = y, y = z\}$ and $\bigl\{x = f(y), y = f(x)\bigr\}$ are in rational solved form. We refer the interested reader to [39–41] for details on the subject.

Given a set of equations $e \in \wp_{\mathrm{f}}(\mathrm{Eqs})$ that is satisfiable in $\mathcal{RT}$, a substitution $\sigma \in \mathrm{RSubst}$ is called a *solution for e in $\mathcal{RT}$* if $\mathcal{RT} \vdash \forall(\sigma \rightarrow e)$, i.e., if theory $\mathcal{RT}$ entails the first order formula $\forall(\sigma \rightarrow e)$. If in addition $\mathrm{vars}(\sigma) \subseteq \mathrm{vars}(e)$, then σ is said to be a *relevant* solution for e. Finally, σ is a *most general solution for e in $\mathcal{RT}$* if $\mathcal{RT} \vdash \forall(\sigma \leftrightarrow e)$. In this paper, the set of all the relevant most general solutions for e in $\mathcal{RT}$ will be denoted by $\mathrm{mgs}(e)$.

In the sequel, in order to model the constraint accumulation process of logic-based languages, we will need to characterize those sets of equations that are stronger than (that can be obtained by adding equations to) a given set of equations.

Definition 1 **($\downarrow(\cdot)$)** *The function* $\downarrow(\cdot)\colon \mathrm{RSubst} \rightarrow \wp(\mathrm{RSubst})$ *is defined, for each* $\sigma \in \mathrm{RSubst}$*, by*

$$\downarrow \sigma \stackrel{\mathrm{def}}{=} \bigl\{\, \tau \in \mathrm{RSubst} \bigm| \exists \sigma' \in \mathrm{RSubst} \mathrel{.} \tau \in \mathrm{mgs}(\sigma \cup \sigma') \,\bigr\}.$$

The next result shows that $\downarrow(\cdot)$ corresponds to the closure by entailment in $\mathcal{RT}$.

Proposition 2 *Let* $\sigma \in \mathrm{RSubst}$*. Then*

$$\downarrow \sigma = \left\{ \tau \in \mathrm{RSubst} \;\middle|\; \mathcal{RT} \vdash \forall(\tau \to \sigma) \right\}.$$

2.3 Boolean Functions

Boolean functions have already been extensively used for data-flow analysis of logic-based languages. An important class of these functions used for tracking groundness dependencies is Pos [42]. This domain was introduced in [43] under the name *Prop* and further refined and studied in [44,45].

The formal definition of the set of Boolean functions over a finite set of variables is based on the notion of Boolean valuation. Note that in all the following definitions we abuse notation by assuming that the finite set of variables V is clear from context, so as to avoid using it as a suffix everywhere.

Definition 3 (Boolean valuation.) *Let* $V \in \wp_{\mathrm{f}}(\mathrm{Vars})$ *and* $\mathrm{Bool} \stackrel{\mathrm{def}}{=} \{0, 1\}$*. The set of* Boolean valuations over V *is given by*

$$\mathrm{Bval} \stackrel{\mathrm{def}}{=} V \to \mathrm{Bool}.$$

For each $a \in \mathrm{Bval}$*, each* $x \in V$*, and each* $c \in \mathrm{Bool}$ *the valuation* $a[c/x] \in \mathrm{Bval}$ *is given, for each* $y \in V$*, by*

$$a[c/x](y) \stackrel{\mathrm{def}}{=} \begin{cases} c, & \text{if } x = y; \\ a(y), & \text{otherwise.} \end{cases}$$

If $X = \{x_1, \ldots, x_k\} \subseteq V$*, then* $a[c/X]$ *denotes* $a[c/x_1] \cdots [c/x_k]$*.*

The distinguished elements $\mathbf{0}, \mathbf{1} \in \mathrm{Bval}$ are given by

$$\mathbf{0} \stackrel{\mathrm{def}}{=} \lambda x \in V \,.\, 0,$$
$$\mathbf{1} \stackrel{\mathrm{def}}{=} \lambda x \in V \,.\, 1.$$

Definition 4 (Boolean function.) *The set of* Boolean functions over V *is*

$$\mathrm{Bfun} \stackrel{\mathrm{def}}{=} \mathrm{Bval} \to \mathrm{Bool}.$$

Bfun *is partially ordered by the relation* $\models$ *where, for each* $\phi, \psi \in \mathrm{Bfun}$*,*

$$\phi \models \psi \quad \stackrel{\mathrm{def}}{\iff} \quad \bigl(\forall a \in \mathrm{Bval} : \phi(a) = 1 \implies \psi(a) = 1\bigr).$$

For $\phi \in \mathrm{Bfun}$, $x \in V$, and $c \in \mathrm{Bool}$, the Boolean function $\phi[c/x] \in \mathrm{Bfun}$ is given, for each $a \in \mathrm{Bval}$, by

$$\phi[c/x](a) \stackrel{\text{def}}{=} \phi\bigl(a[c/x]\bigr).$$

When $X \subseteq V$, $\phi[c/X]$ is defined in the expected way. If $\phi \in \mathrm{Bfun}$ and $x, y \in V$ the function $\phi[y/x] \in \mathrm{Bfun}$ is given, for each $a \in \mathrm{Bval}$, by

$$\phi[y/x](a) \stackrel{\text{def}}{=} \phi\Bigl(a\bigl[a(y)/x\bigr]\Bigr).$$

Boolean functions are constructed from the elementary functions corresponding to variables and by means of the usual logical connectives. Thus, for each $x \in V$, x also denotes the Boolean function ϕ such that, for each $a \in \mathrm{Bval}$, $\phi(a) = 1$ if and only if $a(x) = 1$; for $\phi \in \mathrm{Bfun}$, we write $\neg\phi$ to denote the function ψ such that, for each $a \in \mathrm{Bval}$, $\psi(a) = 1$ if and only if $\phi(a) = 0$; for $\phi_1, \phi_2 \in \mathrm{Bfun}$, we write $\phi_1 \vee \phi_2$ to denote the function ϕ such that, for each $a \in \mathrm{Bval}$, $\phi(a) = 0$ if and only if both $\phi_1(a) = 0$ and $\phi_2(a) = 0$. A variable is restricted away using Schröder's elimination principle [46]:

$$\exists x \,.\, \phi \stackrel{\text{def}}{=} \phi[1/x] \vee \phi[0/x].$$

Note that existential quantification is both monotonic and extensive on Bfun. The other Boolean connectives and quantifiers are handled similarly. The distinguished elements $\bot, \top \in \mathrm{Bfun}$ are the functions defined by

$$\begin{aligned} \bot &\stackrel{\text{def}}{=} \lambda a \in \mathrm{Bval} \,.\, 0, \\ \top &\stackrel{\text{def}}{=} \lambda a \in \mathrm{Bval} \,.\, 1. \end{aligned}$$

For notational convenience, when $X \subseteq V$, we inductively define

$$\bigwedge X \stackrel{\text{def}}{=} \begin{cases} \top, & \text{if } X = \varnothing; \\ x \wedge \bigwedge\bigl(X \setminus \{x\}\bigr), & \text{if } x \in X. \end{cases}$$

$\mathrm{Pos} \subset \mathrm{Bfun}$ consists precisely of those functions assuming the true value under the *everything-is-true* assignment, i.e.,

$$\mathrm{Pos} \stackrel{\text{def}}{=} \Bigl\{\, \phi \in \mathrm{Bfun} \Bigm| \phi(\mathbf{1}) = 1 \,\Bigr\}.$$

For each $\phi \in \mathrm{Bfun}$, the *positive part of* ϕ, denoted $\mathrm{pos}(\phi)$, is the strongest Pos formula that is entailed by ϕ. Formally,

$$\mathrm{pos}(\phi) \stackrel{\text{def}}{=} \phi \vee \bigwedge V.$$

For each $\phi \in \mathrm{Bfun}$, the set of *variables necessarily true for* ϕ and the set of *variables necessarily false for* ϕ are given, respectively, by

$$\mathrm{true}(\phi) \stackrel{\mathrm{def}}{=} \bigl\{\, x \in V \bigm| \forall a \in \mathrm{Bval} : \phi(a) = 1 \implies a(x) = 1 \,\bigr\},$$
$$\mathrm{false}(\phi) \stackrel{\mathrm{def}}{=} \bigl\{\, x \in V \bigm| \forall a \in \mathrm{Bval} : \phi(a) = 1 \implies a(x) = 0 \,\bigr\}.$$

3 The Concrete Domain

A knowledge of the basic concepts of abstract interpretation theory [47,48] is assumed. In this paper, the concrete domain consists of pairs of the form (Σ, V), where V is a finite set of *variables of interest* [44] and Σ is a (possibly infinite) set of substitutions in rational solved form.

Definition 5 (The concrete domain.) *Let* $\mathcal{D}^\flat \stackrel{\mathrm{def}}{=} \wp(\mathrm{RSubst}) \times \wp_{\mathrm{f}}(\mathrm{Vars})$. *If* $(\Sigma, V) \in \mathcal{D}^\flat$, *then* (Σ, V) *represents the (possibly infinite) set of first-order formulas* $\bigl\{\, \exists\Delta \,.\, \sigma \bigm| \sigma \in \Sigma, \Delta = \mathrm{vars}(\sigma) \setminus V \,\bigr\}$ *where* σ *is interpreted as the logical conjunction of the equations corresponding to its bindings.*

The operation of projecting $x \in \mathrm{Vars}$ *away from* $(\Sigma, V) \in \mathcal{D}^\flat$ *is defined as follows:*

$$\exists\, x \,.\, (\Sigma, V) \stackrel{\mathrm{def}}{=} \left\{\, \sigma' \in \mathrm{RSubst} \;\middle|\; \begin{array}{l} \sigma \in \Sigma, \overline{V} = \mathrm{Vars} \setminus V, \\ \mathcal{RT} \vdash \forall\bigl(\exists\overline{V} \,.\, (\sigma' \leftrightarrow \exists x \,.\, \sigma)\bigr) \end{array} \right\}.$$

Concrete domains for constraint languages would be similar. If the analyzed language allows the use of constraints on various domains to restrict the values of the variable leaves of rational trees, the corresponding concrete domain would have one or more extra components to account for the constraints (see [49] for an example).

The concrete element $\Bigl(\bigl\{\{x \mapsto f(y)\}\bigr\}, \{x, y\}\Bigr)$ expresses a dependency between x and y. In contrast, $\Bigl(\bigl\{\{x \mapsto f(y)\}\bigr\}, \{x\}\Bigr)$ only constrains x. The same concept can be expressed by saying that in the first case the variable name 'y' matters, but it does not in the second case. Thus, the set of variables of interest is crucial for defining the meaning of the concrete and abstract descriptions. Despite this, always specifying the set of variables of interest would significantly clutter the presentation. Moreover, most of the needed functions on concrete and abstract descriptions preserve the set of variables of interest. For these reasons, we assume the existence of a set $\mathrm{VI} \in \wp_{\mathrm{f}}(\mathrm{Vars})$ that contains, at each stage of the analysis, the current variables of interest.[7] As a consequence, when the context makes it clear, we will write $\Sigma \in \mathcal{D}^\flat$ as a shorthand for $(\Sigma, \mathrm{VI}) \in \mathcal{D}^\flat$.

4 An Abstract Domain for Finite-Tree Analysis

Finite-tree analysis applies to logic-based languages computing over a domain of rational trees where cyclic structures are allowed. In contrast, analyses aimed at occurs-check reduction [50,51] apply to programs that are meant to compute on a domain of finite trees only, but have to be executed over systems that are either designed for rational trees or intended just for the finite trees but omit the occurs-check for efficiency reasons. Despite their different objectives, finite-tree and occurs-check analyses have much in common: in both cases, it is important to detect all program points where cyclic structures can be generated.

Note however that, when performing occurs-check reduction, one can take advantage of the following invariant: all data structures generated so far are finite. This property is maintained by transforming the program so as to force finiteness whenever it is possible that a cyclic structure could have been built.[8] In contrast, a finite-tree analysis has to deal with the more general case when some of the data structures computed so far may be cyclic. It is therefore natural to consider an abstract domain made up of two components. The first one simply represents the set of variables that are guaranteed not to be bound to infinite terms. We will denote this *finiteness component* by H (from *Herbrand*).

Definition 6 (The finiteness component.) *The* finiteness component *is the set* $H \stackrel{\mathrm{def}}{=} \wp(\mathrm{VI})$ *partially ordered by reverse subset inclusion.*

The second component of the finite-tree domain should maintain any kind of information that may be useful for computing finiteness information.

It is well-known that sharing information as a whole, therefore including possible variable aliasing, definite linearity, and definite freeness, has a crucial role

[7] This parallels what happens in the efficient implementation of data-flow analyzers. In fact, almost all the abstract domains currently in use do not need to represent explicitly the set of variables of interest. In contrast, this set is maintained externally and in a unique copy, typically by the fixpoint computation engine.

[8] Such a requirement is typically obtained by replacing the unification with a call to the standard predicate `unify_with_occurs_check/2`. As an alternative, in some systems based on rational trees it is possible to insert, after each problematic unification, a finiteness test for the generated term.

in occurs-check reduction so that, as observed before, it can be exploited for finite-tree analysis too. Thus, a first choice for the second component of the finite-tree domain would be to consider one of the standard combinations of sharing, freeness and linearity as defined, e.g., in [28,29,52,53]. However, this would tie our specification to a particular sharing analysis domain, whereas the overall approach is inherently more general. For this reason, we will define a finite-tree analysis based on the abstract domain schema $H \times P$, where the generic *sharing component* P is a parameter of the abstract domain construction. This approach can be formalized as an application of the *open product* operator [30], where the interaction between the H and P components is modeled by defining a suite of generic query operators: thus, the overall accuracy of the finite-tree analysis will heavily depend on the accuracy with which any specific instance of the parameter P is able to answer these queries.

4.1 The parameter Component P

Elements of P can encode any kind of information. We only require that substitutions that are equivalent in the theory $\mathcal{RT}$ are identified in P.

Definition 7 (The parameter component.) *The parameter component P is an abstract domain related to the concrete domain $\mathcal{D}^\flat$ by means of the* concretization function $\gamma_P \colon P \to \wp(\mathrm{RSubst})$ *such that, for all $p \in P$,*

$$\Bigl(\sigma \in \gamma_P(p) \wedge \bigl(\mathcal{RT} \vdash \forall(\sigma \leftrightarrow \tau)\bigr)\Bigr) \implies \tau \in \gamma_P(p).$$

The interface between H and P is provided by a set of abstract operators that satisfy suitable correctness criteria. We only specify those that are useful for defining abstract unification and projection on the combined domain $H \times P$. Other operations needed for a full description of the analysis, such as renaming and upper bound, are very simple and, as usual, do not pose any problems.

Definition 8 (Abstract operators on P.) *Let $s, t \in$ HTerms be finite terms. For each $p \in P$, we specify the following predicates:*

s and t are independent in p *if and only if* $\mathrm{ind}_p \colon \mathrm{HTerms}^2 \to \mathrm{Bool}$ *holds for (s, t), where*

$$\mathrm{ind}_p(s,t) \implies \forall \sigma \in \gamma_P(p) : \mathrm{vars}\bigl(\mathrm{rt}(s,\sigma)\bigr) \cap \mathrm{vars}\bigl(\mathrm{rt}(t,\sigma)\bigr) = \varnothing;$$

s and t share linearly in p *if and only if* $\mathrm{share\_lin}_p \colon \mathrm{HTerms}^2 \to \mathrm{Bool}$ *holds for (s, t), where*

$$\mathrm{share\_lin}_p(s,t) \implies \forall \sigma \in \gamma_P(p) :$$

$$\forall y \in \mathrm{vars}\bigl(\mathrm{rt}(s,\sigma)\bigr) \cap \mathrm{vars}\bigl(\mathrm{rt}(t,\sigma)\bigr) : \\ \mathrm{occ\_lin}\bigl(y, \mathrm{rt}(s,\sigma)\bigr) \wedge \mathrm{occ\_lin}\bigl(y, \mathrm{rt}(t,\sigma)\bigr);$$

t is ground in p *if and only if* $\mathrm{ground}_p \colon \mathrm{HTerms} \to \mathrm{Bool}$ *holds for t, where*

$$\mathrm{ground}_p(t) \implies \forall \sigma \in \gamma_P(p) : \mathrm{rt}(t,\sigma) \in \mathrm{GTerms};$$

t is ground-or-free in p *if and only if* $\mathrm{gfree}_p \colon \mathrm{HTerms} \to \mathrm{Bool}$ *holds for t, where*

$$\mathrm{gfree}_p(t) \implies \forall \sigma \in \gamma_P(p) : \mathrm{rt}(t,\sigma) \in \mathrm{GTerms} \vee \mathrm{rt}(t,\sigma) \in \mathrm{Vars};$$

s is linear in p *if and only if* $\mathrm{lin}_p \colon \mathrm{HTerms} \to \mathrm{Bool}$ *holds for s, where*

$$\mathrm{lin}_p(s) \implies \forall \sigma \in \gamma_P(p) : \mathrm{rt}(s,\sigma) \in \mathrm{LTerms};$$

s and t are or-linear in p *if and only if* $\mathrm{or\_lin}_p \colon \mathrm{HTerms}^2 \to \mathrm{Bool}$ *holds for (s,t), where*

$$\mathrm{or\_lin}_p(s,t) \implies \forall \sigma \in \gamma_P(p) : \mathrm{rt}(s,\sigma) \in \mathrm{LTerms} \vee \mathrm{rt}(t,\sigma) \in \mathrm{LTerms};$$

For each $p \in P$, the following functions compute subsets of the set of variables of interest:

the function $\mathrm{share\_same\_var}_p \colon \mathrm{HTerms} \times \mathrm{HTerms} \to \wp(\mathrm{VI})$ *returns a set of variables that may share with the given terms via the same variable. For each pair of terms $s, t \in \mathrm{HTerms}$,*

$$\mathrm{share\_same\_var}_p(s,t) \supseteq \left\{ y \in \mathrm{VI} \;\middle|\; \begin{array}{l} \exists \sigma \in \gamma_P(p) \;. \\ \quad \exists z \in \mathrm{vars}\bigl(\mathrm{rt}(y,\sigma)\bigr) \;. \\ \qquad z \in \mathrm{vars}\bigl(\mathrm{rt}(s,\sigma)\bigr) \cap \mathrm{vars}\bigl(\mathrm{rt}(t,\sigma)\bigr) \end{array} \right\};$$

the function $\mathrm{share\_with}_p \colon \mathrm{HTerms} \to \wp(\mathrm{VI})$ *yields a set of variables that may share with the given term. For each $t \in \mathrm{HTerms}$,*

$$\mathrm{share\_with}_p(t) \stackrel{\mathrm{def}}{=} \bigl\{\, y \in \mathrm{VI} \bigm| y \in \mathrm{share\_same\_var}_p(y,t) \,\bigr\}.$$

The function $\mathrm{amgu}_P \colon P \times \mathrm{Bind} \to P$ *correctly captures the effects of a binding on an element of P. For each $(x \mapsto t) \in \mathrm{Bind}$ and $p \in P$, let*

$$p' \stackrel{\mathrm{def}}{=} \mathrm{amgu}_P(p, x \mapsto t);$$

for all $\sigma \in \gamma_P(p)$*, if* $\tau \in \operatorname{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$*, then* $\tau \in \gamma_P(p')$*.*

The function $\mathrm{proj}_P \colon P \times \mathrm{VI} \to P$ *correctly captures the operation of projecting away a variable from an element of* P*. For each* $x \in \mathrm{VI}$*,* $p \in P$ *and* $\sigma \in \gamma_P(p)$*, if* $\tau \in \exists x \mathrel{.} \{\sigma\}$*, then* $\tau \in \gamma_P\bigl(\mathrm{proj}_P(p, x)\bigr)$*.*

As it will be shown in Appendix A, some of these generic operators can be directly mapped to the corresponding abstract operators defined for well-known sharing analysis domains. However, the specification given in Definition 8, besides being more general than a particular implementation, also allows for a modular approach when proving correctness results.

4.2 Operators on Substitutions in Rational Solved Form

There are cases when an analysis tries to capture properties of the particular substitutions computed by a specific (ordinary or rational) unification algorithm. This is the case, for example, when the analysis needs to track structure sharing for the purpose of compile-time garbage collection, or provide upper bounds on the amount of memory needed to perform a given computation. More often the interest is on properties of the (finite or rational) trees that are denoted by such substitutions.

When the concrete domain is based on the theory of finite trees, idempotent substitutions provide a finitely computable *strong normal form* for domain elements, meaning that different substitutions describe different sets of finite trees (as usual, this is modulo the possible renaming of variables). In contrast, when working on a concrete domain based on the theory of rational trees, substitutions in rational solved form, while being finitely computable, no longer satisfy this property: there can be an infinite set of substitutions in rational solved form all describing the same set of rational trees (i.e., the same element in the "intended" semantics). For instance, the substitutions

$$\sigma_n = \Bigl\{x \mapsto \overbrace{f(\cdots f(}^{n} x)\cdots)\Bigr\}$$

for $n = 1, 2, \ldots,$ all map the variable x to the same rational tree (which is usually denoted by f^ω).

Ideally, a strong normal form for the set of rational trees described by a substitution $\sigma \in \mathrm{RSubst}$ can be obtained by computing the limit function

$$\sigma^\infty \stackrel{\mathrm{def}}{=} \lambda t \in \mathrm{HTerms} \mathrel{.} \mathrm{rt}(t, \sigma),$$

obtained by fixing the substitution parameter of 'rt'. The problem is that, in general, σ^∞ is not a substitution: while having a finite domain, its "bindings"

$x \mapsto \lim_{i\to\infty} \sigma^i(x)$ can map a domain variable x to an infinite rational term. This poses a non-trivial problem when trying to define a "good" abstraction function, since it would be really desirable for this function to map any two equivalent concrete elements to the same abstract element. Of course, it is important that the properties under investigation are exactly captured, so as to avoid any unnecessary precision loss. Pursuing this goal requires an ability to observe properties of (infinite) rational trees while just dealing with one of their finite representations. This is not always an easy task since even simple properties can be "hidden" when using non-idempotent substitutions. For instance, when σ^∞ maps variable x to an infinite and ground rational tree (i.e., when $\mathrm{rt}(x, \sigma) \in \mathrm{GTerms} \setminus \mathrm{HTerms}$), all of its finite representations in RSubst (i.e., all the $\tau \in \mathrm{RSubst}$ such that $\mathcal{RT} \models \forall(\sigma \leftrightarrow \tau)$) will map the variable x into a finite term that is not ground. These are the motivations behind the introduction of the following computable operators on substitutions.

The groundness operator 'gvars' captures the set of variables that are mapped to ground rational trees by rt. We define it by means of the *occurrence operator* 'occ'. This was introduced in [54] as a replacement for the sharing-group operator 'sg' of [55]. In [54] the 'occ' operator is used to define a new abstraction function for set-sharing analysis that, differently from the classical ones [56,55], maps equivalent substitutions in rational solved form to the same abstract element.

Definition 9 (Occurrence and groundness operators.) *For each $n \in \mathbb{N}$, the* occurrence function $\mathrm{occ}_n \colon \mathrm{RSubst} \times \mathrm{Vars} \to \wp_\mathrm{f}(\mathrm{Vars})$ *is defined, for each $\sigma \in \mathrm{RSubst}$ and each $v \in \mathrm{Vars}$, by*

$$\mathrm{occ}_n(\sigma, v) \stackrel{\mathrm{def}}{=} \begin{cases} \{v\} \setminus \mathrm{dom}(\sigma), & \text{if } n = 0; \\ \bigl\{\, y \in \mathrm{Vars} \bigm| \mathrm{vars}(y\sigma) \cap \mathrm{occ}_{n-1}(\sigma, v) \neq \varnothing \,\bigr\}, & \text{if } n > 0. \end{cases}$$

The occurrence operator $\mathrm{occ} \colon \mathrm{RSubst} \times \mathrm{Vars} \to \wp_\mathrm{f}(\mathrm{Vars})$ *is given, for each $\sigma \in \mathrm{RSubst}$ and $v \in \mathrm{Vars}$, by $\mathrm{occ}(\sigma, v) \stackrel{\mathrm{def}}{=} \mathrm{occ}_\ell(\sigma, v)$, where $\ell = \#\sigma$.*

The groundness operator $\mathrm{gvars} \colon \mathrm{RSubst} \to \wp_\mathrm{f}(\mathrm{Vars})$ *is given, for each substitution $\sigma \in \mathrm{RSubst}$, by*

$$\mathrm{gvars}(\sigma) \stackrel{\mathrm{def}}{=} \bigl\{\, y \in \mathrm{dom}(\sigma) \bigm| \forall v \in \mathrm{vars}(\sigma) : y \notin \mathrm{occ}(\sigma, v) \,\bigr\}.$$

Example 10 *Let*

$$\sigma = \bigl\{ x \mapsto f(y, z), y \mapsto g(z, x), z \mapsto f(a) \bigr\}.$$

Then $\mathrm{gvars}(\sigma) = \{x, y, z\}$, although $\mathrm{vars}(x\sigma^i) \neq \varnothing$ and $\mathrm{vars}(y\sigma^i) \neq \varnothing$, for all $0 \leq i < \infty$.

The *finiteness operator* is defined, like 'occ', by means of a fixpoint construction.

Definition 11 (Finiteness functions.) *For each* $n \in \mathbb{N}$*, the* finiteness function $\mathrm{hvars}_n \colon \mathrm{RSubst} \to \wp(\mathrm{Vars})$ *is defined, for each* $\sigma \in \mathrm{RSubst}$*, by*

$$\mathrm{hvars}_0(\sigma) \stackrel{\mathrm{def}}{=} \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$$

and, for $n > 0$*, by*

$$\mathrm{hvars}_n(\sigma) \stackrel{\mathrm{def}}{=} \mathrm{hvars}_{n-1}(\sigma) \cup \Bigl\{\, y \in \mathrm{dom}(\sigma) \Bigm| \mathrm{vars}(y\sigma) \subseteq \mathrm{hvars}_{n-1}(\sigma) \,\Bigr\}.$$

For each $\sigma \in \mathrm{RSubst}$ and each $i \geq 0$, we have $\mathrm{hvars}_i(\sigma) \subseteq \mathrm{hvars}_{i+1}(\sigma)$ and also that $\mathrm{Vars} \setminus \mathrm{hvars}_i(\sigma) \subseteq \mathrm{dom}(\sigma)$ is a finite set. By these two properties, the chain $\mathrm{hvars}_0(\sigma) \subseteq \mathrm{hvars}_1(\sigma) \subseteq \cdots$ is stationary and finitely computable. In particular, if $\ell = \#\,\sigma$, then, for all $n \geq \ell$, $\mathrm{hvars}_\ell(\sigma) = \mathrm{hvars}_n(\sigma)$.

Definition 12 (Finiteness operator.) *For each* $\sigma \in \mathrm{RSubst}$*, the* finiteness operator $\mathrm{hvars} \colon \mathrm{RSubst} \to \wp(\mathrm{Vars})$ *is given by* $\mathrm{hvars}(\sigma) \stackrel{\mathrm{def}}{=} \mathrm{hvars}_\ell(\sigma)$ *where* $\ell \stackrel{\mathrm{def}}{=} \ell(\sigma) \in \mathbb{N}$ *is such that* $\mathrm{hvars}_\ell(\sigma) = \mathrm{hvars}_n(\sigma)$ *for all* $n \geq \ell$*.*

The following proposition shows that the 'hvars' operator precisely captures the intended property.

Proposition 13 *If* $\sigma \in \mathrm{RSubst}$ *and* $x \in \mathrm{Vars}$ *then*

$$x \in \mathrm{hvars}(\sigma) \iff \mathrm{rt}(x, \sigma) \in \mathrm{HTerms}.$$

Example 14 *Consider* $\sigma \in \mathrm{RSubst}$*, where*

$$\sigma = \bigl\{ x_1 \mapsto f(x_2), x_2 \mapsto g(x_5), x_3 \mapsto f(x_4), x_4 \mapsto g(x_3) \bigr\}.$$

Then,

$$\begin{aligned} \mathrm{hvars}_0(\sigma) &= \mathrm{Vars} \setminus \{x_1, x_2, x_3, x_4\}, \\ \mathrm{hvars}_1(\sigma) &= \mathrm{Vars} \setminus \{x_1, x_3, x_4\}, \\ \mathrm{hvars}_2(\sigma) &= \mathrm{Vars} \setminus \{x_3, x_4\} \\ &= \mathrm{hvars}(\sigma). \end{aligned}$$

Thus, $x_1 \in \mathrm{hvars}(\sigma)$*, although* $\mathrm{vars}(x_1\sigma) \subseteq \mathrm{dom}(\sigma)$*.*

The following proposition states how 'gvars' and 'hvars' behave with respect to the further instantiation of variables.

Proposition 15 *Let* $\sigma, \tau \in \mathrm{RSubst}$*, where* $\tau \in {\downarrow}\,\sigma$*. Then*

$$\mathrm{hvars}(\sigma) \supseteq \mathrm{hvars}(\tau), \tag{15a}$$

$$\mathrm{gvars}(\sigma) \cap \mathrm{hvars}(\sigma) \subseteq \mathrm{gvars}(\tau) \cap \mathrm{hvars}(\tau). \tag{15b}$$

4.3 The Abstraction Function for H

A Galois connection between the concrete domain $\wp(\mathrm{RSubst})$ and the finiteness component $H = \wp(\mathrm{VI})$ can now be defined naturally.

Definition 16 (The Galois connection between $\wp(\mathrm{RSubst})$ and H.) *The abstraction function* $\alpha_H \colon \mathrm{RSubst} \to H$ *is defined, for each* $\sigma \in \mathrm{RSubst}$*, by*

$$\alpha_H(\sigma) \stackrel{\mathrm{def}}{=} \mathrm{VI} \cap \mathrm{hvars}(\sigma).$$

The concrete domain $\mathcal{D}^\flat$ *is related to* H *by means of the* abstraction function $\alpha_H \colon \mathcal{D}^\flat \to H$ *such that, for each* $\Sigma \in \wp(\mathrm{RSubst})$*,*

$$\alpha_H(\Sigma) \stackrel{\mathrm{def}}{=} \bigcap \bigl\{ \alpha_H(\sigma) \bigm| \sigma \in \Sigma \bigr\}.$$

Since the abstraction function α_H *is additive, the concretization function is given by its adjoint* [47]*: whenever* $h \in H$*,*

$$\begin{aligned}\gamma_H(h) &\stackrel{\mathrm{def}}{=} \bigl\{ \sigma \in \mathrm{RSubst} \bigm| \alpha_H(\sigma) \supseteq h \bigr\} \\ &\stackrel{\mathrm{def}}{=} \bigl\{ \sigma \in \mathrm{RSubst} \bigm| \mathrm{hvars}(\sigma) \supseteq h \bigr\}.\end{aligned}$$

With these definitions, we have the desired result: equivalent substitutions in rational solved form have the same finiteness abstraction.

Theorem 17 *If* $\sigma, \tau \in \mathrm{RSubst}$ *and* $\mathcal{RT} \vdash \forall(\sigma \leftrightarrow \tau)$*, then* $\alpha_H(\sigma) = \alpha_H(\tau)$*.*

4.4 Abstract Unification and Projection on $H \times P$

The abstract unification for the combined domain $H \times P$ is defined by using the abstract predicates and functions as specified for P as well as a new finiteness predicate for the domain H.

Definition 18 (Abstract unification on $H \times P$.) *A term* $t \in \mathrm{HTerms}$ *is a* finite tree in $h \in H$ *if and only if the predicate* $\mathrm{hterm}_h \colon \mathrm{HTerms} \to \mathrm{Bool}$ *holds for* t*, where*

$$\mathrm{hterm}_h(t) \stackrel{\mathrm{def}}{=} \bigl(\mathrm{vars}(t) \subseteq h\bigr).$$

The function $\mathrm{amgu}_H \colon (H \times P) \times \mathrm{Bind} \to H$ *captures the effects of a binding on an* H *element. Let* $\langle h, p \rangle \in H \times P$ *and* $(x \mapsto t) \in \mathrm{Bind}$. *Then*

$$\mathrm{amgu}_H\Bigl(\langle h, p \rangle, x \mapsto t\Bigr) \stackrel{\mathrm{def}}{=} h',$$

where h' *is given by the first case that applies in*

$$h' \stackrel{\mathrm{def}}{=} \begin{cases} h \cup \mathrm{vars}(t), & \textit{if } \mathrm{hterm}_h(x) \wedge \mathrm{ground}_p(x); \\ h \cup \{x\}, & \textit{if } \mathrm{hterm}_h(t) \wedge \mathrm{ground}_p(t); \\ h, & \textit{if } \mathrm{hterm}_h(x) \wedge \mathrm{hterm}_h(t) \\ & \quad \wedge \mathrm{ind}_p(x, t) \wedge \mathrm{or\_lin}_p(x, t); \\ h, & \textit{if } \mathrm{hterm}_h(x) \wedge \mathrm{hterm}_h(t) \\ & \quad \wedge \mathrm{gfree}_p(x) \wedge \mathrm{gfree}_p(t); \\ h \setminus \mathrm{share\_same\_var}_p(x, t), & \textit{if } \mathrm{hterm}_h(x) \wedge \mathrm{hterm}_h(t) \\ & \quad \wedge \mathrm{share\_lin}_p(x, t) \\ & \quad \wedge \mathrm{or\_lin}_p(x, t); \\ h \setminus \mathrm{share\_with}_p(x), & \textit{if } \mathrm{hterm}_h(x) \wedge \mathrm{lin}_p(x); \\ h \setminus \mathrm{share\_with}_p(t), & \textit{if } \mathrm{hterm}_h(t) \wedge \mathrm{lin}_p(t); \\ h \setminus \Bigl(\mathrm{share\_with}_p(x) \cup \mathrm{share\_with}_p(t)\Bigr), & \textit{otherwise.} \end{cases}$$

The abstract unification function $\mathrm{amgu} \colon (H \times P) \times \mathrm{Bind} \to H \times P$, *for any* $\langle h, p \rangle \in H \times P$ *and* $(x \mapsto t) \in \mathrm{Bind}$, *is given by*

$$\mathrm{amgu}\Bigl(\langle h, p \rangle, x \mapsto t\Bigr) \stackrel{\mathrm{def}}{=} \Bigl\langle \mathrm{amgu}_H\Bigl(\langle h, p \rangle, x \mapsto t\Bigr), \mathrm{amgu}_P(p, x \mapsto t) \Bigr\rangle.$$

In the computation of h' (the new finiteness component resulting from the abstract evaluation of a binding) there are eight cases based on properties holding for the concrete terms described by x and t.

(1) In the first case, the concrete term described by x is both finite and ground. Thus, after a successful execution of the binding, any concrete term described by t will be finite. Note that t could have contained variables which may be possibly bound to cyclic terms just before the execution of the binding.

(2) The second case is symmetric to the first one. Note that these are the only cases when a "positive" propagation of finiteness information is correct. In contrast, in all the remaining cases, the goal is to limit as much as possible the propagation of "negative" information, i.e., the possible cyclicity of terms.

(3) The third case exploits the classical results proved in research work on occurs-check reduction [50,51]. Accordingly, it is required that both x and t describe finite terms that do not share. The use of the implicitly

disjunctive predicate $\mathrm{or\_lin}_p$ allows for the application of this case even when neither x nor t are known to be definitely linear. For instance, as observed in [50], this may happen when the component P embeds the domain Pos for groundness analysis. [9]

(4) The fourth case exploits the observation that cyclic terms cannot be created when unifying two finite terms that are either ground or free. Ground-or-freeness [28,29] is a safe, more precise and inexpensive replacement for the classical freeness property when combining sharing analysis domains.
(5) The fifth case applies when unifying a linear and finite term with another finite term possibly sharing with it, provided they can only share linearly (namely, all the shared variables occur linearly in the considered terms). In such a context, only the shared variables can introduce cycles.
(6) In the sixth case, we drop the assumption about the finiteness of the term described by t. As a consequence, all variables sharing with x become possibly cyclic. However, provided x describes a finite and linear term, all finite variables independent from x preserve their finiteness.
(7) The seventh case is symmetric to the sixth one.
(8) The last case states that term finiteness is preserved for all variables that are independent from both x and t.

The following result, together with the assumption on amgu_P as specified in Definition 8, ensures that abstract unification on the combined domain $H \times P$ is correct.

Theorem 19 *Let* $\langle h, p\rangle \in H \times P$ *and* $(x \mapsto t) \in \mathrm{Bind}$*, where* $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$*. Let also* $\sigma \in \gamma_H(h) \cap \gamma_P(p)$ *and* $h' = \mathrm{amgu}_H\bigl(\langle h, p\rangle, x \mapsto t\bigr)$*. Then*

$$\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr) \implies \tau \in \gamma_H(h').$$

Abstract projection on the composite domain $H \times P$ is much simpler than abstract unification, because in this case there is no interaction between the two components of the abstract domain.

Definition 20 (Abstract projection on $H \times P$.) *The function* $\mathrm{proj}_H \colon H \times \mathrm{VI} \to H$ *captures the effects, on the* H *component, of projecting away a variable. For each* $h \in H$ *and* $x \in \mathrm{VI}$*,*

$$\mathrm{proj}_H(h, x) \stackrel{\mathrm{def}}{=} h \cup \{x\}.$$

The abstract variable projection function $\mathrm{proj} \colon (H \times P) \times \mathrm{VI} \to H \times P$*, for*

---

[9] Let t be y. Let also P be Pos. Then, given the Pos formula $\phi \stackrel{\mathrm{def}}{=} (x \vee y)$, both $\mathrm{ind}_\phi(x, y)$ and $\mathrm{or\_lin}_\phi(x, y)$ satisfy the conditions in Definition 4. Note that from ϕ we cannot infer that x is definitely linear and neither that y is definitely linear.

any $\langle h, p \rangle \in H \times P$ *and* $x \in \mathrm{VI}$*, is given by*

$$\operatorname{proj}\Bigl(\langle h, p \rangle, x\Bigr) \stackrel{\mathrm{def}}{=} \Bigl\langle \operatorname{proj}_H(h, x), \operatorname{proj}_P(p, x) \Bigr\rangle.$$

As a consequence, as far as the H component is concerned, the correctness of the projection function does not depend on the assumption on proj_P as specified in Definition 8.

Theorem 21 *Let* $x \in \mathrm{VI}$*,* $h \in H$ *and* $\sigma \in \gamma_H(h)$*. Then*

$$\tau \in \exists x \,.\, \{\sigma\} \implies \tau \in \gamma_H\Bigl(\operatorname{proj}_H(h, x)\Bigr).$$

We do not consider the disjunction and conjunction operations here. The implementation (and therefore proof of correctness) for disjunction is straightforward and omitted. The implementation of *independent conjunction* where the descriptions are renamed apart is also straightforward. On the other hand, full conjunction, which is only needed for a top-down analysis framework, can be approximated by combining unification and independent conjunction, obtaining a correct (although possibly less precise) analysis.

Several abstract domains for sharing analysis can be used to implement the parameter component P. As a basic implementation, one could consider the well-known set-sharing domain of Jacobs and Langen [55]. In such a case, most of the required correctness results have already been established in [54]. Note however that, since no freeness and linearity information is recorded in the plain set-sharing domain, some of the predicates of Definition 8 need to be grossly approximated. For instance, the predicate gfree_p will provide useful information only when applied to an argument that is known to be definitely ground. Another possibility would be to use the domain based on pair-sharing, definite groundness and definite linearity described in [37]. A more precise choice is constituted by the SFL domain (an acronym standing from Set-sharing plus Freeness plus Linearity) introduced in [57,33]. Even in this case, all the non-trivial correctness results have already been proved. In particular, in [32,33] it is shown that the abstraction function satisfies the requirement of Definition 7 and that the abstract unification operator is correct with respect to rational-tree unification. In order to better highlight the generality of our specification of the sharing component P, the instantiation of P to SFL is presented in Appendix A. Notice that the quest for more precision does not end with SFL: a number of possible precision improvements are presented and discussed in [28,29].

5 Finite-Tree Dependencies

The precision of the finite-tree analysis based on $H \times P$ is highly dependent on the precision of the generic component P. As explained before, the information provided by P on groundness, freeness, linearity, and sharing of variables is exploited, in the combination $H \times P$, to circumscribe as much as possible the creation and propagation of cyclic terms. However, finite-tree analysis can also benefit from other kinds of relational information. In particular, we now show how *finite-tree dependencies* allow a positive propagation of finiteness information.

Let us consider the finite terms $t_1 = f(x)$, $t_2 = g(y)$, and $t_3 = h(x, y)$: it is clear that, for each assignment of rational terms to x and y, t_3 is finite if and only if t_1 and t_2 are so. We can capture this by the Boolean formula $t_3 \leftrightarrow (t_1 \wedge t_2)$.[10] The reasoning is based on the following facts:

(1) t_1, t_2, and t_3 are finite terms, so that the finiteness of their instances depends only on the finiteness of the terms that take the place of x and y.
(2) $\mathrm{vars}(t_3) \supseteq \mathrm{vars}(t_1) \cup \mathrm{vars}(t_2)$, that is, t_3 *covers* both t_1 and t_2; this means that, if an assignment to the variables of t_3 produces a finite instance of t_3, that very assignment will necessarily result in finite instances of t_1 and t_2. Conversely, an assignment producing non-finite instances of t_1 or t_2 will forcibly result in a non-finite instance of t_3.
(3) Similarly, t_1 and t_2, taken together, cover t_3.

The important point to notice is that this dependency will keep holding for any further simultaneous instantiation of t_1, t_2, and t_3. In other words, such dependencies are preserved by forward computations (which proceed by consistently instantiating program variables).

Consider $x \mapsto t \in \mathit{Bind}$ where $t \in \mathrm{HTerms}$ and $\mathrm{vars}(t) = \{y_1, \ldots, y_n\}$. After this binding has been successfully applied, the destinies of x and t concerning term-finiteness are tied together: forever. This tie can be described by the dependency formula

$$x \leftrightarrow (y_1 \wedge \cdots \wedge y_n), \tag{2}$$

meaning that x will be bound to a finite term if and only if y_i is bound to a finite term, for each $i = 1, \ldots, n$. While the dependency expressed by (2) is a correct description of any computation state following the application of the binding $x \mapsto t$, it is not as precise as it could be. Suppose that x and y_k are

[10] The introduction of such Boolean formulas, called *dependency formulas*, is originally due to P. W. Dart [58].

indeed the same variable. Then (2) is logically equivalent to

$$x \rightarrow (y_1 \wedge \cdots \wedge y_{k-1} \wedge y_{k+1} \wedge \cdots \wedge y_n). \tag{3}$$

Although this is correct —whenever x is bound to a finite term, all the other variables will be bound to finite terms— it misses the point that x has just been bound, irrevocably, to a non-finite term: no forward computation can change this. Thus, the implication (3) holds vacuously. A more precise and correct description for the state of affairs caused by the cyclic binding is, instead, the negated atom $\neg x$, whose intuitive reading is "x is not (and never will be) finite."

We are building an abstract domain for finite-tree dependencies where we are making the deliberate choice of including only information that cannot be withdrawn by forward computations. The reason for this choice is that we want the concrete constraint accumulation process to be paralleled, at the abstract level, by another constraint accumulation process: logical conjunction of Boolean formulas. For this reason, it is important to distinguish between *permanent* and *contingent* information. Permanent information, once established for a program point p, maintains its validity in all points that follow p in any forward computation. Contingent information, instead, does not carry its validity beyond the point where it is established. An example of contingent information is given by the h component of $H \times P$: having $x \in h$ in the description of some program point means that x is definitely bound to a finite term *at that point*; nothing is claimed about the finiteness of x at later program points and, in fact, unless x is ground, x can still be bound to a non-finite term. However, if at some program point x is finite and ground, then x will remain finite. In this case we will ensure our Boolean dependency formula entails the positive atom x.

At this stage, we already know something about the abstract domain we are designing. In particular, we have positive and negated atoms, the requirement of describing program predicates of any arity implies that arbitrary conjunctions of these atomic formulas must be allowed and, finally, it is not difficult to observe that the merge-over-all-paths operation [47] will be logical disjunction, so that the domain will have to be closed under this operation. This means that the carrier of our domain must be able to express any Boolean function over the finite set VI of the variables of interest: Bfun is the carrier.

Definition 22 **($\gamma_F \colon \mathrm{Bfun} \rightarrow \wp(\mathrm{RSubst})$.)** *The function* $\mathrm{hval} \colon \mathrm{RSubst} \rightarrow \mathrm{Bval}$ *is defined, for each* $\sigma \in \mathrm{RSubst}$ *and each* $x \in \mathrm{VI}$*, by*

$$\mathrm{hval}(\sigma)(x) = 1 \quad \stackrel{\mathrm{def}}{\Longleftrightarrow} \quad x \in \mathrm{hvars}(\sigma).$$

The concretization function $\gamma_F \colon \mathrm{Bfun} \rightarrow \wp(\mathrm{RSubst})$ *is defined, for* $\phi \in \mathrm{Bfun}$*,* *by*

$$\gamma_F(\phi) \stackrel{\text{def}}{=} \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\, \sigma : \phi\bigl(\mathrm{hval}(\tau)\bigr) = 1 \,\Bigr\}.$$

The domain of positive Boolean functions Pos used, among other things, for groundness analysis is so popular that our use of the domain Bfun deserves some further comments. For the representation of finite-tree dependencies, the presence in the domain of negative functions such as $\neg x$, meaning that x is bound to an infinite term, is an important feature. One reason why it is so is that knowing about definite non-finiteness can improve the information on definite finiteness. The easiest example goes as follows: if we know that either x or y is finite (i.e., $x \vee y$) and we know that x is *not* finite (i.e., $\neg x$), then we can deduce that y must be finite (i.e., y). It is important to observe that this reasoning can be applied, verbatim, to groundness: a knowledge of non-groundness may improve groundness information. The big difference is that non-finiteness is information of the permanent kind while non-groundness is only contingent. As a consequence, a knowledge of finiteness and non-finiteness can be monotonically accumulated along computation paths by computing the logical conjunction of Boolean formulae. An approach where groundness and non-groundness information is represented by elements of Bfun would need to use a much more complex operation and significant extra information to correctly model the constraint accumulation process.

The other reason why the presence of negative functions in the domain is beneficial is efficiency. The most efficient implementations of Pos and Bfun, such as the ones described in [42,59], are based on Reduced Ordered Binary Decision Diagrams (ROBDD) [60]. While an ROBDD representing the imprecise information given by the formula (3) has a worst case complexity that is exponential in n, the more precise formula $\neg x$ has constant complexity.

The following theorem shows how most of the operators needed to compute the concrete semantics of a logic program can be correctly approximated on the abstract domain Bfun. Notice how the addition of equations is modeled by logical conjunction and projection of a variable is modeled by existential quantification.

Theorem 23 *Let* $\Sigma, \Sigma_1, \Sigma_2 \in \wp(\mathrm{RSubst})$ *and* $\phi, \phi_1, \phi_2 \in \mathrm{Bfun}$ *be such that* $\gamma_F(\phi) \supseteq \Sigma$, $\gamma_F(\phi_1) \supseteq \Sigma_1$, *and* $\gamma_F(\phi_2) \supseteq \Sigma_2$. *Let also* $(x \mapsto t) \in \mathrm{Bind}$, *where* $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$. *Then the following hold:*

$$\gamma_F\Bigl(x \leftrightarrow \bigwedge \mathrm{vars}(t)\Bigr) \supseteq \Bigl\{\{x \mapsto t\}\Bigr\}; \tag{23a}$$

$$\gamma_F(\neg x) \supseteq \Bigl\{\{x \mapsto t\}\Bigr\}, \ \textit{if } x \in \mathrm{vars}(t); \tag{23b}$$

$$\gamma_F(x) \supseteq \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| x \in \mathrm{gvars}(\sigma) \cap \mathrm{hvars}(\sigma) \,\Bigr\}; \tag{23c}$$

$$\gamma_F(\phi_1 \wedge \phi_2) \supseteq \Bigl\{\, \mathrm{mgs}(\sigma_1 \cup \sigma_2) \Bigm| \sigma_1 \in \Sigma_1, \sigma_2 \in \Sigma_2 \,\Bigr\}; \tag{23d}$$

$$\gamma_F(\phi_1 \vee \phi_2) \supseteq \Sigma_1 \cup \Sigma_2; \tag{23e}$$
$$\gamma_F(\exists x \,.\, \phi) \supseteq \exists\!\!\exists\, x \,.\, \Sigma. \tag{23f}$$

Cases (23a), (23b), and (23d) of Theorem 23 ensure that the following definition of amgu_F provides a correct approximation on Bfun of the concrete unification of rational trees.

Definition 24 *The function* $\mathrm{amgu}_F \colon \mathrm{Bfun} \times \mathrm{Bind} \to \mathrm{Bfun}$ *captures the effects of a binding on a finite-tree dependency formula. Let* $\phi \in \mathrm{Bfun}$ *and* $(x \mapsto t) \in \mathrm{Bind}$ *be such that* $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$. *Then*

$$\mathrm{amgu}_F(\phi, x \mapsto t) \stackrel{\mathrm{def}}{=} \begin{cases} \phi \wedge \Bigl(x \leftrightarrow \bigwedge \mathrm{vars}(t)\Bigr), & \text{if } x \notin \mathrm{vars}(t); \\ \phi \wedge \neg x, & \text{otherwise.} \end{cases}$$

Other semantic operators, such as the consistent renaming of variables, are very simple and omitted for the sake of brevity.

The next result shows how finite-tree dependencies may improve the finiteness information encoded in the h component of the domain $H \times P$.

Theorem 25 *Let* $h \in H$ *and* $\phi \in \mathrm{Bfun}$. *Let also* $h' \stackrel{\mathrm{def}}{=} \mathrm{true}\Bigl(\phi \wedge \bigwedge h\Bigr)$. *Then*

$$\gamma_H(h) \cap \gamma_F(\phi) = \gamma_H(h') \cap \gamma_F(\phi).$$

Example 26 *Consider the following program, where it is assumed that the only "external" query is '*`?- r(X, Y)`*':*

```
p(X, Y) :- X = f(Y, _).
q(X, Y) :- X = f(_, Y).
r(X, Y) :- p(X, Y), q(X, Y), acyclic_term(X).
```

Then the predicate `p/2` *in the clause defining* `r/2` *will be called with* `X` *and* `Y` *both unbound. Computing on the abstract domain* $H \times P$ *gives us the finiteness description* $h_p = \{x, y\}$, *expressing the fact that both* `X` *and* `Y` *are bound to finite terms. Computing on the finite-tree dependencies domain* Bfun, *gives us the Boolean formula* $\phi_p = x \to y$ *(*`Y` *is finite if* `X` *is so).*

Considering now the call to the predicate `q/2`, *we note that, since variable* `X` *is already bound to a non-variable term sharing with* `Y`, *all the finiteness information encoded by* H *will be lost (i.e.,* $h_q = \varnothing$*). So, both* `X` *and* `Y` *are detected as possibly cyclic. However, the finite-tree dependency information is preserved, since we have* $\phi_q = (x \to y) \wedge (x \to y) = x \to y$.

Finally, consider the effect of the abstract evaluation of `acyclic_term(X)`*. On the $H \times P$ domain we can only infer that variable* `X` *cannot be bound to an infinite term, while* `Y` *will be still considered as possibly cyclic, so that $h_r = \{x\}$. On the domain* Bfun *we can just confirm that the finite-tree dependency computed so far still holds, so that $\phi_r = x \rightarrow y$ (no stronger finite-tree dependency can be inferred, since the finiteness of* `X` *is only contingent). Thus, by applying the result of* Theorem 25, *we can recover the finiteness of* `Y`*:*

$$h'_r = \mathrm{true}\Bigl(\phi_r \wedge \bigwedge h_r\Bigr) = \mathrm{true}\bigl((x \rightarrow y) \wedge x\bigr) = \mathrm{true}(x \wedge y) = \{x, y\}.$$

Information encoded in $H \times P$ and Bfun is not completely orthogonal and the following result provides a kind of consistency check.

Theorem 27 *Let $h \in H$ and $\phi \in \mathrm{Bfun}$. Then*

$$\gamma_H(h) \cap \gamma_F(\phi) \neq \varnothing \quad \Longrightarrow \quad h \cap \mathrm{false}\Bigl(\phi \wedge \bigwedge h\Bigr) = \varnothing.$$

Note however that, provided the abstract operators are correct, the computed descriptions will always be mutually consistent, unless $\phi = \bot$.

6 Groundness Dependencies

Since information about the groundness of variables is crucial for many applications, it is natural to consider a static analysis domain including both a finite-tree and a groundness component. In fact, any reasonably precise implementation of the parameter component P of the abstract domain specified in Section 4 will include some kind of groundness information. [11] We highlight similarities, differences and connections relating the domain Bfun for finite-tree dependencies to the abstract domain Pos for groundness dependencies. Note that these results also hold when considering a combination of Bfun with the groundness domain Def [42].

We first define how elements of Pos represent sets of substitutions in rational solved form.

[11] One could define P so that it explicitly contains the abstract domain Pos. Even when this is not the case, it should be noted that, as soon as the parameter P includes the set-sharing domain of Jacobs and Langen [61], then it will subsume the groundness information captured by the domain Def [62,63].

Definition 28 (**$\gamma_G \colon \mathrm{Pos} \to \wp(\mathrm{RSubst})$.**) *The function* $\mathrm{gval} \colon \mathrm{RSubst} \to \mathrm{Bval}$ *is defined as follows, for each* $\sigma \in \mathrm{RSubst}$ *and each* $x \in \mathrm{VI}$*:*

$$\mathrm{gval}(\sigma)(x) = 1 \quad \overset{\mathrm{def}}{\iff} \quad x \in \mathrm{gvars}(\sigma).$$

The concretization function $\gamma_G \colon \mathrm{Pos} \to \wp(\mathrm{RSubst})$ *is defined, for each* $\psi \in \mathrm{Pos}$*,*

$$\gamma_G(\psi) \stackrel{\mathrm{def}}{=} \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in \mathord{\downarrow}\, \sigma : \psi\bigl(\mathrm{gval}(\tau)\bigr) = 1 \,\Bigr\}.$$

The following is a simple variant of the standard abstract unification operator for groundness analysis over finite-tree domains: the only difference concerns the case of cyclic bindings [64].

Definition 29 *The function* $\mathrm{amgu}_G \colon \mathrm{Pos} \times \mathrm{Bind} \to \mathrm{Pos}$ *captures the effects of a binding on a groundness dependency formula. Let* $\psi \in \mathrm{Pos}$ *and* $(x \mapsto t) \in \mathrm{Bind}$ *be such that* $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$*. Then*

$$\mathrm{amgu}_G(\psi, x \mapsto t) \stackrel{\mathrm{def}}{=} \psi \wedge \Bigl(x \leftrightarrow \bigwedge\bigl(\mathrm{vars}(t) \setminus \{x\}\bigr)\Bigr).$$

The next result shows how, by exploiting the finiteness component H, the finite-tree dependencies (Bfun) component and the groundness dependencies (Pos) component can improve each other.

Theorem 30 *Let* $h \in H$*,* $\phi \in \mathrm{Bfun}$ *and* $\psi \in \mathrm{Pos}$*. Let also* $\phi' \in \mathrm{Bfun}$ *and* $\psi' \in \mathrm{Pos}$ *be defined as* $\phi' = \exists \mathrm{VI} \setminus h \mathrel{.} \psi$ *and* $\psi' = \mathrm{pos}(\exists \mathrm{VI} \setminus h \mathrel{.} \phi)$*. Then*

$$\gamma_H(h) \cap \gamma_F(\phi) \cap \gamma_G(\psi) = \gamma_H(h) \cap \gamma_F(\phi) \cap \gamma_G(\psi \wedge \psi'); \tag{30a}$$

$$\gamma_H(h) \cap \gamma_F(\phi) \cap \gamma_G(\psi) = \gamma_H(h) \cap \gamma_F(\phi \wedge \phi') \cap \gamma_G(\psi). \tag{30b}$$

Moreover, even without any knowledge of the H component, combining Theorem 25 and Eq. (30a), the groundness dependencies component can be improved.

Theorem 31 *Let* $\phi \in \mathrm{Bfun}$ *and* $\psi \in \mathrm{Pos}$*. Then*

$$\gamma_F(\phi) \cap \gamma_G(\psi) = \gamma_F(\phi) \cap \gamma_G\Bigl(\psi \wedge \bigwedge \mathrm{true}(\phi)\Bigr).$$

The following example shows that, when computing on rational trees, finite-tree dependencies may provide groundness information that is not captured by the usual approaches.

Example 32 *Consider the program:*

```
p(a, Y).
p(X, a).
q(X, Y) :- p(X, Y), X = f(X, Z).
```

The abstract semantics of `p/2`*, for both finite-tree and groundness dependencies, is* $\phi_p = \psi_p = x \vee y$*. The finite-tree dependency for* `q/2` *is* $\phi_q = (x \vee y) \wedge \neg x = \neg x \wedge y$*. Using* Definition 29*, the groundness dependency for* `q/2` *is*

$$\psi_q = \exists z \mathrel{.} \Bigl((x \vee y) \wedge (x \leftrightarrow z) \Bigr) = x \vee y.$$

This can be improved, using Theorem 31*, to*

$$\psi'_q = \psi_q \wedge \bigwedge \mathrm{true}(\phi_q) = y.$$

It is worth noticing that the groundness information can be improved regardless of whether, like Pos, the groundness domain captures disjunctive information: groundness information represented by the less expressive domain Def [42] can be improved as well. The next example illustrates this point.

Example 33 *Consider the following program:*

```
p(a, a).
p(X, Y) :- X = f(X, _).
q(X, Y) :- p(X, Y), X = a.
```

Consider the predicate `p/2`*. Concerning finite-tree dependencies, the abstract semantics of* `p/2` *is expressed by the Boolean formula* $\phi_p = (x \wedge y) \vee \neg x = x \rightarrow y$ *(*`Y` *is finite if* `X` *is so). In contrast, the* Pos*-groundness abstract semantics of* `p/2` *is a plain "don't know": the Boolean formula* $\psi_p = (x \wedge y) \vee \top = \top$*. In fact, the groundness of* `X` *and* `Y` *can be completely decided by the call-pattern of* `p/2`*.*

Consider now the predicate `q/2`*. The finiteness semantics of* `q/2` *is given by* $\phi_q = (x \rightarrow y) \wedge x = x \wedge y$*, whereas the* Pos *formula expressing groundness dependencies is* $\psi_q = \top \wedge x = x$*. By* Theorem 31*, we obtain*

$$\psi'_q = \psi_q \wedge \bigwedge \mathrm{true}(\phi_q) = x \wedge y,$$

therefore recovering the groundness of variable y*.*

Since better groundness information, besides being useful in itself, may also improve the precision of many other analyses such as sharing [28,29,62], the reduction steps given by Theorems 30 and 31 can trigger improvements to the precision of other components. Theorem 30 can also be exploited to recover precision after the application of a widening operator on either the groundness dependencies or the finite-tree dependencies component.

7 Experimental Results

The work described here has been experimentally evaluated in the framework provided by CHINA [64], a data-flow analyzer for constraint logic languages (i.e., ISO Prolog, CLP($\mathcal{R}$), `clp(FD)` and so forth). CHINA performs bottom-up analysis deriving information on both call-patterns and success-patterns by means of program transformations and optimized fixpoint computation techniques. [12] An abstract description is computed for the call- and success-patterns for each predicate defined in the program.

We implemented and compared the three domains Pattern(P), Pattern($H \times P$) and Pattern(Bfun $\times$ H $\times$ P), [13] where the parameter component P has been instantiated to the domain Pos $\times$ SFL$_2$ [28,32,33] for tracking groundness, freeness, linearity and (non-redundant) set-sharing information. The Pattern($\cdot$) operator [49] further upgrades the precision of its argument by adding explicit structural information. Note that the analyzer tracks the finiteness of the terms that can be bound to those abstract variables occurring as leaves in the acyclic term structure computed by the Pattern($\cdot$) component; therefore, in order to show that an abstract variable is definitely bound to a finite term, the basic domain Pattern(P) has to prove that this variable is definitely free. [14]

Concerning the Bfun component, the implementation was straightforward, since all the techniques described in [59] (and almost all the code, including the widenings) was reused unchanged, obtaining comparable efficiency. As a consequence, most of the implementation effort was in the coding of the abstract operators on the H component and in the reduction processes between the

[12] More precisely, CHINA uses a variation of the *Magic Templates* algorithm [65], in order to obtain goal-dependent information, and a sophisticated chaotic iteration strategy proposed in [66,67] (recursive fixpoint iteration on the weak topological ordering defined by partitioning of the call graph into strongly-connected subcomponents).

[13] For ease of notation, the domain names are shortened to P, H and B, respectively.

[14] Put in other words, by considering just the variables occurring inside the pattern structure, we systematically disregard those cases when the basic domain is able to prove that a particular argument position is definitely bound to a finite and ground term such as $f(a)$. Clearly, the same approach is consistently adopted when considering the more accurate analysis domains.

different components. A key choice, in this sense, is *when* the reduction steps given in Theorems 25 and 30 should be applied. When striving for maximum precision, a trivial strategy is to perform reductions immediately after any application of any abstract operator. This is how predicates like `acyclic_term/1` should be handled: after adding the variables of the argument to the H component, the reduction process is applied to propagate the new information to all domain components. However, such an approach turns out to be unnecessarily inefficient. In fact, the next result shows that Theorems 25 and 30 cannot lead to a precision improvement if applied just after the abstract evaluation of the merge-over-all-paths or the existential quantification operations (provided the initial descriptions are already reduced).

Theorem 34 *Let* $x \in \mathrm{VI}$*,* $h, h' \in H$ $\phi, \phi' \in \mathrm{Bfun}$ *and* $\psi, \psi' \in \mathrm{Pos}$ *and suppose that* $\gamma_H(h) \cap \gamma_F(\phi) \neq \varnothing$*. Let*

$$
\begin{aligned}
h_1 &\stackrel{\mathrm{def}}{=} h \cap h', & \phi_1 &\stackrel{\mathrm{def}}{=} \phi \vee \phi', & \psi_1 &\stackrel{\mathrm{def}}{=} \psi \vee \psi', \\
h_2 &\stackrel{\mathrm{def}}{=} \mathrm{proj}_H(h, x), & \phi_2 &\stackrel{\mathrm{def}}{=} \exists x \mathrel{.} \phi, & \psi_2 &\stackrel{\mathrm{def}}{=} \exists x \mathrel{.} \psi.
\end{aligned}
$$

Let also

$$
\begin{aligned}
h &\supseteq \mathrm{true}\Bigl(\phi \wedge \bigwedge h\Bigr), & \phi &\models (\exists \mathrm{VI} \setminus h \mathrel{.} \psi), & \psi &\models \mathrm{pos}(\exists \mathrm{VI} \setminus h \mathrel{.} \phi), \\
h' &\supseteq \mathrm{true}\Bigl(\phi' \wedge \bigwedge h'\Bigr), & \phi' &\models (\exists \mathrm{VI} \setminus h' \mathrel{.} \psi'), & \psi' &\models \mathrm{pos}(\exists \mathrm{VI} \setminus h' \mathrel{.} \phi').
\end{aligned}
$$

Then, for $i = 1$*,* 2*,*

$$
h_i \supseteq \mathrm{true}\Bigl(\phi_i \wedge \bigwedge h_i\Bigr), \quad \phi_i \models (\exists \mathrm{VI} \setminus h_i \mathrel{.} \psi_i), \quad \psi_i \models \mathrm{pos}(\exists \mathrm{VI} \setminus h_i \mathrel{.} \phi_i).
$$

A goal-dependent analysis was run for all the programs in our benchmark suite.[15] For 116 of them, the analyzer detects that the program in not amenable to goal-dependent analysis, either because the entry points are unknown or because the program uses builtins in a way that every predicate can be called with any call-pattern, so that the analysis provides results that are so imprecise to be irrelevant. The precision results for the remaining 248 programs are summarized in Table 1. Here, the precision is measured as the percentage of

[15] The suite comprises all the logic programs we have access to (including everything we could find by systematically dredging the Internet): 364 programs, 24 MB of code, 800 K lines. Besides classical benchmarks, several real programs of respectable size are included, the largest one containing 10063 clauses in 45658 lines of code. The suite also comprises a few synthetic benchmarks, which are artificial programs explicitly constructed to stress the capabilities of the analyzer and of its abstract domains with respect to precision and/or efficiency. The interested reader can find more information at the URI `http://www.cs.unipr.it/China/`.

Prec. class	P	H	B
$p = 100$	2	84	86
$80 \leq p < 100$	1	31	36
$60 \leq p < 80$	7	26	23
$40 \leq p < 60$	6	41	40
$20 \leq p < 40$	47	47	46
$0 \leq p < 20$	185	19	17

Prec. improvement	P $\rightarrow$ H	H $\rightarrow$ B
$i > 20$	185	4
$10 < i \leq 20$	31	3
$5 < i \leq 10$	11	6
$2 < i \leq 5$	4	10
$0 < i \leq 2$	2	24
no improvement	15	201

Table 1
The precision on finite variables when using P, H and B.

the total number of variables that the analyser can show to be finite. Two alternative views are provided.

In the first view, each column is labeled by an analysis domain and each row is labeled by a precision interval. For instance, the value '31' at the intersection of column 'H' and row '$80 \leq p < 100$' is to be read as "*for 31 benchmarks, the percentage p of the total number of variables that the analyzer can show to be finite using the domain H is between 80% and 100%.*"

The second view provides a better picture of the precision *improvements* obtained when moving from P to H (in the column 'P $\rightarrow$ H') and from H to B (in the column 'H $\rightarrow$ B'). For instance, the value '10' at the intersection of column 'H $\rightarrow$ B' and row '$2 < i \leq 5$' is to be read as "*when moving from H to B, for 10 benchmarks the improvement i in the percentage of the total number of variables shown to be finite was between 2% and 5%.*"

It can be seen from Table 1 that, even though the H domain is remarkably precise, the inclusion of the Bfun component allows for a further, and sometimes significant, precision improvement for a number of benchmarks. It is worth noting that the current implementation of CHINA does not yet fully exploit the finite-tree dependencies arising when evaluating many of the built-in predicates, therefore incurring an avoidable precision loss. We are working on this issue and we expect that the specialized implementation of the abstract evaluation of some built-ins will result in more and better precision improvements. The experimentation has also shown that, in practice, the Bfun component does not improve the groundness information.

Concerning efficiency, our experimentation shown that the techniques we propose are really practical. The total analysis time for the 248 programs for which we give precision results in Table 1 is 596 seconds for P, 602 seconds for H, and 1211 seconds for B.[16] It should be stressed that, as mentioned before, the implementation of Bfun was derived in a straightforward way from the one of Pos described in [59]. We believe that a different tuning of the widenings we employ in that component could reduce the gap between the efficiency of H and the one of B.

8 Conclusion

Several modern logic-based languages offer a computation domain based on rational trees. On the one hand, the use of such trees is encouraged by the possibility of using efficient and correct unification algorithms and by an increase in expressivity. On the other hand, these gains are countered by the extra problems rational trees bring with themselves and that can be summarized as follows: several built-ins, library predicates, program analysis and manipulation techniques are only well-defined for program fragments working with finite trees.

As a consequence, those applications that exploit rational trees tend to do so in a very controlled way, that is, most program variables can only be bound to finite terms. By detecting the program variables that may be bound to infinite terms with a good degree of accuracy, we can significantly reduce the disadvantages of using rational trees.

In this paper we have proposed an abstract-interpretation based solution to this problem, where the composite abstract domain $H \times P$ allows tracking of the creation and propagation of infinite terms. Even though this information is crucial to any finite-tree analysis, propagating the guarantees of finiteness that come from several built-ins (including those that are explicitly provided to test term-finiteness) is also important. Therefore, we have introduced a domain of Boolean functions Bfun for finite-tree dependencies which, when

[16] On a PC system equipped with an Athlon XP 2800 CPU, 1 GB of RAM memory and running GNU/Linux.

coupled to the domain $H \times P$, can enhance its expressive power. Since Bfun has many similarities with the domain Pos used for groundness analysis, we have investigated how these two domains relate to each other and, in particular, the synergy arising from their combination in the "global" domain of analysis.

Acknowledgment

We would like to express our gratitude to the Journal referees for their useful comments that have helped improve the final versions of the paper.

References

[1] A. Colmerauer, Prolog and infinite trees, in: K. L. Clark, S. Å. Tärnlund (Eds.), Logic Programming, APIC Studies in Data Processing, Vol. 16, Academic Press, New York, 1982, pp. 231–251.

[2] A. Colmerauer, An introduction to Prolog-III, Communications of the ACM 33 (7) (1990) 69–90.

[3] Swedish Institute of Computer Science, Intelligent Systems Laboratory, SICStus Prolog User's Manual, release 3.9 Edition (2002).

[4] G. Smolka, R. Treinen, Records for logic programming, Journal of Logic Programming 18 (3) (1994) 229–258.

[5] V. Santos Costa, L. Damas, R. Reis, R. Azevedo, YAP User's Manual, Universidade do Porto, version 4.3.20 Edition (2001).

[6] P. R. Eggert, K. P. Chow, Logic programming, graphics and infinite terms, Tech. Rep. UCSB DoCS TR 83-02, Department of Computer Science, University of California at Santa Barbara (1983).

[7] F. Giannesini, J. Cohen, Parser generation and grammar manipulation using Prolog's infinite trees, Journal of Logic Programming 3 (1984) 253–265.

[8] P. Cousot, R. Cousot, Formal language, grammar and set-constraint-based program analysis by abstract interpretation, in: Proceedings of the Seventh ACM Conference on Functional Programming Languages and Computer Architecture, ACM Press, La Jolla, California, 1995, pp. 170–181.

[9] G. Janssens, M. Bruynooghe, Deriving descriptions of possible values of program variables by means of abstract interpretation, Journal of Logic Programming 13 (2&3) (1992) 205–258.

[10] P. Van Hentenryck, A. Cortesi, B. Le Charlier, Type analysis of Prolog using type graphs, Journal of Logic Programming 22 (3) (1995) 179–209,.

[11] M. Filgueiras, A Prolog interpreter working with infinite terms, in: Campbell [68], pp. 250–258.

[12] S. Haridi, D. Sahlin, Efficient implementation of unification of cyclic structures, in: Campbell [68], pp. 234–249.

[13] M. Carro, An application of rational trees in a logic programming interpreter for a procedural language, Tech. Rep. `arXiv:cs.DS/0403028`, School of Computer Science, Technical University of Madrid (UPM), available from `http://arxiv.org/` (2004).

[14] K. Mukai, Constraint logic programming and the unification of information, Ph.D. thesis, Department of Computer Science, Faculty of Engineering, Tokio Institute of Technology (1991).

[15] C. Pollard, I. A. Sag, Head-Driven Phrase Structure Grammar, University of Chicago Press, Chicago, 1994.

[16] B. Carpenter, The Logic of Typed Feature Structures with Applications to Unification-based Grammars, Logic Programming and Constraint Resolution, Vol. 32 of Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, New York, 1992.

[17] G. Erbach, ProFIT: Prolog with Features, Inheritance and Templates, in: Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics, Dublin, Ireland, 1995, pp. 180–187.

[18] M. Codish, C. Taboch, A semantic basis for termination analysis of logic programs and its realization using symbolic norm constraints, in: M. Hanus, J. Heering, K. Meinke (Eds.), Algebraic and Logic Programming, 6th International Joint Conference, Vol. 1298 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Southampton, U.K., 1997, pp. 31–45.

[19] M. Codish, C. Taboch, A semantic basis for the termination analysis of logic programs, Journal of Logic Programming 41 (1) (1999) 103–123.

[20] N. Lindenstrauss, Y. Sagiv, A. Serebrenik, TermiLog: A system for checking termination of queries to logic programs, in: O. Grumberg (Ed.), Computer Aided Verification: Proceedings of the 9th International Conference, Vol. 1250 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Haifa, Israel, 1997, pp. 444–447.

[21] F. Mesnard, R. Bagnara, cTI: A constraint-based termination inference tool for ISO-Prolog, Theory and Practice of Logic Programming 5 (1&2), to appear.

[22] R. F. Stärk, Total correctness of pure Prolog programs: A formal approach, in: R. Dyckhoff, H. Herre, P. Schroeder-Heister (Eds.), Extensions of Logic Programming: Proceedings of the 5th International Workshop, Vol. 1050 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Leipzig, Germany, 1996, pp. 237–254.

[23] R. F. Stärk, The theoretical foundations of LPTP (a Logic Program Theorem Prover), Journal of Logic Programming 36 (3) (1998) 241–269.

[24] A. Cortesi, B. Le Charlier, S. Rossi, Specification-based automatic verification of Prolog programs, in: J. P. Gallagher (Ed.), Logic Program Synthesis and Transformation: Proceedings of the 6th International Workshop, Vol. 1207 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Stockholm, Sweden, 1997, pp. 38–57.

[25] S. Debray, N.-W. Lin, Cost analysis of logic programs, ACM Transactions on Programming Languages and Systems 15 (5) (1993) 826–875.

[26] M. V. Hermenegildo, F. Bueno, G. Puebla, P. López, Program analysis, debugging, and optimization using the ciao system preprocessor, in: D. De Schreye (Ed.), Logic Programming: The 1999 International Conference, MIT Press Series in Logic Programming, The MIT Press, Las Cruces, New Mexico, 1999, pp. 52–66.

[27] ISO/IEC, ISO/IEC 13211-1: 1995 Information technology — Programming languages — Prolog — Part 1: General core, International Standard Organization (1995).

[28] R. Bagnara, E. Zaffanella, P. M. Hill, Enhanced sharing analysis techniques: A comprehensive evaluation, in: M. Gabbrielli, F. Pfenning (Eds.), Proceedings of the 2nd International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, Association for Computing Machinery, Montreal, Canada, 2000, pp. 103–114.

[29] R. Bagnara, E. Zaffanella, P. M. Hill, Enhanced sharing analysis techniques: A comprehensive evaluation, Theory and Practice of Logic Programming 5 (1&2), to appear.

[30] A. Cortesi, B. Le Charlier, P. Van Hentenryck, Combinations of abstract domains for logic programming: Open product and generic pattern construction, Science of Computer Programming 38 (1–3) (2000) 27–71.

[31] P. Cousot, R. Cousot, Abstract interpretation and applications to logic programs, Journal of Logic Programming 13 (2&3) (1992) 103–179.

[32] P. M. Hill, E. Zaffanella, R. Bagnara, A correct, precise and efficient integration of set-sharing, freeness and linearity for the analysis of finite and rational tree languages, Theory and Practice of Logic Programming 4 (3) (2004) 289–323, to appear.

[33] E. Zaffanella, Correctness, precision and efficiency in the sharing analysis of real logic languages, Ph.D. thesis, School of Computing, University of Leeds, Leeds, U.K., available at `http://www.cs.unipr.it/~zaffanella/` (2001).

[34] R. Bagnara, R. Gori, P. M. Hill, E. Zaffanella, Finite-tree analysis for constraint logic-based languages, in: P. Cousot (Ed.), Static Analysis: 8th International Symposium, SAS 2001, Vol. 2126 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Paris, France, 2001, pp. 165–184.

[35] R. Bagnara, E. Zaffanella, R. Gori, P. M. Hill, Boolean functions for finite-tree dependencies, in: R. Nieuwenhuis, A. Voronkov (Eds.), Proceedings of the 8th

International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2001), Vol. 2250 of Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, Havana, Cuba, 2001, pp. 579–594.

[36] A. Berarducci, M. Venturini Zilli, Generalizations of unification, Journal of Symbolic Computation 15 (1993) 479–491.

[37] A. King, Pair-sharing over rational trees, Journal of Logic Programming 46 (1–2) (2000) 139–155.

[38] A. Colmerauer, Equations and inequations on finite and infinite trees, in: Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS'84), ICOT, Tokyo, Japan, 1984, pp. 85–99.

[39] J. Jaffar, J.-L. Lassez, M. J. Maher, Prolog-II as an instance of the logic programming scheme, in: M. Wirsing (Ed.), Formal Descriptions of Programming Concepts III, North-Holland, Amsterdam, 1987, pp. 275–299.

[40] T. Keisu, Tree constraints, Ph.D. thesis, The Royal Institute of Technology, Stockholm, Sweden, also available in the SICS Dissertation Series: SICS/D–16–SE (May 1994).

[41] M. J. Maher, Complete axiomatizations of the algebras of finite, rational and infinite trees, in: Proceedings, Third Annual Symposium on Logic in Computer Science, IEEE Computer Society Press, Edinburgh, Scotland, 1988, pp. 348–357.

[42] T. Armstrong, K. Marriott, P. Schachte, H. Søndergaard, Two classes of Boolean functions for dependency analysis, Science of Computer Programming 31 (1) (1998) 3–45.

[43] K. Marriott, H. Søndergaard., Notes for a tutorial on abstract interpretation of logic programs, North American Conference on Logic Programming, Cleveland, Ohio, USA (1989).

[44] A. Cortesi, G. Filé, W. Winsborough, *Prop* revisited: Propositional formula as abstract domain for groundness analysis, in: Proceedings, Sixth Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, Amsterdam, The Netherlands, 1991, pp. 322–327.

[45] K. Marriott, H. Søndergaard, Precise and efficient groundness analysis for logic programs, ACM Letters on Programming Languages and Systems 2 (1–4) (1993) 181–196.

[46] E. Schröder, Der Operationskreis des Logikkalkuls, B. G. Teubner, Leibzig, 1877.

[47] P. Cousot, R. Cousot, Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages, ACM Press, New York, 1977, pp. 238–252.

[48] P. Cousot, R. Cousot, Abstract interpretation frameworks, Journal of Logic and Computation 2 (4) (1992) 511–547.

[49] R. Bagnara, P. M. Hill, E. Zaffanella, Efficient structural information analysis for real CLP languages, in: M. Parigot, A. Voronkov (Eds.), Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning (LPAR 2000), Vol. 1955 of Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, Réunion Island, France, 2000, pp. 189–206.

[50] L. Crnogorac, A. D. Kelly, H. Søndergaard, A comparison of three occur-check analysers, in: R. Cousot, D. A. Schmidt (Eds.), Static Analysis: Proceedings of the 3rd International Symposium, Vol. 1145 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Aachen, Germany, 1996, pp. 159–173.

[51] H. Søndergaard, An application of abstract interpretation of logic programs: Occur check reduction, in: B. Robinet, R. Wilhelm (Eds.), Proceedings of the 1986 European Symposium on Programming, Vol. 213 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Saarbrücken, Federal Republic of Germany, 1986, pp. 327–338.

[52] M. Bruynooghe, M. Codish, A. Mulkers, Abstract unification for a composite domain deriving sharing and freeness properties of program variables, in: F. S. de Boer, M. Gabbrielli (Eds.), Verification and Analysis of Logic Languages, Proceedings of the W2 Post-Conference Workshop, International Conference on Logic Programming, Santa Margherita Ligure, Italy, 1994, pp. 213–230.

[53] W. Hans, S. Winkler, Aliasing and groundness analysis of logic programs through abstract interpretation and its safety, Tech. Rep. 92–27, Technical University of Aachen (RWTH Aachen) (1992).

[54] P. M. Hill, R. Bagnara, E. Zaffanella, Soundness, idempotence and commutativity of set-sharing, Theory and Practice of Logic Programming 2 (2) (2002) 155–201.

[55] D. Jacobs, A. Langen, Accurate and efficient approximation of variable aliasing in logic programs, in: E. L. Lusk, R. A. Overbeek (Eds.), Logic Programming: Proceedings of the North American Conference, MIT Press Series in Logic Programming, The MIT Press, Cleveland, Ohio, USA, 1989, pp. 154–165.

[56] A. Cortesi, G. Filé, Sharing is optimal, Journal of Logic Programming 38 (3) (1999) 371–386.

[57] P. M. Hill, E. Zaffanella, R. Bagnara, A correct, precise and efficient integration of set-sharing, freeness and linearity for the analysis of finite and rational tree languages, Quaderno 273, Dipartimento di Matematica, Università di Parma, Italy, available at `http://www.cs.unipr.it/Publications/`. Also published as technical report No. 2001.22, School of Computing, University of Leeds, U.K. (2001).

[58] P. W. Dart, On derived dependencies and connected databases, Journal of Logic Programming 11 (1&2) (1991) 163–188.

[59] R. Bagnara, P. Schachte, Factorizing equivalent variable pairs in ROBDD-based implementations of *Pos*, in: A. M. Haeberer (Ed.), Proceedings of the "Seventh International Conference on Algebraic Methodology and Software Technology (AMAST'98)", Vol. 1548 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Amazonia, Brazil, 1999, pp. 471–485.

[60] R. E. Bryant, Symbolic boolean manipulation with ordered binary-decision diagrams, ACM Computing Surveys 24 (3) (1992) 293–318.

[61] D. Jacobs, A. Langen, Static analysis of logic programs for independent AND parallelism, Journal of Logic Programming 13 (2&3) (1992) 291–314.

[62] M. Codish, H. Søndergaard, P. J. Stuckey, Sharing and groundness dependencies in logic programs, ACM Transactions on Programming Languages and Systems 21 (5) (1999) 948–976.

[63] A. Cortesi, G. Filé, W. Winsborough, The quotient of an abstract interpretation for comparing static analyses, Theoretical Computer Science 202 (1&2) (1998) 163–192.

[64] R. Bagnara, Data-flow analysis for constraint logic-based languages, Ph.D. thesis, Dipartimento di Informatica, Università di Pisa, Pisa, Italy, printed as Report TD-1/97 (Mar. 1997).

[65] R. Ramakrishnan, Magic Templates: A spellbinding approach to logic programs, in: R. A. Kowalski, K. A. Bowen (Eds.), Logic Programming: Proceedings of the Fifth International Conference and Symposium on Logic Programming, MIT Press Series in Logic Programming, The MIT Press, Seattle, USA, 1988, pp. 140–159.

[66] F. Bourdoncle, Efficient chaotic iteration strategies with widenings, in: D. Bjørner, M. Broy, I. V. Pottosin (Eds.), Proceedings of the International Conference on "Formal Methods in Programming and Their Applications", Vol. 735 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Academgorodok, Novosibirsk, Russia, 1993, pp. 128–141.

[67] F. Bourdoncle, Sémantiques des langages impératifs d'ordre supérieur et interprétation abstraite, PRL Research Report 22, DEC Paris Research Laboratory (1993).

[68] J. A. Campbell (Ed.), Implementations of Prolog, Ellis Horwood/Halsted Press/Wiley, 1984.

[69] R. Bagnara, P. M. Hill, E. Zaffanella, Set-sharing is redundant for pair-sharing, Theoretical Computer Science 277 (1-2) (2002) 3–46.

[70] E. Zaffanella, P. M. Hill, R. Bagnara, Decomposing non-redundant sharing by complementation, Theory and Practice of Logic Programming 2 (2) (2002) 233–261.

[71] M. Codish, D. Dams, E. Yardeni, Derivation and safety of an abstract unification algorithm for groundness and aliasing analysis, in: K. Furukawa

(Ed.), Logic Programming: Proceedings of the Eighth International Conference on Logic Programming, MIT Press Series in Logic Programming, The MIT Press, Paris, France, 1991, pp. 79–93.

[72] F. Scozzari, Abstract domains for sharing analysis by optimal semantics, in: J. Palsberg (Ed.), Static Analysis: 7th International Symposium, SAS 2000, Vol. 1824 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Santa Barbara, CA, USA, 2000, pp. 397–412.

[73] K. L. Clark, Negation as failure, in: H. Gallaire, J. Minker (Eds.), Logic and Databases, Plenum Press, Toulouse, France, 1978, pp. 293–322.

[74] P. M. Hill, R. Bagnara, E. Zaffanella, The correctness of set-sharing, in: G. Levi (Ed.), Static Analysis: Proceedings of the 5th International Symposium, Vol. 1503 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Pisa, Italy, 1998, pp. 99–114.

A An Instance of the Parameter Domain P

As discussed in Section 4, several abstract domains for sharing analysis can be used to implement the parameter component P. We here consider the abstract domain SFL [32,33], integrating the set-sharing domain of Jacobs and Langen with definite freeness and linearity information.

Definition 35 (The set-sharing domain SH.) *The set* SH *is defined by* $\mathrm{SH} \stackrel{\mathrm{def}}{=} \wp(\mathrm{SG})$*, where* $\mathrm{SG} \stackrel{\mathrm{def}}{=} \wp(\mathrm{VI}) \setminus \{\varnothing\}$ *is the set of* sharing groups. SH *is ordered by subset inclusion.*

The information about definite freeness and linearity is encoded by two sets of variables, one for each property.

Definition 36 (The domain SFL.) *Let* $F \stackrel{\mathrm{def}}{=} \wp(\mathrm{VI})$ *and* $L \stackrel{\mathrm{def}}{=} \wp(\mathrm{VI})$ *be partially ordered by reverse subset inclusion. The domain* SFL *is defined by the Cartesian product* $\mathrm{SFL} \stackrel{\mathrm{def}}{=} \mathrm{SH} \times F \times L$ *ordered by '*$\leq_S$*', the component-wise extension of the orderings defined on the sub-domains; the bottom element is* $\bot_S \stackrel{\mathrm{def}}{=} \langle \varnothing, \mathrm{VI}, \mathrm{VI} \rangle$.

In the next definition we introduce a few well-known operations on the set-sharing domain SH. These will be used to define the operations on the domain SFL.

Definition 37 (Abstract operators on SH.) *For each* $\mathrm{sh} \in \mathrm{SH}$ *and each* $V \subseteq \mathrm{VI}$*, the extraction of the* relevant component of sh with respect to V *is given by the function* $\mathrm{rel} \colon \wp(\mathrm{VI}) \times \mathrm{SH} \to \mathrm{SH}$ *defined as*

$$\mathrm{rel}(V, \mathrm{sh}) \stackrel{\mathrm{def}}{=} \{\, S \in \mathrm{sh} \mid S \cap V \neq \varnothing \,\}.$$

For each $\mathrm{sh} \in \mathrm{SH}$ *and each* $V \subseteq \mathrm{VI}$*, the function* $\overline{\mathrm{rel}} \colon \wp(\mathrm{VI}) \times \mathrm{SH} \to \mathrm{SH}$ *gives the* irrelevant component of sh with respect to V. *It is defined as*

$$\overline{\mathrm{rel}}(V, \mathrm{sh}) \stackrel{\mathrm{def}}{=} \mathrm{sh} \setminus \mathrm{rel}(V, \mathrm{sh}).$$

The function $(\cdot)^\star \colon \mathrm{SH} \to \mathrm{SH}$*, called* star-union*, is given, for each* $\mathrm{sh} \in \mathrm{SH}$*, by*

$$\mathrm{sh}^\star \stackrel{\mathrm{def}}{=} \left\{ S \in \mathrm{SG} \;\middle|\; \exists n \geq 1 \,.\, \exists T_1, \ldots, T_n \in \mathrm{sh} \,.\, S = \bigcup_{i=1}^{n} T_i \right\}.$$

For each $\mathrm{sh}_1, \mathrm{sh}_2 \in \mathrm{SH}$*, the function* $\mathrm{bin} \colon \mathrm{SH} \times \mathrm{SH} \to \mathrm{SH}$*, called* binary union*, is given by*

$$\mathrm{bin}(\mathrm{sh}_1, \mathrm{sh}_2) \stackrel{\mathrm{def}}{=} \{\, S_1 \cup S_2 \mid S_1 \in \mathrm{sh}_1, S_2 \in \mathrm{sh}_2 \,\}.$$

For each $\mathrm{sh} \in \mathrm{SH}$ *and each* $(x \mapsto t) \in \mathrm{Bind}$*, the function* $\mathrm{cyclic}_x^t \colon \mathrm{SH} \to \mathrm{SH}$ *strengthens the sharing set* sh *by forcing the coupling of* x *with* t*:*

$$\mathrm{cyclic}_x^t(\mathrm{sh}) \stackrel{\mathrm{def}}{=} \overline{\mathrm{rel}}\Big(\{x\} \cup \mathrm{vars}(t), \mathrm{sh}\Big) \cup \mathrm{rel}\Big(\mathrm{vars}(t) \setminus \{x\}, \mathrm{sh}\Big).$$

For each $\mathrm{sh} \in \mathrm{SH}$ *and each* $x \in \mathrm{VI}$*, the function* $\mathrm{proj}_{\mathrm{SH}} \colon \mathrm{SH} \times \mathrm{VI} \to \mathrm{SH}$ *projects away variable* x *from* sh*:*

$$\mathrm{proj}_{\mathrm{SH}}(\mathrm{sh}, x) \stackrel{\mathrm{def}}{=} \Big\{\{x\}\Big\} \cup \Big\{\, S \setminus \{x\} \;\Big|\; S \in \mathrm{sh}, S \neq \{x\} \,\Big\}.$$

It is now possible to define the implementation, on the domain SFL, of all the predicates and functions specified in Definition 8.

Definition 38 (Abstract operators on SFL.**)** *For each* $\mathrm{d} \in \mathrm{SFL}$ *and* $s, t \in \mathrm{HTerms}$*, where* $\mathrm{d} = \langle \mathrm{sh}, f, l \rangle$ *and* $\mathrm{vars}(s) \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$*, let* $\mathrm{sh}_s = \mathrm{rel}\Big(\mathrm{vars}(s), \mathrm{sh}\Big)$ *and* $\mathrm{sh}_t = \mathrm{rel}\Big(\mathrm{vars}(t), \mathrm{sh}\Big)$*. Then*

$$\begin{aligned}
\mathrm{ind}_{\mathrm{d}}(s,t) &\stackrel{\mathrm{def}}{=} \Big(\mathrm{sh}_s \cap \mathrm{sh}_t = \varnothing\Big);\\
\mathrm{ground}_{\mathrm{d}}(t) &\stackrel{\mathrm{def}}{=} \Big(\mathrm{vars}(t) \subseteq \mathrm{VI} \setminus \mathrm{vars}(\mathrm{sh})\Big);\\
\mathrm{occ_lin}_{\mathrm{d}}(y,t) &\stackrel{\mathrm{def}}{=} \mathrm{ground}_{\mathrm{d}}(y) \lor \Big(\mathrm{occ_lin}(y,t) \land (y \in l)\\
&\qquad\qquad \land \forall z \in \mathrm{vars}(t) : \Big(y \neq z \implies \mathrm{ind}_{\mathrm{d}}(y,z)\Big)\Big);\\
\mathrm{share_lin}_{\mathrm{d}}(s,t) &\stackrel{\mathrm{def}}{=} \forall y \in \mathrm{vars}(\mathrm{sh}_s \cap \mathrm{sh}_t) :\\
&\qquad y \in \mathrm{vars}(s) \implies \mathrm{occ_lin}_{\mathrm{d}}(y,s)\\
&\qquad \land\, y \in \mathrm{vars}(t) \implies \mathrm{occ_lin}_{\mathrm{d}}(y,t);\\
\mathrm{free}_{\mathrm{d}}(t) &\stackrel{\mathrm{def}}{=} \exists y \in \mathrm{VI} \,.\, (y = t) \land (y \in f);\\
\mathrm{gfree}_{\mathrm{d}}(t) &\stackrel{\mathrm{def}}{=} \mathrm{ground}_{\mathrm{d}}(t) \lor \mathrm{free}_{\mathrm{d}}(t);\\
\mathrm{lin}_{\mathrm{d}}(t) &\stackrel{\mathrm{def}}{=} \forall y \in \mathrm{vars}(t) : \mathrm{occ_lin}_{\mathrm{d}}(y,t);\\
\mathrm{or_lin}_{\mathrm{d}}(s,t) &\stackrel{\mathrm{def}}{=} \mathrm{lin}_{\mathrm{d}}(s) \lor \mathrm{lin}_{\mathrm{d}}(t);\\
\mathrm{share_same_var}_{\mathrm{d}}(s,t) &\stackrel{\mathrm{def}}{=} \mathrm{vars}(\mathrm{sh}_s \cap \mathrm{sh}_t);\\
\mathrm{share_with}_{\mathrm{d}}(t) &\stackrel{\mathrm{def}}{=} \mathrm{vars}(\mathrm{sh}_t).
\end{aligned}$$

The function $\mathrm{amgu}_S \colon \mathrm{SFL} \times \mathrm{Bind} \to \mathrm{SFL}$ *captures the effects of a binding on an element of* SFL*. Let* $\mathrm{d} = \langle \mathrm{sh}, f, l \rangle \in \mathrm{SFL}$ *and* $(x \mapsto t) \in \mathrm{Bind}$*, where* $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$*. Let also*

$$\mathrm{sh}' \stackrel{\mathrm{def}}{=} \mathrm{cyclic}_x^t(\mathrm{sh}_- \cup \mathrm{sh}''),$$

where

$$\mathrm{sh}_x \stackrel{\mathrm{def}}{=} \mathrm{rel}\Big(\{x\}, \mathrm{sh}\Big), \qquad \mathrm{sh}_t \stackrel{\mathrm{def}}{=} \mathrm{rel}\Big(\mathrm{vars}(t), \mathrm{sh}\Big),$$
$$\mathrm{sh}_{xt} \stackrel{\mathrm{def}}{=} \mathrm{sh}_x \cap \mathrm{sh}_t, \qquad \mathrm{sh}_- \stackrel{\mathrm{def}}{=} \overline{\mathrm{rel}}\Big(\{x\} \cup \mathrm{vars}(t), \mathrm{sh}\Big),$$

$$\mathrm{sh}'' \stackrel{\mathrm{def}}{=} \begin{cases} \mathrm{bin}(\mathrm{sh}_x, \mathrm{sh}_t), & \textit{if } \mathrm{free_d}(x) \vee \mathrm{free_d}(t); \\ \mathrm{bin}\Big(\mathrm{sh}_x \cup \mathrm{bin}(\mathrm{sh}_x, \mathrm{sh}_{xt}^\star), & \\ \qquad\qquad \mathrm{sh}_t \cup \mathrm{bin}(\mathrm{sh}_t, \mathrm{sh}_{xt}^\star)\Big), & \textit{if } \mathrm{lin_d}(x) \wedge \mathrm{lin_d}(t); \\ \mathrm{bin}(\mathrm{sh}_x^\star, \mathrm{sh}_t), & \textit{if } \mathrm{lin_d}(x); \\ \mathrm{bin}(\mathrm{sh}_x, \mathrm{sh}_t^\star), & \textit{if } \mathrm{lin_d}(t); \\ \mathrm{bin}(\mathrm{sh}_x^\star, \mathrm{sh}_t^\star), & \textit{otherwise.} \end{cases}$$

Letting $S_x \stackrel{\mathrm{def}}{=} \mathrm{share\_with_d}(x)$ *and* $S_t \stackrel{\mathrm{def}}{=} \mathrm{share\_with_d}(t)$*, we also define*

$$f' \stackrel{\mathrm{def}}{=} \begin{cases} f, & \textit{if } \mathrm{free_d}(x) \wedge \mathrm{free_d}(t); \\ f \setminus S_x, & \textit{if } \mathrm{free_d}(x); \\ f \setminus S_t, & \textit{if } \mathrm{free_d}(t); \\ f \setminus (S_x \cup S_t), & \textit{otherwise;} \end{cases}$$
$$l' \stackrel{\mathrm{def}}{=} \Big(\mathrm{VI} \setminus \mathrm{vars}(\mathrm{sh}')\Big) \cup f' \cup l'',$$

where

$$l'' \stackrel{\mathrm{def}}{=} \begin{cases} l \setminus (S_x \cap S_t), & \textit{if } \mathrm{lin_d}(x) \wedge \mathrm{lin_d}(t); \\ l \setminus S_x, & \textit{if } \mathrm{lin_d}(x); \\ l \setminus S_t, & \textit{if } \mathrm{lin_d}(t); \\ l \setminus (S_x \cup S_t), & \textit{otherwise.} \end{cases}$$

Then

$$\mathrm{amgu}_S\Big(\mathrm{d}, x \mapsto t\Big) \stackrel{\mathrm{def}}{=} \langle \mathrm{sh}', f', l' \rangle.$$

The function $\mathrm{proj}_S \colon \mathrm{SFL} \times \mathrm{VI} \to \mathrm{SFL}$ *correctly captures the operation of projecting away a variable from an element of* SFL*. For each* $\mathrm{d} \in \mathrm{SFL}$ *and* $x \in \mathrm{VI}$*,*

$$\mathrm{proj}_S(\mathrm{d}, x) \stackrel{\mathrm{def}}{=} \begin{cases} \bot_S, & \textit{if } \mathrm{d} = \bot_S; \\ \Big\langle \mathrm{proj_{SH}}(\mathrm{sh}, x), f \cup \{x\}, l \cup \{x\} \Big\rangle, & \textit{if } \mathrm{d} = \langle \mathrm{sh}, f, l \rangle \neq \bot_S. \end{cases}$$

Observe that a set-sharing domain such as SFL is strictly more precise for term finiteness information than a pair-sharing domain such as SFL_2 [32,33] (where the set-sharing component SH in SFL is replaced by the domain PSD as defined in [69,70]). To see this, consider the abstract evaluation of the binding $x \mapsto y$ and the description $\langle h, \mathrm{d} \rangle \in H \times \mathrm{SFL}$, where $h = \{x, y, z\}$

and $\mathrm{d} = \langle \mathrm{sh}, f, l \rangle$ is such that $\mathrm{sh} = \bigl\{\{x,y\},\{x,z\},\{y,z\}\bigr\}$, $f = \varnothing$ and $l = \{x,y,z\}$. Then $z \notin \mathrm{share\_same\_var}_{\mathrm{d}}(x,y)$ so that we have $h' = \{z\}$. In contrast, when using a pair sharing domain such as SFL_2 the element d is equivalent to $\mathrm{d}' = \langle \mathrm{sh}', f, l \rangle$, where $\mathrm{sh}' = \mathrm{sh} \cup \bigl\{\{x,y,z\}\bigr\}$. Hence we have $z \in \mathrm{share\_same\_var}_{\mathrm{d}'}(x,y)$ and $h' = \varnothing$. Thus, in sh the information provided by the sharing group $\{x,y,z\}$ is redundant for the pair-sharing and groundness properties, but not redundant for term finiteness. Note that the above observation holds regardless of the pair-sharing variant considered, so that similar examples can be obtained for ASub [71,51] and $\mathsf{Sh}^{\mathsf{PSh}}$ [72].

Although the domain SFL described here is very precise and used to implement the parameter component P for computing our experimental results, it is not intended as the target of the generic specification given in Definition 8; more powerful sharing domains can also satisfy this schema, including all the enhanced combinations considered in [28,29]. For instance, as the predicate $\mathrm{gfree}_{\mathrm{d}}$ defined on SFL does not fully exploit the disjunctive nature of its generic specification gfree_p, the precision of the analysis may be improved by adding a domain component explicitly tracking ground-or-freeness, as proposed in [28,29]. The same argument applies to the predicate $\mathrm{or\_lin}_{\mathrm{d}}$, with respect to $\mathrm{or\_lin}_p$, when considering the combination with the groundness domain Pos.

B Proofs of the Stated Results

This appendix provides the proofs of the results stated in the paper. Section B.1 introduces the notations and preliminary concepts that are subsequently used in the proofs. In Section B.2 we recall a few general results holding for (syntactic) equality theories and provide the proof of Proposition 2. The definition of (strongly) variable idempotent substitutions is given in Section B.3, together with some properties holding for them; these are then used in Section B.4 to prove some general results on operators on substitutions in RSubst, Propositions 13 and 15. Section B.4 is propaedeutic to Section B.5, where we prove Theorem 17 and to Section B.6, where we provide the proofs of Theorems 19 and 21. Results in Section B.4 are then used in Section B.7 to prove Theorems 23, 25 and 27, and in Section B.8 to prove Theorems 30 and 34.

B.1 Notations and Preliminaries for the Proofs

To simplify the expressions in the paper, any variable in a formula that is not in the scope of an explicit quantifier is assumed to be universally quantified.

A path $p \in \big(\mathbb{N} \setminus \{0\}\big)^\star$ is any finite sequence of non-zero natural numbers. The empty path is denoted by ϵ, whereas $i \,.\, p$ denotes the path obtained by concatenating the sequence formed by the natural number $i \neq 0$ with the sequence of the path p. Given a path p and a (possibly infinite) term $t \in \mathrm{Terms}$, we denote by $t[p]$ the subterm of t found by following path p. Formally,

$$t[p] = \begin{cases} t & \text{if } p = \epsilon; \\ t_i[q] & \text{if } p = i \,.\, q \wedge (1 \leq i \leq n) \wedge t = f(t_1, \ldots, t_n). \end{cases}$$

Note that $t[p]$ is only defined for those paths p actually corresponding to subterms of t.

The function $\mathrm{size}\colon \mathrm{HTerms} \to \mathbb{N}$ is defined, for each $t \in \mathrm{HTerms}$, by

$$\mathrm{size}(t) \stackrel{\mathrm{def}}{=} \begin{cases} 1, & \text{if } t \in \mathrm{Vars}; \\ 1 + \sum_{i=1}^{n} \mathrm{size}(t_i), & \text{if } t = f(t_1, \ldots, t_n), \text{ where } n \geq 0. \end{cases}$$

A substitution σ is *idempotent* if, for all $t \in \mathrm{HTerms}$, we have $t\sigma\sigma = t\sigma$. The set of all idempotent substitutions is denoted by ISubst and $\mathrm{ISubst} \subset \mathrm{RSubst}$.

If $t \in \mathrm{HTerms}$, we denote the set of variables that occur more than once in t by:

$$\mathrm{nlvars}(t) \stackrel{\mathrm{def}}{=} \Big\{\, y \in \mathrm{vars}(t) \;\Big|\; \neg\mathrm{occ\_lin}(y, t) \,\Big\}.$$

If $\bar{s} = (s_1, \ldots, s_n) \in \mathrm{HTerms}^n$ and $\bar{t} = (t_1, \ldots, t_n) \in \mathrm{HTerms}^n$ are two tuples of finite terms, then we let $\bar{s} = \bar{t}$ denote the set of equations between corresponding components of $\bar{s}$ and $\bar{t}$. Namely,

$$(\bar{s} = \bar{t}) \stackrel{\mathrm{def}}{=} \{\, s_i = t_i \mid 1 \leq i \leq n \,\}.$$

Moreover, we overload the functions mvars, occ_lin and nlvars to work on tuples of terms; thus, we will say that $\bar{s}$ is linear if and only if $\mathrm{nlvars}(\bar{s}) = \varnothing$.

B.1.1 Equality Theories

Let $\{s, t, s_1, \ldots, s_n, t_1, \ldots, t_m\} \subseteq \mathrm{HTerms}$. We assume that any equality theory T over Terms includes the *congruence axioms* denoted by the following schemata:

$$s = s, \tag{B.1}$$
$$s = t \leftrightarrow t = s, \tag{B.2}$$
$$r = s \wedge s = t \to r = t, \tag{B.3}$$
$$s_1 = t_1 \wedge \cdots \wedge s_n = t_n \to f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n). \tag{B.4}$$

In logic programming and most implementations of Prolog it is usual to assume an equality theory based on syntactic identity. This consists of the congruence axioms together with the *identity axioms* denoted by the following schemata, where f and g are distinct function symbols or $n \neq m$:

$$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n) \rightarrow s_1 = t_1 \wedge \cdots \wedge s_n = t_n, \tag{B.5}$$

$$\neg\Bigl(f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m)\Bigr). \tag{B.6}$$

The axioms characterized by schemata (B.5) and (B.6) ensure the equality theory depends only on the syntax. The equality theory for a non-syntactic domain replaces these axioms by ones that depend instead on the semantics of the domain and, in particular, on the interpretation given to functor symbols.

The equality theory of Clark [73] on which pure logic programming is based, usually called the *Herbrand* equality theory and denoted $\mathcal{FT}$, is given by the congruence axioms, the identity axioms, and the axiom schema

$$\forall z \in \mathrm{Vars} : \forall t \in (\mathrm{HTerms} \setminus \mathrm{Vars}) : z \in \mathrm{vars}(t) \rightarrow \neg(z = t). \tag{B.7}$$

Axioms characterized by the schema (B.7) are called the *occurs-check axioms* and are an essential part of the standard unification procedure in SLD-resolution.

An alternative approach used in some implementations of Prolog, does not require the occurs-check axioms. This approach is based on the theory of rational trees $\mathcal{RT}$ [1,38]. It assumes the congruence axioms and the identity axioms together with a *uniqueness axiom* for each substitution in rational solved form. Informally speaking these state that, after assigning a ground rational tree to each parameter variable, the substitution uniquely defines a ground rational tree for each of its domain variables.

In the sequel we will use the expression "equality theory" to denote any consistent, decidable theory T satisfying the congruence axioms. We will also use the expression "syntactic equality theory" to denote any equality theory T also satisfying the identity axioms. [17] Note that both $\mathcal{FT}$ and $\mathcal{RT}$ are syntactic equality theories. When the equality theory T is clear from the context, it is convenient to adopt the notations $\sigma \implies \tau$ and $\sigma \iff \tau$, where σ, τ are sets of equations, to denote $T \vdash \forall(\sigma \rightarrow \tau)$ and $T \vdash \forall(\sigma \leftrightarrow \tau)$, respectively.

Given an equality theory T, and a set of equations in rational solved form σ, we say that σ is *satisfiable* in T if $T \vdash \forall \mathrm{Vars} \setminus \mathrm{dom}(\sigma) : \exists\, \mathrm{dom}(\sigma) \,.\, \sigma$. Observe

[17] Note that, as a consequence of axiom (B.6) and the assumption that there are at least two distinct function symbols in the language, one of which is a constant, there exist two terms $a_1, a_2 \in \mathrm{GTerms} \cap \mathrm{HTerms}$ such that, for any syntactic equality theory T, we have $T \vdash a_1 \neq a_2$.

that, given an arbitrary equality theory T, a substitution in RSubst may not be satisfiable in T. For example, $\exists x \,.\, \bigl\{x = f(x)\bigr\}$ is false in the Clark equality theory. However, as every element of RSubst satisfies the identity axioms as well as the axioms (B.5) and (B.6) and, as the uniqueness axioms do not affect satisfiability, every element of RSubst is satisfiable in $\mathcal{RT}$.

B.2 Properties of Equality Theories

Proof of Proposition 2 on page 10. Suppose $\tau \in \downarrow\sigma$. Then, by Definition 1, for some $\upsilon \in \mathrm{RSubst}$, $\tau \in \mathrm{mgs}(\sigma \cup \upsilon)$. and hence $\mathcal{RT} \vdash \forall\bigl(\tau \leftrightarrow (\sigma \cup \upsilon)\bigr)$. Therefore $\mathcal{RT} \vdash \forall(\tau \to \sigma)$.

Conversely, suppose $\mathcal{RT} \vdash \forall(\tau \to \sigma)$. Then $\mathcal{RT} \vdash \forall\bigl(\tau \to (\sigma \cup \tau)\bigr)$ so that, as $\vdash \forall\bigl((\sigma \cup \tau) \to \tau\bigr)$, we have $\mathcal{RT} \vdash \forall\bigl(\tau \leftrightarrow \sigma \cup \tau\bigr)$. Therefore $\tau \in \mathrm{mgs}(\sigma \cup \tau)$ so that, by Definition 1, $\tau \in \downarrow\sigma$. □

We now prove a number of results about substitutions in RSubst, assuming suitable equality theories, that will be used in the proofs of our main results.

Lemma 39 *Let $\sigma \in \mathrm{RSubst}$ and $\{x \mapsto t\} \in \mathrm{RSubst}$ be both satisfiable in the equality theory T, where $x \notin \mathrm{dom}(\sigma)$ and $\mathrm{vars}(t) \cap \mathrm{dom}(\sigma) = \varnothing$. Define also $\sigma' \stackrel{\mathrm{def}}{=} \sigma \cup \{x \mapsto t\}$. Then $\sigma' \in \mathrm{RSubst}$ and σ' is satisfiable in T.*

PROOF. Note that σ' is a substitution, since $\sigma \in \mathrm{RSubst}$ and $x \notin \mathrm{dom}(\sigma)$. Moreover, as $\mathrm{vars}(t) \cap \mathrm{dom}(\sigma) = \varnothing$, σ' cannot contain circular subsets. Hence, $\sigma' \in \mathrm{RSubst}$.

Since both σ and $\{x \mapsto t\}$ are satisfiable in T, we have

$$T \vdash \forall \mathrm{Vars} \setminus \mathrm{dom}(\sigma) : \exists\, \mathrm{dom}(\sigma) \,.\, \sigma,$$
$$T \vdash \forall \mathrm{Vars} \setminus \{x\} : \exists x \,.\, \{x = t\}.$$

Letting $V = \mathrm{Vars} \setminus \bigl(\mathrm{dom}(\sigma) \cup \{x\}\bigr)$, we can rewrite these as

$$T \vdash \forall V : \forall x : \exists\, \mathrm{dom}(\sigma) \,.\, \sigma, \tag{B.8}$$
$$T \vdash \forall V : \forall\, \mathrm{dom}(\sigma) : \exists x \,.\, \{x = t\}. \tag{B.9}$$

Then, as $\mathrm{vars}(x = t) \cap \mathrm{dom}(\sigma) = \varnothing$, it follows from (B.9) that

$$T \vdash \forall V : \exists x \,.\, \{x = t\}.$$

Combining this with (B.8) gives

$$T \vdash \forall V : \Big(\big(\forall x : \exists\,\mathrm{dom}(\sigma) \,.\, \sigma\big) \wedge \big(\exists x \,.\, \{x = t\}\big)\Big).$$

Thus we have

$$T \vdash \forall V : \exists x \,.\, \big(\exists\,\mathrm{dom}(\sigma) \,.\, \sigma \wedge \{x = t\}\big),$$

and hence, as $\mathrm{vars}(x = t) \cap \mathrm{dom}(\sigma) = \varnothing$,

$$T \vdash \forall V : \exists x \,.\, \exists\,\mathrm{dom}(\sigma) \,.\, \big(\sigma \wedge \{x = t\}\big).$$

Therefore,

$$T \vdash \forall V : \exists\big(\mathrm{dom}(\sigma) \cup \{x\}\big) \,.\, \sigma \cup \{x = t\}.$$

Thus σ' is satisfiable in T. $\square$

Lemma 40 *Assume T is an equality theory and $\sigma \in \mathrm{RSubst}$. Then, for each $t \in \mathrm{HTerms}$,*

$$T \vdash \forall\big(\sigma \rightarrow (t = t\sigma)\big).$$

PROOF. Proved in [54, Lemma 2]. $\square$

Lemma 41 *Assume T is an equality theory and $\sigma \in \mathrm{RSubst}$. Then, for each $s, t \in \mathrm{HTerms}$,*

$$T \vdash \forall\big(\sigma \cup \{s = t\} \leftrightarrow \sigma \cup \{s = t\sigma\}\big).$$

PROOF. First, note, using the congruence axioms (B.2) and (B.3), that, for any terms $p, q, r \in \mathrm{HTerms}$,

$$T \vdash \forall(p = q \wedge q = r) \leftrightarrow \forall(p = r \wedge q = r). \tag{B.10}$$

Secondly note that, using Lemma 40, for any substitution $\tau \in \mathrm{RSubst}$ and term $r \in \mathrm{HTerms}$, $T \vdash \forall\big(\tau \rightarrow (r = r\tau)\big)$. Thus

$$T \vdash \forall\big(\tau \leftrightarrow \tau \cup \{r = r\tau\}\big). \tag{B.11}$$

Using these results, we obtain

$$T \vdash \forall\big(\sigma \cup \{s = t\} \leftrightarrow \sigma \cup \{s = t, t = t\sigma\}\big), \qquad \text{[by (B.11)]}$$

$$T \vdash \forall\Big(\sigma \cup \{s = t\} \leftrightarrow \sigma \cup \{s = t\sigma, t = t\sigma\}\Big), \qquad \text{[by (B.10)]}$$
$$T \vdash \forall\Big(\sigma \cup \{s = t\} \leftrightarrow \sigma \cup \{s = t\sigma\}\Big). \qquad \text{[by (B.11)]}$$

□

Lemma 42 *Let* $\sigma \in \mathrm{RSubst}$ *be satisfiable in a syntactic equality theory* T *and* $s, t \in \mathrm{HTerms}$, *where* $T \vdash \forall\big(\sigma \to (s = t)\big)$. *Then* $\mathrm{rt}(s, \sigma) = \mathrm{rt}(t, \sigma)$.

PROOF. We suppose, towards a contradiction, that $\mathrm{rt}(s, \sigma) \neq \mathrm{rt}(t, \sigma)$. Then there exists a finite path p such that:

a. $x = \mathrm{rt}(s, \sigma)[p] \in \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$, $y = \mathrm{rt}(t, \sigma)[p] \in \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$ and $x \neq y$; or
b. $x = \mathrm{rt}(s, \sigma)[p] \in \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$ and $r = \mathrm{rt}(t, \sigma)[p] \notin \mathrm{Vars}$ or, symmetrically, $r = \mathrm{rt}(s, \sigma)[p] \notin \mathrm{Vars}$ and $x = \mathrm{rt}(t, \sigma)[p] \in \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$; or
c. $r_1 = \mathrm{rt}(s, \sigma)[p] \notin \mathrm{Vars}$, $r_2 = \mathrm{rt}(t, \sigma)[p] \notin \mathrm{Vars}$ and r_1 and r_2 have different principal functors.

Then, by definition of 'rt', there exists an index $i \in \mathbb{N}$ such that one of these holds:

1. $x = s\sigma^i[p] \in \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$, $y = t\sigma^i[p] \in \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$ and $x \neq y$; or
2. $x = s\sigma^i[p] \in \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$ and $r = t\sigma^i[p] \notin \mathrm{Vars}$ or, in a symmetrical way, $r = s\sigma^i[p] \notin \mathrm{Vars}$ and $x = t\sigma^i[p] \in \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$; or
3. $r_1 = s\sigma^i[p] \notin \mathrm{Vars}$ and $r_2 = t\sigma^i[p] \notin \mathrm{Vars}$ have different principal functors.

By Lemma 40, we have $T \vdash \forall\big(\sigma \to (s\sigma^i = t\sigma^i)\big)$; from this, since T is a syntactic equality theory, we obtain that

$$T \vdash \forall\Big(\sigma \to (s\sigma^i[p] = t\sigma^i[p])\Big). \tag{B.12}$$

We now prove that each case leads to a contradiction.

Consider case 1. Let $r_1, r_2 \in \mathrm{GTerms} \cap \mathrm{HTerms}$ be two terms having different principal functors, so that $T \vdash \forall(r_1 \neq r_2)$. Then, as σ is satisfiable in T, by Lemma 39, we have that $\sigma' = \sigma \cup \{x \mapsto r_1, y \mapsto r_2\} \in \mathrm{RSubst}$ is satisfiable in T and also $T \vdash \forall(\sigma' \to \sigma)$, $T \vdash \forall\big(\sigma' \to (x = r_1)\big)$, $T \vdash \forall\big(\sigma' \to (y = r_2)\big)$. This is a contradiction, since, by (B.12), we have $T \vdash \forall\big(\sigma \to (x = y)\big)$.

Consider case 2. Without loss of generality, consider the first subcase, where $x = s\sigma^i[p] \in \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$ and $r = t\sigma^i[p] \notin \mathrm{Vars}$. Let $r' \in \mathrm{GTerms} \cap \mathrm{HTerms}$ be such that r and r' have different principal functors, so that $T \vdash \forall(r \neq r')$. By Lemma 39, as σ is satisfiable in T, $\sigma' = \sigma \cup \{x \mapsto r'\} \in \mathrm{RSubst}$ is satisfiable in T; we also have that $T \vdash \forall(\sigma' \rightarrow \sigma)$ and $T \vdash \forall\bigl(\sigma' \rightarrow (x = r')\bigr)$. This is a contradiction as, by (B.12), $T \vdash \forall\bigl(\sigma \rightarrow (x = r)\bigr)$.

Finally, consider case 3. In this case $T \vdash \forall(r_1 \neq r_2)$. This immediately leads to a contradiction, since, by (B.12), $T \vdash \forall\bigl(\sigma \rightarrow (r_1 = r_2)\bigr)$. $\square$

Lemma 43 *Let T be a syntactic equality theory. Let $s \in \mathrm{HTerms} \cap \mathrm{GTerms}$ and $t \in \mathrm{Terms}$ be such that $\mathrm{size}(t) > \mathrm{size}(s)$. Then $T \vdash \forall(s \neq t)$.*

PROOF. By induction on $m = \mathrm{size}(s)$. For the base case, when $m = 1$, we have that s is a term functor of arity 0. Since $\mathrm{size}(t) > 1$, then $t = f(t_1, \ldots, t_n)$, where $n > 0$. Then, by the identity axioms, we have $T \vdash \forall(s \neq t)$.

For the inductive case, when $m > 1$, assume that the result holds for all $m' < m$ and let $s = f(s_1, \ldots, s_n)$, where $n > 0$. Since $\mathrm{size}(t) > m$, we have $t = f'(t_1, \ldots, t_{n'})$, where $n' > 0$. If $f \neq f'$ or $n \neq n'$ then, by the identity axioms, we have $T \vdash \forall(s \neq t)$. Otherwise, let $f = f'$ and $n = n'$. Note that, for all $i \in \{1, \ldots, n\}$, we have $\mathrm{size}(s_i) < m$. Also, there exists an index $j \in \{1, \ldots, n\}$ such that $\mathrm{size}(t_j) > \mathrm{size}(s_j)$. By the inductive hypothesis, $T \vdash \forall(s_j \neq t_j)$ so that, by the identity axioms, $T \vdash \forall(s \neq t)$. $\square$

The next two propositions establish useful properties of the function rt.

Proposition 44 *Let $\sigma \in \mathrm{RSubst}$ and $t \in \mathrm{HTerms}$. Then*

$$\mathrm{vars}\bigl(\mathrm{rt}(t, \sigma)\bigr) \cap \mathrm{dom}(\sigma) = \varnothing, \tag{44a}$$

$$\mathrm{rt}(t, \sigma) \in \mathrm{HTerms} \iff \exists i \in \mathbb{N} \,.\, \mathrm{rt}(t, \sigma) = t\sigma^i. \tag{44b}$$

PROOF.

(44a) Let $x \in \mathrm{dom}(\sigma)$ and, towards a contradiction, suppose $x \in \mathrm{vars}\bigl(\mathrm{rt}(t, \sigma)\bigr)$. Thus, there exists a finite path p such that $x = \mathrm{rt}(t, \sigma)[p]$. Thus, by definition of 'rt', there exists an index $i \in \mathbb{N}$ such that $x = \sigma^i(t)[p]$. Since $x \in \mathrm{dom}(\sigma)$, then $x \neq x\sigma$, so that $x \neq \sigma^{i+1}(t)[p]$. Also note that, since $\sigma \in \mathrm{RSubst}$, σ contains no circular subsets, so that we have $x \neq \sigma^j(t)[p]$, for each index $j > i$. This implies $x \neq \mathrm{rt}(t, \sigma)[p]$, which is a contradiction. Since no such finite path p can exist, we can conclude $x \notin \mathrm{vars}\bigl(\mathrm{rt}(t, \sigma)\bigr)$.

(44b) Since substitutions map finite terms into finite terms, a finite number of applications cannot produce an infinite term, so that the left implication holds. Proving the right implication by contraposition, suppose that $\mathrm{rt}(t, \sigma) \neq t\sigma^i$, for all $i \in \mathbb{N}$. Then, by definition of 'rt', we have $t\sigma^i \neq t\sigma^{i+1}$, for all $i \in \mathbb{N}$.

Letting $n \in \mathbb{N}$ be the number of bindings in $\sigma \in \mathrm{RSubst}$, for all $i \in \mathbb{N}$ we have that $\mathrm{size}(t\sigma^i) < \mathrm{size}(t\sigma^{i+n})$, because σ has no circular subsets. Thus $\mathrm{rt}(t, \sigma) \notin \mathrm{HTerms}$, because there is no finite upper bound to the number of function symbols occurring in $\mathrm{rt}(t, \sigma)$. $\square$

Proposition 45 *Let* $\sigma, \tau \in \mathrm{RSubst}$ *be satisfiable in a syntactic equality theory* T *and* $W \subseteq \mathrm{Vars}$, *where* $T \vdash \forall(\exists W \,.\, \tau \to \exists W \,.\, \sigma)$, *and* $x \in \mathrm{Vars} \setminus W$. *Then*

$$\mathrm{rt}(x, \tau) \in \mathrm{HTerms} \implies \mathrm{rt}(x, \sigma) \in \mathrm{HTerms}.$$

PROOF. We assume that $\mathrm{rt}(x, \tau) \in \mathrm{HTerms}$ but $\mathrm{rt}(x, \sigma) \notin \mathrm{HTerms}$ and derive a contradiction. By hypothesis $\mathrm{rt}(x, \tau) \in \mathrm{HTerms}$, so that by Proposition 44, there exists $i \in \mathbb{N}$ such that $\mathrm{rt}(x, \tau) = x\tau^i$ and also $\mathrm{vars}(x\tau^i) \cap \mathrm{dom}(\tau) = \varnothing$. Let $t \in \mathrm{GTerms} \cap \mathrm{HTerms}$ and

$$\upsilon \stackrel{\mathrm{def}}{=} \left\{\, y \mapsto t \;\middle|\; y \in \mathrm{vars}(x\tau^i) \,\right\}.$$

Then, as τ is satisfiable in T, by Lemma 39, $\tau' \stackrel{\mathrm{def}}{=} \tau \cup \upsilon \in \mathrm{RSubst}$ is also satisfiable in T. Moreover, we have that $x\tau^i\tau' \in \mathrm{GTerms} \cap \mathrm{HTerms}$. Define now $n \stackrel{\mathrm{def}}{=} \mathrm{size}(x\tau^i\tau')$. As $\mathrm{rt}(x, \sigma) \notin \mathrm{HTerms}$, there exists $j \in \mathbb{N}$ such that $\mathrm{size}(x\sigma^j) > n$. Therefore, by Lemma 43,

$$T \vdash \forall(x\tau^i\tau' \neq x\sigma^j). \tag{B.14}$$

By Lemma 40,

$$T \vdash \forall\Bigl(\tau \to (x = x\tau^i)\Bigr). \tag{B.15}$$

Also, by Lemma 40, $T \vdash \forall\Bigl(\sigma \to (x = x\sigma^j)\Bigr)$ so that, as T is a first-order theory,

$$T \vdash \forall\Bigl(\exists W \,.\, \sigma \to \exists W \,.\, (x = x\sigma^j)\Bigr). \tag{B.16}$$

By definition of τ', $\forall(\tau' \to \tau)$. Hence, by hypothesis and the logically true statement $\forall(\tau \to \exists W \,.\, \tau)$, we obtain $T \vdash \forall(\tau' \to \exists W \,.\, \sigma)$. Observe that $\mathrm{vars}(x = x\tau^i\tau') = \{x\}$ and, as a consequence, $\mathrm{vars}(x = x\tau^i\tau') \cap W = \varnothing$. Therefore, by (B.15) and (B.16), we obtain

$$\begin{aligned}
&T \vdash \forall\Bigl(\tau' \to (x = x\tau^i\tau' \land \exists W \,.\, x = x\sigma^j)\Bigr) \\
&\quad\iff T \vdash \forall\Bigl(\tau' \to \exists W \,.\, (x = x\tau^i\tau' \land x = x\sigma^j)\Bigr) \\
&\quad\iff T \vdash \forall\Bigl(\tau' \to \exists W \,.\, (x\tau^i\tau' = x\sigma^j)\Bigr)
\end{aligned}$$

which contradicts (B.14). $\square$

In [54], (weak) variable-idempotent substitutions were introduced as a subclass of substitutions in rational solved form in order to allow a more convenient reasoning about the sharing of variables for possibly non-idempotent substitutions. In [74] a stronger definition was used, taking into consideration also the variables in the domain of the substitution. *Strong* variable-idempotence is a useful concept when dealing with the finiteness of a rational term and the multiplicity of variables occurring in it (e.g., when linearity is a property of interest). In the following we consider this stronger definition, also adopted in [32,33].

Definition 46 (Variable-idempotence.) *A substitution $\sigma \in \mathrm{RSubst}$ is said to be (strongly)* variable-idempotent *if and only if for all $t \in \mathrm{HTerms}$ we have*

$$\mathrm{vars}(t\sigma\sigma) = \mathrm{vars}(t\sigma).$$

The set of variable-idempotent substitutions is denoted VSubst*.*

Note that we have ISubst $\subset$ VSubst $\subset$ RSubst.

Definition 47 ($\mathcal{S}$-transformation.) *The relation $\overset{\mathcal{S}}{\longmapsto} \subseteq \mathrm{RSubst} \times \mathrm{RSubst}$, called $\mathcal{S}$*-step*, is defined by*

$$\frac{(x \mapsto t) \in \sigma \qquad (y \mapsto s) \in \sigma \qquad x \neq y}{\sigma \overset{\mathcal{S}}{\longmapsto} \big(\sigma \setminus \{y \mapsto s\}\big) \cup \big\{y \mapsto s\{x \mapsto t\}\big\}}.$$

If we have a finite sequence of $\mathcal{S}$-steps $\sigma_1 \overset{\mathcal{S}}{\longmapsto} \cdots \overset{\mathcal{S}}{\longmapsto} \sigma_n$ mapping σ_1 to σ_n, then we write $\sigma_1 \overset{\mathcal{S}}{\longmapsto}{}^{*} \sigma_n$ and say that σ_1 can be rewritten, by $\mathcal{S}$-transformation, to σ_n.

The following theorems show that considering substitutions in VSubst is not a restrictive hypothesis.

Theorem 48 *Suppose $\sigma \in \mathrm{RSubst}$ and $\sigma \overset{\mathcal{S}}{\longmapsto}{}^{*} \sigma'$. Then we have $\sigma' \in \mathrm{RSubst}$,* $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$*, and* $\mathrm{vars}(\sigma) = \mathrm{vars}(\sigma')$*. Moreover, if T is any equality theory, we have $T \vdash \forall(\sigma \leftrightarrow \sigma')$.*

PROOF. Proved in [54, Theorem 1]. □

Theorem 49 *Suppose $\sigma \in \mathrm{RSubst}$. Then there exists $\sigma' \in \mathrm{VSubst}$ such that $\sigma \overset{\mathcal{S}}{\longmapsto}{}^{*} \sigma'$ and, for all $\tau \subseteq \sigma'$, $\tau \in \mathrm{VSubst}$.*

PROOF. The proof is the same given for [54, Theorem 2], where a weaker result, using *weak* variable-idempotence, was stated. $\square$

Theorem 50 *Let T be an equality theory and $\sigma \in \mathrm{RSubst}$. Then there exists $\sigma' \in \mathrm{VSubst}$ such that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$, $\mathrm{vars}(\sigma) = \mathrm{vars}(\sigma')$, $T \vdash \forall(\sigma \leftrightarrow \sigma')$ and for all $\tau \subseteq \sigma'$, $\tau \in \mathrm{VSubst}$.*

PROOF. The result easily follows from Theorems 48 and 49. $\square$

The next result concerning a useful property of variable idempotent substitutions will be needed in Subsection B.6 for proving Theorem 19.

Lemma 51 *Let $\sigma \in \mathrm{VSubst}$ be satisfiable in a syntactic equality theory T. Let $s \in \mathrm{HTerms} \cap \mathrm{GTerms}$ and $t \in \mathrm{HTerms}$ and suppose that $T \vdash \forall(\sigma \rightarrow s = t)$. Then $s = t\sigma$.*

PROOF. By hypothesis, $T \vdash \forall(\sigma \rightarrow s = t)$ and $s, t \in \mathrm{HTerms}$ so that we can apply Lemma 42 to obtain

$$\mathrm{rt}(s, \sigma) = \mathrm{rt}(t, \sigma). \tag{B.17}$$

By Proposition 44, there exists $i, j \in \mathbb{N}$ such that $\mathrm{rt}(s, \sigma) = s\sigma^i$ and $\mathrm{rt}(t, \sigma) = t\sigma^j$ and $\mathrm{dom}(\sigma) \cap \mathrm{vars}(t\sigma^j) = \varnothing$. Thus, if $j = 0$, we have $t\sigma^j = t = t\sigma$. On the other hand, if $j > 0$, as $\sigma \in \mathrm{VSubst}$, $\mathrm{vars}(t\sigma^j) = \mathrm{vars}(t\sigma)$ so that $\mathrm{dom}(\sigma) \cap \mathrm{vars}(t\sigma) = \varnothing$ and hence $t\sigma = t\sigma^j$. As $s \in \mathrm{GTerms}$, $\mathrm{vars}(s) = \varnothing$ so that $s = s\sigma^i$. Thus, by (B.17) we have $s = t\sigma$. $\square$

B.4 Some Results on the Groundness and Finiteness Operators

The following proposition is proved in [54], and shows that the function 'gvars' precisely captures the intended property.

Proposition 52 *Let $\sigma \in \mathrm{RSubst}$ and $x \in \mathrm{Vars}$. Then*

$$y \in \mathrm{gvars}(\sigma) \iff \mathrm{rt}(y, \sigma) \in \mathrm{GTerms}.$$

When computing $\mathrm{hvars}(\sigma)$ by means of the fixpoint computation given in Definition 11 on page 18, the fixpoint is reached after a single iteration if $\sigma \in \mathrm{VSubst}$.

Lemma 53 *For each $\sigma \in \mathrm{VSubst}$ we have $\mathrm{hvars}(\sigma) = \mathrm{hvars}_1(\sigma)$.*

PROOF. We show that $\mathrm{hvars}_2(\sigma) \subseteq \mathrm{hvars}_1(\sigma)$. Let $y \in \mathrm{hvars}_2(\sigma)$. By Definition 11, we have two cases:

(1) if $y \in \mathrm{hvars}_1(\sigma)$ then there is nothing to prove;
(2) assume now $y \in \mathrm{dom}(\sigma)$ and $\mathrm{vars}(y\sigma) \subseteq \mathrm{hvars}_1(\sigma)$. By Definition 11, we have two subcases:
  (a) $\mathrm{vars}(y\sigma) \subseteq \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$.
  Then $\mathrm{vars}(y\sigma) \subseteq \mathrm{hvars}_0(\sigma)$, so that $y \in \mathrm{hvars}_1(\sigma)$;
  (b) $V = \mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) \neq \varnothing$ and, for all $z \in V$, $\mathrm{vars}(z\sigma) \cap \mathrm{dom}(\sigma) = \varnothing$.
  Let $z \in V$ so that $z \in \mathrm{vars}(y\sigma)$. By hypothesis, we have $\sigma \in \mathrm{VSubst}$ so that $z \in \mathrm{vars}(y\sigma\sigma)$. As $z \in \mathrm{dom}(\sigma)$ and $\mathrm{vars}(z\sigma) \cap \mathrm{dom}(\sigma) = \varnothing$, $z \notin \mathrm{vars}(z\sigma)$. This means that $z \notin \mathrm{vars}(y\sigma\sigma)$, which is a contradiction since $\sigma \in \mathrm{VSubst}$.

□

Proposition 54 *For each* $\sigma \in \mathrm{VSubst}$*, we have*

$$\mathrm{hvars}(\sigma) = \left\{ y \in \mathrm{Vars} \;\middle|\; \mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) = \varnothing \right\}.$$

PROOF. The result is obtained by applying Lemma 53 and then unfolding Definition 11. □

Proposition 55 *Let* $\sigma \in \mathrm{VSubst}$ *and* $r \in \mathrm{HTerms}$*, where* $\mathrm{vars}(r) \subseteq \mathrm{hvars}(\sigma)$*. Then*

$$\begin{aligned} \mathrm{rt}(r, \sigma) &= r\sigma, \\ \mathrm{vars}(r\sigma) \cap \mathrm{dom}(\sigma) &= \varnothing. \end{aligned}$$

PROOF. Suppose $y \in \mathrm{vars}(r)$. Then, by Proposition 54, $\mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) = \varnothing$. Thus, for any $i > 0$, we have $y\sigma^i = y\sigma \in \mathrm{HTerms}$. Thus $\mathrm{rt}(y, \sigma) = y\sigma$. As this holds for all $y \in \mathrm{vars}(r)$, it follows that $\mathrm{rt}(r, \sigma) = r\sigma$ and $\mathrm{vars}(r\sigma) \cap \mathrm{dom}(\sigma) = \varnothing$. □

The following result shows that, for a variable-idempotent substitution, the finiteness operator precisely captures the intended property.

Lemma 56 *Let* $\sigma \in \mathrm{VSubst}$ *and* $y \in \mathrm{Vars}$*. Then*

$$\mathrm{rt}(y, \sigma) \in \mathrm{HTerms} \iff y \in \mathrm{hvars}(\sigma).$$

PROOF. Since $\sigma \in \mathrm{VSubst}$, by Proposition 54 we have $y \in \mathrm{hvars}(\sigma)$ if and only if $\mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) = \varnothing$.

Let $\mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) = \varnothing$. Then, for any $i > 0$, we have $y\sigma^i = y\sigma \in \mathrm{HTerms}$. Hence $\mathrm{rt}(y, \sigma) = y\sigma \in \mathrm{HTerms}$.

In order to prove the other inclusion, let now $\mathrm{rt}(y, \sigma) \in \mathrm{HTerms}$. By Proposition 44, there exists an $i \in \mathbb{N}$ such that $\mathrm{rt}(y, \sigma) = y\sigma^i$ and $\mathrm{vars}(y\sigma^i) \cap \mathrm{dom}(\sigma) = \varnothing$. Since $\sigma \in VSubst$, we have $\mathrm{vars}(y\sigma^i) = \mathrm{vars}(y\sigma)$, so that $\mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) = \varnothing$. $\square$

In order to prove Proposition 13, i.e., to show that the finiteness operator precisely captures the intended property even for arbitrary substitutions in RSubst, we now prove that this operator is invariant under the application of S-steps.

Lemma 57 *Let* $\sigma, \sigma' \in \mathrm{RSubst}$ *where* $\sigma \overset{S}{\longmapsto} \sigma'$. *Then* $\mathrm{hvars}(\sigma) = \mathrm{hvars}(\sigma')$.

PROOF. Let $(x \mapsto t), (y \mapsto s) \in \sigma$, where $x \neq y$, such that

$$\sigma' = \Bigl(\sigma \setminus \{y \mapsto s\}\Bigr) \cup \Bigl\{y \mapsto s\{x \mapsto t\}\Bigr\}.$$

If $x \notin \mathrm{vars}(s)$ then we have $\sigma = \sigma'$ and the result trivially holds. Thus, we assume $x \in \mathrm{vars}(s)$. We prove the two inclusions separately.

In order to prove $\mathrm{hvars}(\sigma) \subseteq \mathrm{hvars}(\sigma')$ we show, by induction on $m \geq 0$, that we have

$$\mathrm{hvars}_m(\sigma) \subseteq \mathrm{hvars}_m(\sigma').$$

For the base case, when $m = 0$, by Theorem 48 we have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$ so that

$$\begin{aligned} \mathrm{hvars}_0(\sigma) &= \mathrm{Vars} \setminus \mathrm{dom}(\sigma) \\ &= \mathrm{Vars} \setminus \mathrm{dom}(\sigma') \\ &= \mathrm{hvars}_0(\sigma'). \end{aligned}$$

For the inductive step, when $m > 0$, assume $\mathrm{hvars}_{m-1}(\sigma) \subseteq \mathrm{hvars}_{m-1}(\sigma')$ and let $z \in \mathrm{hvars}_m(\sigma)$. By Definition 11, we have two cases: if $z \in \mathrm{hvars}_{m-1}(\sigma)$ then the result follows by a straight application of the inductive hypothesis; otherwise, we have

$$z \in \mathrm{dom}(\sigma) \wedge \mathrm{vars}(z\sigma) \subseteq \mathrm{hvars}_{m-1}(\sigma).$$

Now, if $z \neq y$ we have $z\sigma = z\sigma'$, so that, by Theorem 48 and the inductive hypothesis we have

$$z \in \mathrm{dom}(\sigma') \wedge \mathrm{vars}(z\sigma') \subseteq \mathrm{hvars}_{m-1}(\sigma'),$$

so that, by Definition 11, $z \in \mathrm{hvars}_m(\sigma')$. Otherwise, if $z = y$, then

$$\begin{aligned}\mathrm{vars}(z\sigma) &= \mathrm{vars}(s)\\ &\subseteq \mathrm{hvars}_{m-1}(\sigma).\end{aligned}$$

Since, by hypothesis, $x \in \mathrm{vars}(s)$,

$$\begin{aligned}\mathrm{vars}(z\sigma') &= \mathrm{vars}\bigl(s\{x \mapsto t\}\bigr)\\ &= \bigl(\mathrm{vars}(s) \setminus \{x\}\bigr) \cup \mathrm{vars}(t),\end{aligned}$$

and we need to show $\mathrm{vars}(z\sigma') \subseteq \mathrm{hvars}_{m-1}(\sigma')$. By the inductive hypothesis we have

$$\mathrm{vars}(s) \subseteq \mathrm{hvars}_{m-1}(\sigma');$$

Note that, since $x \in \mathrm{vars}(s)$, it follows $x \in \mathrm{hvars}_{m-1}(\sigma')$ so that, by Definition 11,

$$\begin{aligned}\mathrm{vars}(t) &\subseteq \mathrm{hvars}_{m-2}(\sigma')\\ &\subseteq \mathrm{hvars}_{m-1}(\sigma').\end{aligned}$$

In order to prove $\mathrm{hvars}(\sigma) \supseteq \mathrm{hvars}(\sigma')$ we show, by induction on $m \geq 0$, that we have

$$\mathrm{hvars}_{m+1}(\sigma) \supseteq \mathrm{hvars}_m(\sigma').$$

For the base case, when $m = 0$, by Definition 11 and Theorem 48 we have

$$\begin{aligned}\mathrm{hvars}_1(\sigma) &\supseteq \mathrm{hvars}_0(\sigma)\\ &= \mathrm{Vars} \setminus \mathrm{dom}(\sigma)\\ &= \mathrm{Vars} \setminus \mathrm{dom}(\sigma')\\ &= \mathrm{hvars}_0(\sigma').\end{aligned}$$

For the inductive step, when $m > 0$, assume $\mathrm{hvars}_m(\sigma) \supseteq \mathrm{hvars}_{m-1}(\sigma')$ and let $z \in \mathrm{hvars}_m(\sigma')$. By Definition 11, we have two cases: if $z \in \mathrm{hvars}_{m-1}(\sigma')$ then the result follows by the inductive hypothesis and by Definition 11; otherwise, we have

$$z \in \mathrm{dom}(\sigma') \wedge \mathrm{vars}(z\sigma') \subseteq \mathrm{hvars}_{m-1}(\sigma').$$

Now, if $z \neq y$ we have $z\sigma = z\sigma'$, so that, by Theorem 48 and the inductive hypothesis we have

$$z \in \mathrm{dom}(\sigma) \wedge \mathrm{vars}(z\sigma) \subseteq \mathrm{hvars}_m(\sigma),$$

so that, by Definition 11, $z \in \mathrm{hvars}_{m+1}(\sigma)$. Otherwise, if $z = y$, by definition of σ', the inductive hypothesis and Definition 11, we have

$$\begin{aligned}
\mathrm{vars}(z\sigma') &= \mathrm{vars}\Big(s\{x \mapsto t\}\Big) \\
&= \Big(\mathrm{vars}(s) \setminus \{x\}\Big) \cup \mathrm{vars}(t) \\
&\subseteq \mathrm{hvars}_{m-1}(\sigma') \\
&\subseteq \mathrm{hvars}_m(\sigma) \\
&\subseteq \mathrm{hvars}_{m+1}(\sigma).
\end{aligned}$$

Also note that we have

$$\begin{aligned}
\mathrm{vars}(x\sigma) &= \mathrm{vars}(t) \\
&\subseteq \mathrm{hvars}_m(\sigma)
\end{aligned}$$

so that, by Definition 11 we have

$$x \in \mathrm{hvars}_{m+1}(\sigma).$$

The result follows by observing that

$$\mathrm{vars}(z\sigma) = \mathrm{vars}(s) = \Big(\mathrm{vars}(s) \setminus \{x\}\Big) \cup \{x\}.$$

□

Lemma 58 *Let* $\sigma, \sigma' \in \mathrm{RSubst}$, *where* $\sigma \overset{\mathcal{S}}{\longmapsto}{}^{*} \sigma'$. *Then* $\mathrm{hvars}(\sigma) = \mathrm{hvars}(\sigma')$.

PROOF. By induction on the length $n \geq 0$ of the derivation. For the base case, when $n = 0$, there is nothing to prove. Suppose now that

$$\sigma = \sigma_0 \overset{\mathcal{S}}{\longmapsto} \cdots \overset{\mathcal{S}}{\longmapsto} \sigma_{n-1} \overset{\mathcal{S}}{\longmapsto} \sigma_n = \sigma',$$

where $n > 0$. By the inductive hypothesis, since the derivation $\sigma \overset{\mathcal{S}}{\longmapsto}{}^{*} \sigma_{n-1}$ has length $n - 1$, we have $\mathrm{hvars}(\sigma) = \mathrm{hvars}(\sigma_{n-1})$. Then the thesis follows by Lemma 57. □

Proof of Proposition 13 on page 18. By Theorem 50, there exists $\sigma' \in \mathrm{VSubst}$ such that $\sigma \overset{\mathcal{S}}{\longmapsto}{}^{*} \sigma'$ and, for all equality theories T, $T \vdash \forall(\sigma \leftrightarrow \sigma')$. By

Lemma 56, for all $x \in \mathrm{Vars}$, $\mathrm{rt}(x, \sigma') \in \mathrm{HTerms}$ if and only if $x \in \mathrm{hvars}(\sigma')$. By Lemma 58, we have $\mathrm{hvars}(\sigma) = \mathrm{hvars}(\sigma')$ and, by Proposition 45, for all $x \in \mathrm{Vars}$, $\mathrm{rt}(x, \sigma') \in \mathrm{HTerms}$ if and only if $\mathrm{rt}(x, \sigma) \in \mathrm{HTerms}$. Therefore, for any $x \in \mathrm{Vars}$, $\mathrm{rt}(x, \sigma) \in \mathrm{HTerms}$ if and only if $x \in \mathrm{hvars}(\sigma)$ $\square$

Proof of Proposition 15 on page 18. We prove the two statements (15a) and (15b) separately.

(15a). By hypothesis, $\tau \in {\downarrow}\sigma$, Thus, by Proposition 2, $\mathcal{RT} \vdash \forall(\tau \to \sigma)$. Suppose $x \in \mathrm{hvars}(\tau)$. Then, by Proposition 13, we have $\mathrm{rt}(x, \tau) \in \mathrm{HTerms}$. Therefore we can apply Proposition 45 to obtain $\mathrm{rt}(x, \sigma) \in \mathrm{HTerms}$ and hence, by Proposition 13, $x \in \mathrm{hvars}(\sigma)$.

(15b). Suppose $x \in \mathrm{hvars}(\sigma) \cap \mathrm{gvars}(\sigma)$. Then, by Propositions 13 and 52, $\mathrm{rt}(x, \sigma) \in \mathrm{GTerms} \cap \mathrm{HTerms}$. Thus, by case (44b) of Proposition 44, there exists $i \in \mathbb{N}$ such that $\mathrm{rt}(x, \sigma) = x\sigma^i$ and also $\mathrm{vars}(x\sigma^i) = \varnothing$. Thus $\mathrm{rt}(x\sigma^i, \tau) = x\sigma^i$. Since by hypothesis we have $\tau \in {\downarrow}\sigma$, by Lemma 40 and transitivity we obtain that $\mathcal{RT} \vdash \forall\bigl(\tau \to (x = x\sigma^i)\bigr)$. Thus, by Lemma 42, $\mathrm{rt}(x, \tau) = \mathrm{rt}(x\sigma^i, \tau) = x\sigma^i$. Therefore, by Propositions 13 and 52, $x \in \mathrm{gvars}(\tau) \cap \mathrm{hvars}(\tau)$. $\square$

Proposition 59 *Let* $\sigma, \tau \in \mathrm{RSubst}$ *be satisfiable in a syntactic equality theory* T *and* $W \subseteq \mathrm{Vars}$*, where* $T \vdash \forall(\exists W \,.\, \sigma \leftrightarrow \exists W \,.\, \tau)$*. Then*

$$\mathrm{hvars}(\sigma) \setminus W = \mathrm{hvars}(\tau) \setminus W.$$

PROOF. Suppose $z \in \mathrm{hvars}(\sigma) \setminus W$. By Proposition 13, $\mathrm{rt}(z, \sigma) \in \mathrm{HTerms}$ and hence, by Proposition 45, $\mathrm{rt}(z, \tau) \in \mathrm{HTerms}$. Therefore, by Proposition 13, $z \in \mathrm{hvars}(\tau)$.

The reverse inclusion follows by symmetry. $\square$

Corollary 60 *Let* $e \subseteq \mathrm{Eqs}$ *be satisfiable in the syntactic equality theory* T*. If* $\sigma, \tau \in \mathrm{mgs}(e)$*, then* $\mathrm{hvars}(\sigma) = \mathrm{hvars}(\tau)$*.*

PROOF. By definition of mgs, we have $\sigma, \tau \in \mathrm{RSubst}$, $T \vdash \forall(\sigma \leftrightarrow e)$ and $T \vdash \forall(\tau \leftrightarrow e)$ so that $T \vdash \forall(\sigma \leftrightarrow \tau)$. Thus the result follows by Proposition 59. $\square$

B.5 Abstracting Finiteness

Proof of Theorem 17 on page 19. By Definition 16, $\alpha_H(\sigma) = \mathrm{hvars}(\sigma) \cap \mathrm{VI}$ and $\alpha_H(\tau) = \mathrm{hvars}(\tau) \cap \mathrm{VI}$. The result is a simple consequence of Proposition 59, since $\mathcal{RT}$ is a syntactic equality theory, $\sigma, \tau \in \mathrm{RSubst}$ are satisfiable in $\mathcal{RT}$ and, by hypothesis, $\mathcal{RT} \vdash \forall(\sigma \leftrightarrow \tau)$. $\square$

B.6 Correctness of Abstract Unification on $H \times P$

For the rest of the appendix it is assumed that the equality theory $\mathcal{RT}$ holds. Note that this means that the congruence and identity axioms hold and also that every substitution in RSubst is satisfiable in $\mathcal{RT}$.

Lemma 61 *Let $\bar{s} = (s_1, \ldots, s_n) \in \mathrm{HTerms}^n$ be linear, and suppose the tuple of terms $\bar{t} = (t_1, \ldots, t_n) \in \mathrm{HTerms}^n$ is such that $\mathrm{vars}(\bar{s}) \cap \mathrm{nlvars}(\bar{t}) = \varnothing$ and $\mathrm{mgs}(\bar{s} = \bar{t}) \neq \varnothing$. Then there exists $\mu \in \mathrm{mgs}(\bar{s} = \bar{t})$ such that, for each variable $z \in \mathrm{dom}(\mu) \setminus \bigl(\mathrm{vars}(\bar{s}) \cap \mathrm{vars}(\bar{t})\bigr)$, we have $\mathrm{vars}(z\mu) \cap \mathrm{dom}(\mu) = \varnothing$.*

PROOF. The proof is by induction on the number of variables in $\mathrm{vars}(\bar{s}) \cup \mathrm{vars}(\bar{t})$.

Suppose first that, for some $i = 1, \ldots, n$, we have $s_i = f(r_1, \ldots, r_m)$ and $t_i = f(u_1, \ldots, u_m)$ (with $m \geq 0$). Let

$$\bar{s}' \stackrel{\mathrm{def}}{=} (s_1, \ldots, s_{i-1}, r_1, \ldots, r_m, s_{i+1}, \ldots, s_n),$$
$$\bar{t}' \stackrel{\mathrm{def}}{=} (t_1, \ldots, t_{i-1}, u_1, \ldots, u_m, t_{i+1}, \ldots, t_n).$$

Then $\mathrm{mvars}(\bar{s}') = \mathrm{mvars}(\bar{s})$ and $\mathrm{mvars}(\bar{t}') = \mathrm{mvars}(\bar{t})$ so that, as $\bar{s}$ is linear, $\bar{s}'$ is linear, $\mathrm{vars}(\bar{s}') \cap \mathrm{nlvars}(\bar{t}') = \varnothing$ and $\mathrm{vars}(\bar{s}') \cap \mathrm{vars}(\bar{t}') = \mathrm{vars}(\bar{s}) \cap \mathrm{vars}(\bar{t})$. Moreover, by the congruence axiom (B.4), $\mathrm{mgs}(\bar{s}' = \bar{t}') = \mathrm{mgs}(\bar{s} = \bar{t})$. We repeat this process until all terms in $\bar{s}'$ and $\bar{t}'$ can not be decomposed any further. (Note that in the case that s_i and t_i are identical constants, we can remove them from $\bar{s}'$ and $\bar{t}'$, since the corresponding equation $s_i = t_i$ holds vacuously.) Thus, as $\bar{s}$ and $\bar{t}$ are finite sequences of finite terms, we can assume that, for all $i = 1, \ldots, n$, either $s_i \in \mathrm{Vars}$ or $t_i \in \mathrm{Vars}$.

Secondly, suppose that for some $i = 1, \ldots, n$, $s_i = t_i$. By the previous paragraph, we can assume that $s_i \in \mathrm{Vars}$. Let

$$\bar{s}_i \stackrel{\mathrm{def}}{=} (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n),$$
$$\bar{t}_i \stackrel{\mathrm{def}}{=} (t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n).$$

Then $\mathrm{mvars}(\bar{s}_i) \cup \{s_i\} = \mathrm{mvars}(\bar{s})$ and $\mathrm{mvars}(\bar{t}_i) \cup \{s_i\} = \mathrm{mvars}(\bar{t})$ so that, as $\bar{s}$ is linear, $\bar{s}_i$ is linear, $\mathrm{vars}(\bar{s}_i) \cap \mathrm{nlvars}(\bar{t}_i) = \varnothing$ and

$$\bigl(\mathrm{vars}(\bar{s}_i) \cap \mathrm{vars}(\bar{t}_i)\bigr) \cup \{s_i\} = \mathrm{vars}(\bar{s}) \cap \mathrm{vars}(\bar{t}).$$

As $\bar{s}$ is linear and $\mathrm{vars}(\bar{s}) \cap \mathrm{nlvars}(\bar{t}) = \varnothing$, $s_i \notin \mathrm{vars}(\bar{s}_i) \cup \mathrm{vars}(\bar{t}_i)$ and hence, for all $\mu \in \mathrm{mgs}(\bar{s} = \bar{t})$, we have $s_i \notin \mathrm{dom}(\mu)$. Therefore

$$\mathrm{dom}(\mu) \setminus \Bigl(\mathrm{vars}(\bar{s}) \cap \mathrm{vars}(\bar{t})\Bigr) = \mathrm{dom}(\mu) \setminus \Bigl(\mathrm{vars}(\bar{s}_i) \cap \mathrm{vars}(\bar{t}_i)\Bigr).$$

Furthermore, by the congruence axiom (B.1), $\mathrm{mgs}(\bar{s}_i = \bar{t}_i) = \mathrm{mgs}(\bar{s} = \bar{t})$. Thus, as $\bar{s}$ and $\bar{t}$ are sequences of finite length n, we can assume that $s_i \neq t_i$, for all $i = 1, \ldots, n$.

Therefore, for the rest of the proof, we will assume that for each $i = 1, \ldots, n$, $s_i \neq t_i$ and either $s_i \in \mathrm{Vars}$ or $t_i \in \mathrm{Vars}$.

For the base case, we have $\mathrm{vars}(\bar{s}) \cup \mathrm{vars}(\bar{t}) = \varnothing$ and the result holds.

For the inductive step, $\mathrm{vars}(\bar{s}) \cup \mathrm{vars}(\bar{t}) \neq \varnothing$ so that $n > 0$. As the order of the equations in $\bar{s} = \bar{t}$ is not relevant to the hypothesis, we assume, without loss of generality that if, for some $i = 1, \ldots, n$, $\mathrm{vars}(s_i) \cap \mathrm{vars}(t_i) = \varnothing$, then $\mathrm{vars}(s_1) \cap \mathrm{vars}(t_1) = \varnothing$. There are three cases we consider separately:

a. for all $i = 1, \ldots, n$, $\mathrm{vars}(s_i) \cap \mathrm{vars}(t_i) \neq \varnothing$;
b. $s_1 \in \mathrm{Vars} \setminus \mathrm{vars}(t_1)$;
c. $t_1 \in \mathrm{Vars} \setminus \mathrm{vars}(s_1)$.

Case a. For all $i = 1, \ldots, n$, $\mathrm{vars}(s_i) \cap \mathrm{vars}(t_i) \neq \varnothing$.

For each $i = 1, \ldots, n$, we are assuming that either $s_i \in \mathrm{Vars}$ or $t_i \in \mathrm{Vars}$, Therefore, for each $i = 1, \ldots, n$, $s_i \in \mathrm{vars}(t_i)$ or $t_i \in \mathrm{vars}(s_i)$ so that, without loss of generality, we can assume, for some k, where $0 \leq k \leq n$, $s_i \in \mathrm{Vars}$ if $1 \leq i \leq k$ and $t_i \in \mathrm{Vars}$ if $k + 1 \leq i \leq n$.

Let

$$\mu \stackrel{\mathrm{def}}{=} \{s_1 = t_1, \ldots, s_k = t_k\} \cup \{t_{k+1} = s_{k+1}, \ldots, t_n = s_n\}.$$

We now show that $\mu \subseteq \mathrm{Eqs}$ is in rational solved form. As $\bar{s}$ is linear, $(s_1, \ldots, s_k)$ is linear. As $\bar{s}$ is linear and $t_i \in \mathrm{vars}(s_i)$ if $k + 1 \leq i \leq n$, then $(t_{k+1}, \ldots, t_n)$ is linear and $\{s_1, \ldots, s_k\} \cap \{t_{k+1}, \ldots, t_n\} = \varnothing$. As we are assuming that, for all $i = 1, \ldots, n$, $s_i \neq t_i$ and $\mathrm{vars}(s_i) \cap \mathrm{vars}(t_i) \neq \varnothing$, it follows that $t_i \notin \mathrm{Vars}$ when $1 \leq i \leq k$ and $s_i \notin \mathrm{Vars}$ when $k + 1 \leq i \leq n$, so that each equation in μ is a binding and μ has no circular subsets. Thus $\mu \in \mathrm{RSubst}$ and hence, by the congruence axiom (B.2), $\mu \in \mathrm{mgs}(\bar{s} = \bar{t})$.

As $s_i \in \mathrm{vars}(t_i)$ when $1 \leq i \leq k$ and $t_i \in \mathrm{vars}(s_i)$ when $k + 1 \leq i \leq n$, $\mathrm{dom}(\mu) \setminus \Bigl(\mathrm{vars}(\bar{s}) \cap \mathrm{vars}(\bar{t})\Bigr) = \varnothing$. Therefore the required result holds.

Case b. $s_1 \in \mathrm{Vars} \setminus \mathrm{vars}(t_1)$.

Let

$$\begin{aligned} \bar{s}_1 &\stackrel{\text{def}}{=} (s_2, \dots, s_n), \\ \bar{t}_1 &\stackrel{\text{def}}{=} \Big(t_2\{s_1 \mapsto t_1\}, \dots, t_n\{s_1 \mapsto t_1\}\Big). \end{aligned} \tag{B.18}$$

As $\bar{s}$ is linear, $s_1 \notin \text{vars}(\bar{s}_1)$. Also, all occurrences of s_1 in $\bar{t}$ are replaced in $\bar{t}_1$ by t_1 so that, as $s_1 \notin \text{vars}(t_1)$, $s_1 \notin \text{vars}(\bar{t}_1)$. Thus

$$s_1 \notin \text{vars}(\bar{s}_1) \cup \text{vars}(\bar{t}_1). \tag{B.19}$$

Therefore $\text{vars}(\bar{s}_1) \cup \text{vars}(\bar{t}_1) \subset \text{vars}(\bar{s}) \cup \text{vars}(\bar{t})$. Now since $\bar{s}$ is linear, $\bar{s}_1$ is linear. Thus, to apply the inductive hypothesis to $\bar{s}_1$ and $\bar{t}_1$, we have to show that

$$\text{vars}(\bar{s}_1) \cap \text{nlvars}(\bar{t}_1) = \varnothing. \tag{B.20}$$

Suppose that $u \in \text{vars}(\bar{s}_1)$ so that $u \in \text{vars}(\bar{s})$. Now, by hypothesis, we have $\text{vars}(\bar{s}) \cap \text{nlvars}(\bar{t}) = \varnothing$. Thus $s_1, u \notin \text{nlvars}(\bar{t})$. If $u \in \text{vars}\Big((t_2, \dots, t_n)\Big)$ so that $u \notin \text{vars}(t_1)$, then $u \notin \text{nlvars}(\bar{t}_1)$. On the other hand, if $u \notin \text{vars}\Big((t_2, \dots, t_n)\Big)$, then, as $s_1 \notin \text{nlvars}\Big((t_2, \dots, t_n)\Big)$ and $u \notin \text{nlvars}(t_1)$, $u \notin \text{nlvars}(\bar{t}_1)$. Thus, for all $u \in \text{vars}(\bar{s}_1)$, $u \notin \text{nlvars}(\bar{t}_1)$. Hence (B.20) holds. It follows that the inductive hypothesis for $\bar{s}_1$ and $\bar{t}_1$ holds. Therefore there exists $\mu_1 \in \text{RSubst}$ where

$$\mu_1 \in \text{mgs}(\bar{s}_1 = \bar{t}_1)$$

such that, for each $z \in \text{dom}(\mu_1) \setminus \Big(\text{vars}(\bar{s}_1) \cap \text{vars}(\bar{t}_1)\Big)$, $\text{vars}(z\mu_1) \cap \text{dom}(\mu_1) = \varnothing$.

Let

$$\mu \stackrel{\text{def}}{=} \{s_1 = t_1\mu_1\} \cup \mu_1. \tag{B.21}$$

We now show that $\mu \subseteq \text{Eqs}$ is in $\text{mgs}(\bar{s} = \bar{t})$. First we show that μ is in rational solved form. By (B.19),

$$s_1 \notin \text{vars}(\mu_1), \tag{B.22}$$

and, as $s_1 \notin \text{vars}(t_1)$, we have

$$s_1 \notin \text{vars}(t_1\mu_1). \tag{B.23}$$

Thus, as $\mu_1 \in \text{RSubst}$, μ has no identities or circular subsets so that $\mu \in \text{RSubst}$. By Lemma 41, $\mu \in \text{mgs}(\bar{s} = \bar{t})$.

Let

$$z \in \text{dom}(\mu) \setminus \Big(\text{vars}(\bar{s}) \cap \text{vars}(\bar{t})\Big). \tag{B.24}$$

Then we have to show that

$$\text{vars}(z\mu) \cap \text{dom}(\mu) = \varnothing. \tag{B.25}$$

It follows from (B.21) and (B.24) that either $z \in \mathrm{dom}(\mu_1)$ so that $z\mu = z\mu_1$ or $z = s_1$ and $z\mu = t_1\mu_1$. We consider these two cases separately.

Suppose first that $z \in \mathrm{dom}(\mu_1)$. By (B.18), we have both $\mathrm{vars}(\bar{s}_1) \subseteq \mathrm{vars}(\bar{s})$ and $\mathrm{vars}(\bar{t}_1) \subseteq \mathrm{vars}(\bar{t})$, so that $\mathrm{vars}(\bar{s}_1) \cap \mathrm{vars}(\bar{t}_1) \subseteq \mathrm{vars}(\bar{s}) \cap \mathrm{vars}(\bar{t})$. Hence we have $z \in \mathrm{dom}(\mu_1) \setminus \bigl(\mathrm{vars}(\bar{s}_1) \cap \mathrm{vars}(\bar{t}_1)\bigr)$. Thus we obtain, by the inductive hypothesis, $\mathrm{vars}(z\mu_1) \cap \mathrm{dom}(\mu_1) = \varnothing$. Now, as $z \in \mathrm{dom}(\mu_1)$ and (B.22) holds, $s_1 \notin \mathrm{vars}(z\mu_1)$. Thus, as $\mathrm{dom}(\mu) = \mathrm{dom}(\mu_1) \cup \{s_1\}$, $\mathrm{vars}(z\mu_1) \cap \mathrm{dom}(\mu) = \varnothing$. Hence, as $z\mu = z\mu_1$, (B.25) holds.

Secondly suppose that $z = s_1$. Then we have that $s_1 \notin \mathrm{vars}(\bar{s}) \cap \mathrm{vars}(\bar{t})$. Hence $\bar{t}_1 = (t_2, \dots, t_n)$. Let u be any variable in $\mathrm{vars}(t_1)$. Then we have that $u \notin \mathrm{vars}(\bar{s}_1) \cap \mathrm{vars}(\bar{t}_1)$, since $\mathrm{vars}(\bar{s}) \cap \mathrm{nlvars}(\bar{t}) = \varnothing$. If $u \in \mathrm{dom}(\mu_1)$, then we can apply the inductive hypothesis to obtain $\mathrm{vars}(u\mu_1) \cap \mathrm{dom}(\mu_1) = \varnothing$. On the other hand, if $u \notin \mathrm{dom}(\mu_1)$, we have $u = u\mu_1$ and $\mathrm{vars}(u\mu_1) \cap \mathrm{dom}(\mu_1) = \varnothing$. Hence $\mathrm{vars}(t_1\mu_1) \cap \mathrm{dom}(\mu_1) = \varnothing$. Thus, as $\mathrm{dom}(\mu) = \mathrm{dom}(\mu_1) \cup \{s_1\}$, by (B.23), $\mathrm{vars}(t_1\mu_1) \cap \mathrm{dom}(\mu) = \varnothing$. Therefore, as $z\mu = t_1\mu_1$, (B.25) holds.

Case c. $t_1 \in \mathrm{Vars} \setminus \mathrm{vars}(s_1)$.

Let

$$\begin{aligned} \bar{s}_1 &\stackrel{\mathrm{def}}{=} \Bigl(s_2\{t_1 \mapsto s_1\}, \dots, s_n\{t_1 \mapsto s_1\}\Bigr), \\ \bar{t}_1 &\stackrel{\mathrm{def}}{=} \Bigl(t_2\{t_1 \mapsto s_1\}, \dots, t_n\{t_1 \mapsto s_1\}\Bigr). \end{aligned} \tag{B.26}$$

All occurrences of t_1 in $\bar{s}$ and $\bar{t}$ are replaced in $\bar{s}_1$ and $\bar{t}_1$ by s_1 so that, since $t_1 \notin \mathrm{vars}(s_1)$,

$$t_1 \notin \mathrm{vars}(\bar{s}_1) \cup \mathrm{vars}(\bar{t}_1). \tag{B.27}$$

Therefore $\mathrm{vars}(\bar{s}_1) \cup \mathrm{vars}(\bar{t}_1) \subset \mathrm{vars}(\bar{s}) \cup \mathrm{vars}(\bar{t})$. Now, $\bar{s}_1$ is linear since $\bar{s}$ is linear. Thus, to apply the inductive hypothesis to $\bar{s}_1$ and $\bar{t}_1$, we have to show that

$$\mathrm{vars}(\bar{s}_1) \cap \mathrm{nlvars}(\bar{t}_1) = \varnothing. \tag{B.28}$$

Suppose u is any variable in $\mathrm{vars}(\bar{s}_1)$. Then either $u \in \mathrm{vars}\bigl((s_2, \dots, s_n)\bigr)$ or we have $u \in \mathrm{vars}(s_1)$ and $t_1 \in \mathrm{vars}\bigl((s_2, \dots, s_n)\bigr)$. By hypothesis, $\mathrm{vars}(\bar{s}) \cap \mathrm{nlvars}(\bar{t}) = \varnothing$, so that $u \notin \mathrm{nlvars}(\bar{t})$. If $u \in \mathrm{vars}\bigl((s_2, \dots, s_n)\bigr)$, then, as $\bar{s}$ is linear, $u \notin \mathrm{vars}(s_1)$. Thus, it follows from (B.26) that $u \notin \mathrm{nlvars}(\bar{t}_1)$. If $t_1 \in \mathrm{vars}\bigl((s_2, \dots, s_n)\bigr)$, then we have $t_1 \notin \mathrm{vars}\bigl((t_2, \dots, t_n)\bigr)$ so that, again by (B.26), $\bar{t}_1 = (t_2, \dots, t_n)$. Thus, for all $u \in \mathrm{vars}(\bar{s}_1)$, $u \notin \mathrm{nlvars}(\bar{t}_1)$. Hence (B.28) holds. It follows that the inductive hypothesis for $\bar{s}_1$ and $\bar{t}_1$ holds. Therefore there exists $\mu_1 \in \mathrm{RSubst}$ where

$$\mu_1 \in \mathrm{mgs}(\bar{s}_1 = \bar{t}_1)$$

such that, for each $z \in \operatorname{dom}(\mu_1) \setminus \bigl(\operatorname{vars}(\bar{s}_1) \cap \operatorname{vars}(\bar{t}_1)\bigr)$, we have $\operatorname{vars}(z\mu_1) \cap \operatorname{dom}(\mu_1) = \varnothing$.

Let

$$\mu \stackrel{\text{def}}{=} \{t_1 = s_1\mu_1\} \cup \mu_1. \tag{B.29}$$

We now show that $\mu \subseteq \text{Eqs}$ is in $\operatorname{mgs}(\bar{s} = \bar{t})$. First we show that μ is in rational solved form. By (B.27),

$$t_1 \notin \operatorname{vars}(\mu_1), \tag{B.30}$$

and, as $t_1 \notin \operatorname{vars}(s_1)$, we have

$$t_1 \notin \operatorname{vars}(s_1\mu_1). \tag{B.31}$$

Thus, as $\mu_1 \in \text{RSubst}$, μ has no identities or circular subsets so that $\mu \in \text{RSubst}$. By Lemma 41, $\mu \in \operatorname{mgs}(\bar{s} = \bar{t})$.

Let

$$z \in \operatorname{dom}(\mu) \setminus \bigl(\operatorname{vars}(\bar{s}) \cap \operatorname{vars}(\bar{t})\bigr). \tag{B.32}$$

Then we have to show that

$$\operatorname{vars}(z\mu) \cap \operatorname{dom}(\mu) = \varnothing. \tag{B.33}$$

It follows from (B.29) and (B.32) that either $z \in \operatorname{dom}(\mu_1)$ so that $z\mu = z\mu_1$ or $z = t_1$ and $z\mu = s_1\mu_1$. We consider these two cases separately.

Suppose first that $z \in \operatorname{dom}(\mu_1)$. To apply the inductive hypothesis to z, we need to show that,

$$\operatorname{vars}(\bar{s}_1) \cap \operatorname{vars}(\bar{t}_1) \subseteq \operatorname{vars}(\bar{s}) \cap \operatorname{vars}(\bar{t}).$$

To see this, let us suppose $u \in \operatorname{vars}(\bar{s}_1) \cap \operatorname{vars}(\bar{t}_1)$. Then, by (B.26), either we have $u \in \operatorname{vars}\bigl((s_2, \ldots, s_n)\bigr)$ or $u \in \operatorname{vars}(s_1)$ and $t_1 \in \operatorname{vars}\bigl((s_2, \ldots, s_n)\bigr)$. If $u \in \operatorname{vars}\bigl((s_2, \ldots, s_n)\bigr)$, then $u \in \operatorname{vars}(\bar{s})$ so that, as $\bar{s}$ is linear, we have also $u \notin \operatorname{vars}(s_1)$ and hence $u \in \operatorname{vars}\bigl((t_2, \ldots, t_n)\bigr)$. Alternatively, if $u \in \operatorname{vars}(s_1)$ and $t_1 \in \operatorname{vars}\bigl((s_2, \ldots, s_n)\bigr)$, then $u, t_1 \in \operatorname{vars}(\bar{s})$. Moreover, by hypothesis, $\operatorname{vars}(\bar{s}) \cap \operatorname{nlvars}(\bar{t}) = \varnothing$, so that $t_1 \notin \operatorname{vars}\bigl((t_2, \ldots, t_n)\bigr)$. Thus $\bar{t}_1 = (t_2, \ldots, t_n)$ and hence $u \in \operatorname{vars}(\bar{t})$. Therefore, in both cases, $u \in \operatorname{vars}(\bar{s}) \cap \operatorname{vars}(\bar{t})$. It follows that $z \in \operatorname{dom}(\mu_1) \setminus \bigl(\operatorname{vars}(\bar{s}_1) \cap \operatorname{vars}(\bar{t}_1)\bigr)$. Thus, by the inductive hypothesis, we have $\operatorname{vars}(z\mu_1) \cap \operatorname{dom}(\mu_1) = \varnothing$. Now, as $z \in \operatorname{dom}(\mu_1)$ and (B.30) holds, $t_1 \notin \operatorname{vars}(z\mu_1)$. Thus, as $\operatorname{dom}(\mu) = \operatorname{dom}(\mu_1) \cup \{t_1\}$, $\operatorname{vars}(z\mu_1) \cap \operatorname{dom}(\mu) = \varnothing$. Hence, as $z\mu = z\mu_1$, (B.33) holds.

Secondly, suppose that $z = t_1$. Then $t_1 \notin \operatorname{vars}(\bar{s}) \cap \operatorname{vars}(\bar{t})$ and, consequently, $\bar{s}_1 = (s_2, \ldots, s_n)$. Let u be any variable in $\operatorname{vars}(s_1)$. Then, as $\bar{s}$ is linear, we have $u \notin \text{vars}(\bar{s}_1)$ so that $u \notin \text{vars}(\bar{s}_1) \cap \text{vars}(\bar{t}_1)$. Thus, if $u \in \text{dom}(\mu_1)$, we can apply the inductive hypothesis to u and obtain $\text{vars}(u\mu_1) \cap \text{dom}(\mu_1) = \varnothing$. On the other hand, if $u \notin \text{dom}(\mu_1)$, $u = u\mu_1$ and $\text{vars}(u\mu_1) \cap \text{dom}(\mu_1) = \varnothing$. Hence $\text{vars}(s_1\mu_1) \cap \text{dom}(\mu_1) = \varnothing$. Thus, as $\text{dom}(\mu) = \text{dom}(\mu_1) \cup \{t_1\}$, by (B.31), $\text{vars}(s_1\mu_1) \cap \text{dom}(\mu) = \varnothing$. Therefore, as $z\mu = s_1\mu_1$, (B.33) holds. □

Lemma 62 *Suppose that the tuple of terms* $\bar{s} \stackrel{\text{def}}{=} (s_1, \dots, s_n) \in \text{HTerms}^n$ *is linear,* $\bar{t} \stackrel{\text{def}}{=} (t_1, \dots, t_n) \in \text{HTerms}^n$ *and* $\text{mgs}(\bar{s} = \bar{t}) \neq \varnothing$. *Then there exists* $\mu \in \text{mgs}(\bar{s} = \bar{t})$ *and, for each* $z \in \text{dom}(\mu) \setminus \text{vars}(\bar{s})$, *the following properties hold:*

(1) $\text{vars}(z\mu) \subseteq \text{vars}(\bar{s})$*;*
(2) $\text{vars}(z\mu) \cap \text{dom}(\mu) = \varnothing$.

PROOF. The proof is by induction on the number of variables in $\text{vars}(\bar{s}) \cup \text{vars}(\bar{t})$.

Suppose first that, for some $i = 1, \dots, n$, we have $s_i = f(r_1, \dots, r_m)$ and $t_i = f(u_1, \dots, u_m)$ $(m \geq 0)$. Let

$$\bar{s}' \stackrel{\text{def}}{=} (s_1, \dots, s_{i-1}, r_1, \dots, r_m, s_{i+1}, \dots, s_n),$$
$$\bar{t}' \stackrel{\text{def}}{=} (t_1, \dots, t_{i-1}, u_1, \dots, u_m, t_{i+1}, \dots, t_n).$$

Then $\text{mvars}(\bar{s}') = \text{mvars}(\bar{s})$ and $\text{mvars}(\bar{t}') = \text{mvars}(\bar{t})$ so that, as $\bar{s}$ is linear, $\bar{s}'$ is linear. Moreover, by the congruence axiom (B.4), $\text{mgs}(\bar{s}' = \bar{t}') = \text{mgs}(\bar{s} = \bar{t})$. We repeat this process until all terms in $\bar{s}'$ and $\bar{t}'$ can not be decomposed any further. (Note that in the case that s_i and t_i are identical constants, we can remove them from $\bar{s}'$ and $\bar{t}'$, since the corresponding equation $s_i = t_i$ holds vacuously.) Thus, as $\bar{s}$ and $\bar{t}$ are finite sequences of finite terms, we can assume that, for all $i = 1, \dots, n$, either $s_i \in \text{Vars}$ or $t_i \in \text{Vars}$.

Secondly, suppose that for some $i = 1, \dots, n$, $s_i = t_i$. By the previous paragraph, we can assume that $s_i \in \text{Vars}$. Let

$$\bar{s}_i \stackrel{\text{def}}{=} (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n),$$
$$\bar{t}_i \stackrel{\text{def}}{=} (t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n).$$

Then $\text{mvars}(\bar{s}_i) \cup \{s_i\} = \text{mvars}(\bar{s})$ and $\text{mvars}(\bar{t}_i) \cup \{s_i\} = \text{mvars}(\bar{t})$ so that, as $\bar{s}$ is linear, $\bar{s}_i$ is linear. Therefore

$$\text{dom}(\mu) \setminus \text{vars}(\bar{s}) \subseteq \text{dom}(\mu) \setminus \text{vars}(\bar{s}_i).$$

Furthermore, by the congruence axiom (B.1), $\text{mgs}(\bar{s}_i = \bar{t}_i) = \text{mgs}(\bar{s} = \bar{t})$. Thus, as $\bar{s}$ and $\bar{t}$ are sequences of finite length n, we can assume that $s_i \neq t_i$, for all $i = 1, \dots, n$.

Therefore, for the rest of the proof, we will assume that $s_i \neq t_i$ and either $s_i \in \mathrm{Vars}$ or $t_i \in \mathrm{Vars}$, for all $i = 1, \ldots, n$.

For the base case, we have $\mathrm{vars}(\bar{s}) \cup \mathrm{vars}(\bar{t}) = \varnothing$ and the result holds.

For the inductive step, $\mathrm{vars}(\bar{s}) \cup \mathrm{vars}(\bar{t}) \neq \varnothing$ so that $n > 0$. As the order of the equations in $\bar{s} = \bar{t}$ is not relevant to the hypothesis, we assume, without loss of generality that if, for some $i = 1, \ldots, n$, $\mathrm{vars}(s_i) \cap \mathrm{vars}(t_i) = \varnothing$ then, we have $\mathrm{vars}(s_1) \cap \mathrm{vars}(t_1) = \varnothing$. There are four cases we consider separately:

a. for all $i = 1, \ldots, n$, $\mathrm{vars}(s_i) \cap \mathrm{vars}(t_i) \neq \varnothing$;
b. $s_1 \in \mathrm{Vars} \setminus \mathrm{vars}(t_1)$;
c. $t_1 \in \mathrm{Vars} \setminus \mathrm{vars}(\bar{s})$ and $s_1 \notin \mathrm{Vars}$;
d. $t_1 \in \mathrm{vars}(\bar{s}) \setminus \mathrm{vars}(s_1)$ and $s_1 \notin \mathrm{Vars}$.

Case a. For all $i = 1, \ldots, n$, $\mathrm{vars}(s_i) \cap \mathrm{vars}(t_i) \neq \varnothing$.

For each $i = 1, \ldots, n$, we are assuming that either $s_i \in \mathrm{Vars}$ or $t_i \in \mathrm{Vars}$, Therefore, for each $i = 1, \ldots, n$, $s_i \in \mathrm{vars}(t_i)$ or $t_i \in \mathrm{vars}(s_i)$ so that, without loss of generality, we can assume, for some k, where $0 \leq k \leq n$, $s_i \in \mathrm{Vars}$ if $1 \leq i \leq k$ and $t_i \in \mathrm{Vars}$ if $k + 1 \leq i \leq n$.

Let
$$\mu \stackrel{\mathrm{def}}{=} \{s_1 = t_1, \ldots, s_k = t_k\} \cup \{t_{k+1} = s_{k+1}, \ldots, t_n = s_n\}.$$
We show that $\mu \subseteq \mathrm{Eqs}$ is in $\mathrm{mgs}(\bar{s} = \bar{t})$. First we must show that $\mu \in \mathrm{RSubst}$. As $\bar{s}$ is linear, $(s_1, \ldots, s_k)$ is linear. As $\bar{s}$ is linear and $t_i \in \mathrm{vars}(s_i)$ if $k + 1 \leq i \leq n$, then $(t_{k+1}, \ldots, t_n)$ is linear and $\{s_1, \ldots, s_k\} \cap \{t_{k+1}, \ldots, t_n\} = \varnothing$. As we are assuming that, for all $i = 1, \ldots, n$, $s_i \neq t_i$ and $\mathrm{vars}(s_i) \cap \mathrm{vars}(t_i) \neq \varnothing$, it follows that $t_i \notin \mathrm{Vars}$ when $1 \leq i \leq k$ and $s_i \notin \mathrm{Vars}$ when $k + 1 \leq i \leq n$, so that each equation in μ is a binding and μ has no circular subsets. Thus $\mu \in \mathrm{RSubst}$ and hence, by the congruence axiom (B.2), $\mu \in \mathrm{mgs}(\bar{s} = \bar{t})$.

As $\{t_{k+1}, \ldots, t_n\} \subseteq \mathrm{vars}\bigl((s_{k+1}, \ldots, s_n)\bigr)$, we have $\mathrm{dom}(\mu) \setminus \mathrm{vars}(\bar{s}) = \varnothing$. Therefore the required result holds.

Case b. $s_1 \in \mathrm{Vars} \setminus \mathrm{vars}(t_1)$.

Let
$$\begin{aligned} \bar{s}_1 &\stackrel{\mathrm{def}}{=} (s_2, \ldots, s_n), \\ \bar{t}_1 &\stackrel{\mathrm{def}}{=} \bigl(t_2\{s_1 \mapsto t_1\}, \ldots, t_n\{s_1 \mapsto t_1\}\bigr). \end{aligned}$$

As $\bar{s}$ is linear, $\bar{s}_1$ is linear and $s_1 \notin \mathrm{vars}(\bar{s}_1)$. Also, all occurrences of s_1 in $\bar{t}$ are replaced in $\bar{t}_1$ by t_1 so that, as $s_1 \notin \mathrm{vars}(t_1)$ (by the assumption for this case), $s_1 \notin \mathrm{vars}(\bar{t}_1)$. Thus
$$s_1 \notin \mathrm{vars}(\bar{s}_1) \cup \mathrm{vars}(\bar{t}_1). \tag{B.34}$$

It follows that $\text{vars}(\bar{s}_1) \cup \text{vars}(\bar{t}_1) \subset \text{vars}(\bar{s}) \cup \text{vars}(\bar{t})$ so that the inductive hypothesis applies to $\bar{s}_1$ and $\bar{t}_1$. Thus there exists $\mu_1 \in \text{RSubst}$ where

$$\mu_1 \in \text{mgs}(\bar{s}_1 = \bar{t}_1)$$

such that, for each $z \in \text{dom}(\mu_1) \setminus \text{vars}(\bar{s}_1)$, properties 1 and 2 hold using μ_1 and $\bar{s}_1$.

Let

$$\mu \stackrel{\text{def}}{=} \{s_1 = t_1\mu_1\} \cup \mu_1.$$

We show that $\mu \subseteq \text{Eqs}$ is in $\text{mgs}(\bar{s} = \bar{t})$. By (B.34), we have $s_1 \notin \text{vars}(\mu_1)$ so that $s_1 \notin \text{dom}(\mu_1)$. Also, since $\mu_1 \in \text{RSubst}$, μ has no identities or circular subsets. Thus we have $\mu \in \text{RSubst}$. By Lemma 41, $\mu \in \text{mgs}(\bar{s} = \bar{t})$.

Suppose that $z \in \text{dom}(\mu) \setminus \text{vars}(\bar{s})$. As

$$\text{vars}(\bar{s}_1) \cup \{s_1\} = \text{vars}(\bar{s})$$

and

$$\text{dom}(\mu_1) \cup \{s_1\} = \text{dom}(\mu),$$

we have

$$\text{dom}(\mu_1) \setminus \text{vars}(\bar{s}_1) = \text{dom}(\mu) \setminus \text{vars}(\bar{s}). \tag{B.35}$$

Therefore $z \in \text{dom}(\mu_1) \setminus \text{vars}(\bar{s}_1)$ and $z\mu_1 = z\mu$. Thus the inductive properties 1 and 2 using μ_1 and $\bar{s}_1$ can be applied to z. We show that properties 1 and 2 using μ and $\bar{s}$ can be applied to z.

(1) By property 1, $\text{vars}(z\mu) \subseteq \text{vars}(\bar{s}_1)$ and hence, $\text{vars}(z\mu) \subseteq \text{vars}(\bar{s})$.
(2) By property 2, we have $\text{vars}(z\mu) \cap \text{dom}(\mu_1) = \varnothing$. Now $s_1 \notin \text{vars}(z\mu)$ because $s_1 \notin \text{vars}(\bar{s}_1)$ (since $\bar{s}$ is linear) and $\text{vars}(z\mu) \subseteq \text{vars}(\bar{s}_1)$ (by property 1). Thus, as $\text{dom}(\mu) = \text{dom}(\mu_1) \cup \{s_1\}$, we have $\text{vars}(z\mu) \cap \text{dom}(\mu) = \varnothing$.

Case c. Assume that $t_1 \in \text{Vars} \setminus \text{vars}(\bar{s})$ and $s_1 \notin \text{Vars}$.

Let

$$\bar{s}_1 \stackrel{\text{def}}{=} (s_2, \ldots, s_n),$$
$$\bar{t}_1 \stackrel{\text{def}}{=} \Big(t_2\{t_1 \mapsto s_1\}, \ldots, t_n\{t_1 \mapsto s_1\}\Big).$$

As $\bar{s}$ is linear, $\bar{s}_1$ is linear. By the assumption for this case, $t_1 \notin \text{vars}(\bar{s}_1)$. Also, all occurrences of t_1 in $\bar{t}$ are replaced in $\bar{t}_1$ by s_1 so that $t_1 \notin \text{vars}(\bar{t}_1)$. Thus

$$t_1 \notin \text{vars}(\bar{s}_1) \cup \text{vars}(\bar{t}_1). \tag{B.36}$$

It follows that $\text{vars}(\bar{s}_1) \cup \text{vars}(\bar{t}_1) \subset \text{vars}(\bar{s}) \cup \text{vars}(\bar{t})$ so that we can apply the inductive hypothesis to $\bar{s}_1$ and $\bar{t}_1$. Thus there exists $\mu_1 \in \text{RSubst}$ where

$$\mu_1 \in \text{mgs}(\bar{s}_1 = \bar{t}_1)$$

such that, for each $z \in \text{dom}(\mu_1) \setminus \text{vars}(\bar{s}_1)$, properties 1 and 2 hold using μ_1 and $\bar{s}_1$. Note that, by (B.36), $t_1 \notin \text{vars}(\mu_1)$ and, in particular, $t_1 \notin \text{dom}(\mu_1)$.

Let

$$\mu \stackrel{\text{def}}{=} \{t_1 = s_1\mu_1\} \cup \mu_1. \tag{B.37}$$

As $s_1 \notin \text{Vars}$ and $\mu_1 \in \text{RSubst}$, $\mu \in \text{Eqs}$ has no identities or circular subsets so that $\mu \in \text{RSubst}$. By Lemma 41, $\mu \in \text{mgs}(\bar{s} = \bar{t})$.

As $t_1 \in \text{dom}(\mu)$ (by (B.37)) and $t_1 \notin \text{vars}(\bar{s})$ (by the assumption for this case), we have

$$\text{dom}(\mu_1) \setminus \text{vars}(\bar{s}_1) \cup \{t_1\} = \text{dom}(\mu) \setminus \text{vars}(\bar{s}).$$

Suppose that $z \in \text{dom}(\mu) \setminus \text{vars}(\bar{s})$. Then either $z \neq t_1$ so that $z\mu = z\mu_1$ and the inductive properties 1 and 2 using μ_1 and $\bar{s}_1$ can be applied to z or $z = t_1$ and $z\mu = s_1\mu_1$. We show that properties 1 and 2 using μ and $\bar{s}$ can be applied to z.

(1) Suppose $z \neq t_1$ so that $z\mu = z\mu_1$. Using property 1, $\text{vars}(z\mu_1) \subseteq \text{vars}(\bar{s}_1)$. As $\text{vars}(\bar{s}_1) \subseteq \text{vars}(\bar{s})$, it follows that $\text{vars}(z\mu) \subseteq \text{vars}(\bar{s})$.

Suppose that $z = t_1$ so that $z\mu = s_1\mu_1$. Let u be any variable in s_1. As $\bar{s}$ is linear, $u \notin \text{vars}(\bar{s}_1)$. Thus, if $u \in \text{dom}(\mu_1)$, we can use property 1 to derive that $\text{vars}(u\mu_1) \subseteq \text{vars}(\bar{s}_1)$. If $u \notin \text{dom}(\mu_1)$, then $u\mu_1 = u$ so that $\text{vars}(u\mu_1) \subseteq \text{vars}(s_1)$. Moreover $\text{vars}(s_1) \cup \text{vars}(\bar{s}_1) = \text{vars}(\bar{s})$ so that

$$\text{vars}(s_1\mu_1) \subseteq \text{vars}(\bar{s}). \tag{B.38}$$

Hence $\text{vars}(z\mu) \subseteq \text{vars}(\bar{s})$.

(2) Suppose $z \neq t_1$ so that $z\mu = z\mu_1$. Then, as property 2 holds, we have $\text{vars}(z\mu) \cap \text{dom}(\mu_1) = \varnothing$. Now $t_1 \notin \text{vars}(z\mu)$ because $\text{vars}(z\mu) \subseteq \text{vars}(\bar{s}_1)$ (by property 1) and $t_1 \notin \text{vars}(\bar{s}_1)$ (by (B.36)). Thus, as $\text{dom}(\mu) = \text{dom}(\mu_1) \cup \{t_1\}$, we have $\text{vars}(z\mu) \cap \text{dom}(\mu) = \varnothing$.

Suppose that $z = t_1$ so that $z\mu = s_1\mu_1$. Let u be any variable in $\text{vars}(s_1)$. Then, as $\bar{s}$ is linear, $u \notin \text{vars}(\bar{s}_1)$. Then either $u \in \text{dom}(\mu_1)$, and we can apply property 2 to u to obtain $\text{vars}(u\mu_1) \cap \text{dom}(\mu_1) = \varnothing$, or $u = u\mu_1$, and $\text{vars}(u\mu_1) \cap \text{dom}(\mu_1) = \varnothing$. Hence we have $\text{vars}(s_1\mu_1) \cap \text{dom}(\mu_1) = \varnothing$. Now $t_1 \notin \text{vars}(s_1\mu_1)$ because $\text{vars}(s_1\mu_1) \subseteq \text{vars}(\bar{s})$ (by (B.38)) and $t_1 \notin \text{vars}(\bar{s})$ (by the assumption for this case). Thus, as $\text{dom}(\mu) = \text{dom}(\mu_1) \cup \{t_1\}$, we have $\text{vars}(z\mu) \cap \text{dom}(\mu) = \varnothing$.

Case d. Assume that $t_1 \in \text{vars}(\bar{s}) \setminus \text{vars}(s_1)$ and $s_1 \notin \text{Vars}$.

Let

$$\bar{s}_1 \stackrel{\text{def}}{=} \Big(s_2\{t_1 \mapsto s_1\}, \dots, s_n\{t_1 \mapsto s_1\}\Big),$$
$$\bar{t}_1 \stackrel{\text{def}}{=} \Big(t_2\{t_1 \mapsto s_1\}, \dots, t_n\{t_1 \mapsto s_1\}\Big).$$

As $\bar{s}$ is linear, there is only one occurrence of t_1 in $\{s_2, \dots, s_n\}$, and, in $\bar{s}_1$, this is replaced by s_1 which is also linear. Thus $\bar{s}_1$ is linear, $\bar{s}_1 \subseteq \bar{s}$ and $t_1 \notin \mathrm{vars}(\bar{s}_1)$. Also, all occurrences of t_1 in $\bar{t}$ are replaced in $\bar{t}_1$ by s_1 so that $t_1 \notin \mathrm{vars}(\bar{t}_1)$. Thus

$$t_1 \notin \mathrm{vars}(\bar{s}_1) \cup \mathrm{vars}(\bar{t}_1). \tag{B.39}$$

It follows that $\mathrm{vars}(\bar{s}_1) \cup \mathrm{vars}(\bar{t}_1) \subset \mathrm{vars}(\bar{s}) \cup \mathrm{vars}(\bar{t})$ so that we can apply the inductive hypothesis to $\bar{s}_1$ and $\bar{t}_1$. Thus, there exists $\mu_1 \in \mathrm{RSubst}$ where

$$\mu_1 \in \mathrm{mgs}(\bar{s}_1 = \bar{t}_1)$$

such that, for each $z \in \mathrm{dom}(\mu_1) \setminus \mathrm{vars}(\bar{s}_1)$, properties 1 and 2 hold using μ_1 and $\bar{s}_1$.

Let

$$\mu \stackrel{\text{def}}{=} \{t_1 = s_1\mu_1\} \cup \mu_1.$$

By (B.39), $t_1 \notin \mathrm{vars}(\mu_1)$. Moreover $\mu_1 \in \mathrm{RSubst}$ and $s_1 \notin \mathrm{Vars}$ so that $\mu \in \mathrm{Eqs}$ has no identities or circular subset. Thus $\mu \in \mathrm{RSubst}$. By Lemma 41, $\mu \in \mathrm{mgs}(\bar{s} = \bar{t})$.

As $\mathrm{vars}(\bar{s}_1) \cup \{t_1\} = \mathrm{vars}(\bar{s})$ and $\mathrm{dom}(\mu_1) \cup \{t_1\} = \mathrm{dom}(\mu)$, we have

$$\mathrm{dom}(\mu_1) \setminus \mathrm{vars}(\bar{s}_1) = \mathrm{dom}(\mu) \setminus \mathrm{vars}(\bar{s}).$$

Suppose $z \in \mathrm{dom}(\mu) \setminus \mathrm{vars}(\bar{s})$. Then $z \neq t_1$, $z\mu = z\mu_1$ and the inductive properties 1 and 2 using μ_1 and $\bar{s}_1$ can be applied to z. We show that the properties 1 and 2 using μ and $\bar{s}$ can be applied to z.

(1) By property 1, $\mathrm{vars}(z\mu) \subseteq \mathrm{vars}(\bar{s}_1)$ and hence, as $\bar{s}_1 \subseteq \bar{s}$, $\mathrm{vars}(z\mu) \subseteq \mathrm{vars}(\bar{s})$.
(2) By property 2, we have $\mathrm{vars}(z\mu) \cap \mathrm{dom}(\mu_1) = \varnothing$. Now $t_1 \notin \mathrm{vars}(z\mu)$ because $t_1 \notin \mathrm{vars}(\bar{s}_1)$ (by (B.39)) and $\mathrm{vars}(z\mu) \subseteq \mathrm{vars}(\bar{s}_1)$ (by property 1). It follows that $\mathrm{vars}(z\mu) \cap \mathrm{dom}(\mu) = \varnothing$, since $\mathrm{dom}(\mu_1) \cup \{t_1\} = \mathrm{dom}(\mu)$.

□

Proposition 63 *Let $p \in P$ and $(x \mapsto t) \in \mathrm{Bind}$, where $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$. Let also $\sigma \in \gamma_P(p) \cap \mathrm{VSubst}$ and suppose that $\{r, r'\} = \{x, t\}$, $\mathrm{vars}(r) \subseteq \mathrm{hvars}(\sigma)$ and $\mathrm{rt}(r, \sigma) \in \mathrm{GTerms}$. Then, for all $\tau \in \mathrm{mgs}\big(\sigma \cup \{x = t\}\big)$, we have*

$$\mathrm{hvars}(\sigma) \cup \mathrm{vars}(r') \subseteq \mathrm{hvars}(\tau). \tag{B.40}$$

PROOF. If $\sigma \cup \{x = t\}$ is not satisfiable, the result is trivial. We therefore assume, for the rest of the proof, that $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$. It follows from Corollary 60 that we just have to show that

(1) $\mathrm{vars}(r') \subseteq \mathrm{hvars}(\tau)$, for some $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$;
(2) $\mathrm{hvars}(\sigma) \subseteq \mathrm{hvars}(\tau)$, for some $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$.

From these, we can then conclude that, for all $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$, (B.40) holds.

Note that, in both cases, since $\sigma \in \mathrm{VSubst}$ and $\mathrm{vars}(r) \subseteq \mathrm{hvars}(\sigma)$, by Proposition 55 we have $\mathrm{rt}(r, \sigma) = r\sigma$, so that $r\sigma \in \mathrm{HTerms} \cap \mathrm{GTerms}$.

We first prove statement 1. We must show that there exists $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$ such that $\mathrm{vars}(r') \subseteq \mathrm{hvars}(\tau)$.

As $\mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr) \neq \varnothing$, by Theorem 50 and the definition of mgs we can assume that there exists $\tau \in \mathrm{VSubst} \cap \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$. Thus

$$\tau \implies \bigl(\sigma \cup \{r = r'\}\bigr).$$

By Lemma 40 and the congruence axioms, we have $\tau \implies \{r\sigma = r'\}$. Since $\tau \in \mathrm{VSubst}$ and $r\sigma \in \mathrm{HTerms} \cap \mathrm{GTerms}$, Lemma 51 applies (with $s = r\sigma$) so that $r\sigma = r'\tau \in \mathrm{HTerms} \cap \mathrm{GTerms}$. Thus, by Proposition 54, $\mathrm{vars}(r') \subseteq \mathrm{hvars}(\tau)$.

We now prove statement 2. In this case, we show that there exists $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$ such that $\mathrm{hvars}(\sigma) \subseteq \mathrm{hvars}(\tau)$.

Let

$$\begin{aligned} \{u_1, \ldots, u_l\} &\stackrel{\mathrm{def}}{=} \mathrm{dom}(\sigma) \cap \mathrm{vars}(r'\sigma), \\ \bar{s} &\stackrel{\mathrm{def}}{=} (u_1, \ldots, u_l, r\sigma), \\ \bar{t} &\stackrel{\mathrm{def}}{=} (u_1\sigma, \ldots, u_l\sigma, r'\sigma). \end{aligned}$$

By Lemma 41 and the congruence axioms, $\sigma \cup \{x = t\} \implies \bar{s} = \bar{t}$. Thus, as $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$, $\mathrm{mgs}(\bar{s} = \bar{t}) \neq \varnothing$. Then, by Theorem 50, there exists $\mu \in \mathrm{VSubst} \cap \mathrm{mgs}(\bar{s} = \bar{t})$. Therefore, since $r\sigma \in \mathrm{HTerms} \cap \mathrm{GTerms}$ and $\mu \implies \{r\sigma = r'\sigma\}$, Lemma 51 applies (with $s = r\sigma$) so that we can conclude $r\sigma = r'\sigma\mu \in \mathrm{HTerms} \cap \mathrm{GTerms}$. Hence, for all $w \in \mathrm{dom}(\mu)$,

$$\mathrm{vars}(w\mu) = \varnothing. \tag{B.41}$$

Let

$$\nu \stackrel{\text{def}}{=} \Bigl\{\, z = z\sigma\mu \Bigm| z \in \mathrm{dom}(\sigma) \setminus \mathrm{vars}(r'\sigma) \,\Bigr\},$$
$$\tau \stackrel{\text{def}}{=} \nu \cup \mu.$$

Then, as $\sigma, \mu \in \mathrm{RSubst}$, it follows from (B.41) that $\nu, \tau \in \mathrm{Eqs}$ have no identities or circular subsets so that $\nu, \tau \in \mathrm{RSubst}$. By Lemma 41, $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$.

Suppose that $y \in \mathrm{hvars}(\sigma)$. Then we show that $y \in \mathrm{hvars}(\tau)$. Using Proposition 55, $\mathrm{rt}(y, \sigma) = y\sigma$ and

$$\mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) = \varnothing. \tag{B.42}$$

We show that $\mathrm{vars}(y\tau) \cap \mathrm{dom}(\tau) = \varnothing$. Now, if $y \notin \mathrm{dom}(\tau)$, the result holds trivially. Suppose that $y \in \mathrm{dom}(\nu)$, then $y\tau = y\sigma\mu$ and $y \in \mathrm{dom}(\sigma)$. Let w be any variable in $\mathrm{vars}(y\sigma)$ so that, by (B.42), $w \notin \mathrm{dom}(\sigma)$. If $w \notin \mathrm{dom}(\mu)$, then $w = w\mu \notin \mathrm{dom}(\tau)$. If $w \in \mathrm{dom}(\mu)$, then, by (B.41), $\mathrm{vars}(w\mu) = \varnothing$. Therefore, $\mathrm{vars}(w\mu) \cap \mathrm{dom}(\tau) = \varnothing$. It follows that $\mathrm{vars}(y\nu) \cap \mathrm{dom}(\tau) = \varnothing$. Finally, suppose $y \in \mathrm{dom}(\mu)$. Then, by (B.41), $\mathrm{vars}(y\mu) = \varnothing$. Therefore $\mathrm{vars}(y\mu) \cap \mathrm{dom}(\tau) = \varnothing$.

Therefore, using Definition 12, we have that $y \in \mathrm{hvars}(\tau)$ as required. □

Proposition 64 *Let $p \in P$ and $(x \mapsto t) \in \mathrm{Bind}$, where $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$. Let also $\sigma \in \gamma_P(p) \cap \mathrm{VSubst}$ and suppose that $x \in \mathrm{hvars}(\sigma)$ and $\mathrm{vars}(t) \subseteq \mathrm{hvars}(\sigma)$. Suppose also that $\mathrm{ind}_p(x, t)$ and that $\mathrm{or\_lin}_p(x, t)$ hold. Then, for all substitutions $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$,*

$$\mathrm{hvars}(\sigma) \subseteq \mathrm{hvars}(\tau). \tag{B.43}$$

PROOF. If $\sigma \cup \{x = t\}$ is not satisfiable, the result is trivial. We therefore assume, for the rest of the proof, that $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$. It follows from Corollary 60 that we just have to show that there exists $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$ such that (B.43) holds.

As $x \in \mathrm{hvars}(\sigma)$ and $\mathrm{vars}(t) \subseteq \mathrm{hvars}(\sigma)$, by using Proposition 55 we obtain $\mathrm{rt}(x, \sigma) = x\sigma$ and $\mathrm{rt}(t, \sigma) = t\sigma$. Also

$$\begin{aligned} \mathrm{vars}(x\sigma) \cap \mathrm{dom}(\sigma) &= \varnothing, \\ \mathrm{vars}(t\sigma) \cap \mathrm{dom}(\sigma) &= \varnothing. \end{aligned} \tag{B.44}$$

As $\mathrm{ind}_p(x, t)$ holds,

$$\mathrm{vars}(x\sigma) \cap \mathrm{vars}(t\sigma) = \varnothing. \tag{B.45}$$

By hypothesis, $\mathrm{or\_lin}(x,t)$ holds so that, by Definition 8, for some $r \in \{x,t\}$, $r\sigma$ is linear. Let $r' \stackrel{\mathrm{def}}{=} \{x,t\} \setminus \{r\}$.

By Lemma 41 and the congruence axioms, $\sigma \cup \{x = t\} \implies \{r\sigma = r'\sigma\}$. Thus, as $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$, $\mathrm{mgs}(r\sigma = r'\sigma) \neq \varnothing$. Thus we can apply Lemma 61 (where $\bar{s} = r\sigma$ and $\bar{t} = r'\sigma$) so that, using (B.45), there exists $\mu \in \mathrm{mgs}(x\sigma = t\sigma)$ such that, for all $w \in \mathrm{dom}(\mu)$,

$$\mathrm{vars}(w\mu) \cap \mathrm{dom}(\mu) = \varnothing. \tag{B.46}$$

Note that, by (B.44),

$$\mathrm{dom}(\sigma) \cap \mathrm{vars}(\mu) = \varnothing. \tag{B.47}$$

Let

$$\nu \stackrel{\mathrm{def}}{=} \left\{ z = z\sigma\mu \;\middle|\; z \in \mathrm{dom}(\sigma) \right\},$$
$$\tau \stackrel{\mathrm{def}}{=} \nu \cup \mu.$$

Then, as $\sigma, \mu \in \mathrm{RSubst}$, it follows from (B.47) that $\nu, \tau \in \mathrm{Eqs}$ have no identities or circular subsets so that $\nu, \tau \in \mathrm{RSubst}$. By Lemma 41, $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$.

Suppose $y \in \mathrm{hvars}(\sigma)$. Then we show that $y \in \mathrm{hvars}(\tau)$. As $y \in \mathrm{HTerms}$, we have, using Proposition 55, $\mathrm{rt}(y,\sigma) = y\sigma$ and

$$\mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) = \varnothing. \tag{B.48}$$

We show that $\mathrm{vars}(y\tau) \cap \mathrm{dom}(\tau) = \varnothing$. If $y \notin \mathrm{dom}(\tau)$, the result holds trivially. Suppose that $y \in \mathrm{dom}(\nu)$, then $y\tau = y\sigma\mu$. Let w be any variable in $\mathrm{vars}(y\sigma)$. Then, by (B.48), $w \notin \mathrm{dom}(\sigma)$. If $w \notin \mathrm{dom}(\mu)$, then $w = w\mu \notin \mathrm{dom}(\tau)$. If $w \in \mathrm{dom}(\mu)$, then $\mathrm{vars}(w\mu) \subseteq \mathrm{vars}(\mu)$ so that, by (B.47), $\mathrm{vars}(w\mu) \cap \mathrm{dom}(\nu) = \varnothing$. Moreover (B.46) applies so that $\mathrm{vars}(w\mu) \cap \mathrm{dom}(\mu) = \varnothing$. Therefore we have $\mathrm{vars}(w\mu) \cap \mathrm{dom}(\tau) = \varnothing$. It follows that $\mathrm{vars}(y\nu) \cap \mathrm{dom}(\tau) = \varnothing$. Finally, suppose $y \in \mathrm{dom}(\mu)$. Then $y\tau = y\mu$ and, by (B.47), we have $\mathrm{vars}(y\mu) \cap \mathrm{dom}(\nu) = \varnothing$. Also (B.46) applies where w is replaced by y so that $\mathrm{vars}(y\mu) \cap \mathrm{dom}(\mu) = \varnothing$. Thus $\mathrm{vars}(y\mu) \cap \mathrm{dom}(\tau) = \varnothing$.

Therefore, using Definition 12, we have that $y \in \mathrm{hvars}(\tau)$ as required. □

Proposition 65 *Let* $p \in P$ *and* $(x \mapsto t) \in \mathrm{Bind}$, *where* $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$. *Let also* $\sigma \in \gamma_P(p) \cap \mathrm{VSubst}$ *and suppose that* $x \in \mathrm{hvars}(\sigma)$ *and* $\mathrm{vars}(t) \subseteq \mathrm{hvars}(\sigma)$. *Suppose also that* $\mathrm{gfree}_p(x)$ *and* $\mathrm{gfree}_p(t)$ *hold. Then, for all* $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$, *we have*

$$\mathrm{hvars}(\sigma) \subseteq \mathrm{hvars}(\tau). \tag{B.49}$$

PROOF. If $\sigma \cup \{x = t\}$ is not satisfiable, the result is trivial. We therefore assume, for the rest of the proof, that $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$. It follows from Corollary 60 that we just have to show that there exists $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$ such that (B.49) holds.

By Definition 8, $\mathrm{gfree}_p(x)$ and $\mathrm{gfree}_p(t)$ imply that either $\mathrm{rt}(x, \sigma) \in \mathrm{GTerms}$ or $\mathrm{rt}(x, \sigma) \in \mathrm{Vars}$, and either $\mathrm{rt}(t, \sigma) \in \mathrm{GTerms}$ or $\mathrm{rt}(t, \sigma) \in \mathrm{Vars}$. Since we have $\mathrm{rt}(x, \sigma), \mathrm{rt}(t, \sigma) \in \mathrm{HTerms}$ and $\sigma \in \mathrm{VSubst}$, as a consequence of Proposition 55, we have $\mathrm{rt}(x, \sigma) = x\sigma$, $\mathrm{rt}(t, \sigma) = t\sigma$ and $x\sigma, t\sigma \notin \mathrm{dom}(\sigma)$. There are three cases:

- $\mathrm{vars}(x\sigma) = \varnothing \vee \mathrm{vars}(t\sigma) = \varnothing$. Then the result follows from Proposition 63.
- $x\sigma = t\sigma \in \mathrm{Vars}$. Then letting $\tau = \sigma$ gives the required result.
- $x\sigma, t\sigma \in \mathrm{Vars}$ are distinct variables. Let $\tau = \sigma \cup \{x\sigma = t\sigma\}$. Then, as $x\sigma, t\sigma \notin \mathrm{dom}(\sigma)$, $\tau \in \mathrm{RSubst}$. Hence, by Lemma 41, $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$. Let y be any variable in $\mathrm{hvars}(\sigma)$. We show that $y \in \mathrm{hvars}(\tau)$.
  Suppose first that $y \neq x\sigma$. Then $y\tau = y\sigma$. Thus using Proposition 55, $\mathrm{rt}(y, \sigma) = y\tau$ and $\mathrm{vars}(y\tau) \cap \mathrm{dom}(\sigma) = \varnothing$. Thus $\mathrm{vars}(y\tau) \cap \mathrm{dom}(\tau) \subseteq \{x\sigma\}$. However, $x\sigma\tau = t\sigma \notin \mathrm{dom}(\tau)$ so that, by Definition 11, $\mathrm{vars}(y\tau) \subseteq \mathrm{hvars}_1(\tau)$ and hence $y \in \mathrm{hvars}_2(\tau)$. Therefore, by Definitions 11 and 12, we have $y \in \mathrm{hvars}(\tau)$.
  Secondly, suppose that $y = x\sigma$. Then $y\tau = t\sigma$. So that, as $t\sigma \in \mathrm{Vars} \setminus \mathrm{dom}(\sigma)$ and $x\sigma \neq t\sigma$, $\mathrm{vars}(y\tau) \cap \mathrm{dom}(\tau) = \varnothing$. Therefore, using Definition 12, we have that $y \in \mathrm{hvars}(\tau)$ as required.

$\square$

Proposition 66 *Let $p \in P$ and $(x \mapsto t) \in \mathrm{Bind}$, where $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$. Let $\sigma \in \gamma_P(p) \cap \mathrm{VSubst}$ and suppose that $x \in \mathrm{hvars}(\sigma)$ and $\mathrm{vars}(t) \subseteq \mathrm{hvars}(\sigma)$. Furthermore, suppose that $\mathrm{or\_lin}_p(x, t)$ and $\mathrm{share\_lin}_p(x, t)$ hold. Then, for all substitutions $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$, we have*

$$\mathrm{hvars}(\sigma) \setminus \mathrm{share\_same\_var}_p(x, t) \subseteq \mathrm{hvars}(\tau). \tag{B.50}$$

PROOF. If $\sigma \cup \{x = t\}$ is not satisfiable, the result is trivial. We therefore assume, for the rest of the proof, that $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$. It follows from Corollary 60 that we just have to show that there exists $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$ such that (B.50) holds.

As $x \in \mathrm{hvars}(\sigma)$ and $\mathrm{vars}(t) \subseteq \mathrm{hvars}(\sigma)$, by using Proposition 55 we obtain $\mathrm{rt}(x, \sigma) = x\sigma$ and $\mathrm{rt}(t, \sigma) = t\sigma$. Also

$$\mathrm{vars}(x\sigma) \cap \mathrm{dom}(\sigma) = \varnothing, \quad \mathrm{vars}(t\sigma) \cap \mathrm{dom}(\sigma) = \varnothing. \tag{B.51}$$

By hypothesis, $\mathrm{or\_lin}_p(x, t)$ holds so that, by Definition 8, for some $r \in \{x, t\}$, $r\sigma$ is linear. Also by hypothesis, $\mathrm{share\_lin}_p(x, t)$ holds so that, by Definition 8, if $r' = \{x, t\} \setminus \{r\}$, for all $z \in \mathrm{vars}(r\sigma) \cap \mathrm{vars}(r'\sigma)$, $\mathrm{occ\_lin}(z, r'\sigma)$ holds. Therefore,

$$\mathrm{vars}(r\sigma) \cap \mathrm{nlvars}(r'\sigma) = \varnothing. \tag{B.52}$$

By Lemma 41 and the congruence axioms, $\sigma \cup \{x = t\} \implies \{r\sigma = r'\sigma\}$. Thus, as $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$, $\mathrm{mgs}(r\sigma = r'\sigma) \neq \varnothing$. Thus, as $r\sigma$ is linear and (B.52) holds, we can apply Lemma 61 (where $\bar{s} = r\sigma$ and $\bar{t} = r'\sigma$) so that there exists $\mu \in \mathrm{mgs}(x\sigma = t\sigma)$ such that, for all $w \in \mathrm{dom}(\mu) \setminus \big(\mathrm{vars}(x\sigma) \cap \mathrm{vars}(t\sigma)\big)$,

$$\mathrm{vars}(w\mu) \cap \mathrm{dom}(\mu) = \varnothing. \tag{B.53}$$

Note that, by (B.51),

$$\mathrm{dom}(\sigma) \cap \mathrm{vars}(\mu) = \varnothing. \tag{B.54}$$

Let

$$\nu \stackrel{\mathrm{def}}{=} \big\{\, z = z\sigma\mu \;\big|\; z \in \mathrm{dom}(\sigma) \,\big\},$$
$$\tau \stackrel{\mathrm{def}}{=} \nu \cup \mu.$$

Then, as $\sigma, \mu \in \mathrm{RSubst}$, it follows from (B.54) that $\nu, \tau \in \mathrm{Eqs}$ have no identities or circular subsets so that $\nu, \tau \in \mathrm{RSubst}$. By Lemma 41, $\tau \in \mathrm{mgs}\big(\sigma \cup \{x = t\}\big)$.

Suppose $y \in \mathrm{hvars}(\sigma) \setminus \mathrm{share\_same\_var}_p(x, t)$. We show that $y \in \mathrm{hvars}(\tau)$. As $y \in \mathrm{hvars}(\sigma)$, using Proposition 55, $\mathrm{rt}(y, \sigma) = y\sigma$ and

$$\mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) = \varnothing. \tag{B.55}$$

As $y \notin \mathrm{share\_same\_var}_p(x, t)$, by Definition 8,

$$\mathrm{vars}(y\sigma) \cap \mathrm{vars}(x\sigma) \cap \mathrm{vars}(t\sigma) = \varnothing. \tag{B.56}$$

Therefore, using (B.56) if $y \notin \mathrm{dom}(\sigma)$ and (B.51) if $y \in \mathrm{dom}(\sigma)$, it follows that

$$y \notin \mathrm{vars}(x\sigma) \cap \mathrm{vars}(t\sigma). \tag{B.57}$$

We show that $\mathrm{vars}(y\tau) \cap \mathrm{dom}(\tau) = \varnothing$. Now, if $y \notin \mathrm{dom}(\tau)$, the result holds trivially. Suppose that $y \in \mathrm{dom}(\nu)$, then $y\tau = y\sigma\mu$. Let w be any variable in $\mathrm{vars}(y\sigma)$. Then, by (B.56), $w \notin \big(\mathrm{vars}(x\sigma) \cap \mathrm{vars}(t\sigma)\big)$ and, by (B.55), $w \notin \mathrm{dom}(\sigma)$. If $w \notin \mathrm{dom}(\mu)$, then $w = w\mu \notin \mathrm{dom}(\tau)$. If $w \in \mathrm{dom}(\mu)$, then $\mathrm{vars}(w\mu) \subseteq \mathrm{vars}(\mu)$ so that, by (B.54), we also have $\mathrm{vars}(w\mu) \cap \mathrm{dom}(\nu) = \varnothing$. Moreover (B.53) applies so that $\mathrm{vars}(w\mu) \cap \mathrm{dom}(\mu) = \varnothing$. Therefore, $\mathrm{vars}(w\mu) \cap \mathrm{dom}(\tau) = \varnothing$. It follows that $\mathrm{vars}(y\nu) \cap \mathrm{dom}(\tau) = \varnothing$. Finally, suppose $y \in$

$\mathrm{dom}(\mu)$. Then $y\tau = y\mu$ and, by (B.54), $\mathrm{vars}(y\mu) \cap \mathrm{dom}(\nu) = \varnothing$. As (B.57) holds, (B.53) applies where w is replaced by y so that $\mathrm{vars}(y\mu) \cap \mathrm{dom}(\mu) = \varnothing$. Thus $\mathrm{vars}(y\mu) \cap \mathrm{dom}(\tau) = \varnothing$.

Therefore, using Definition 12, we have that $y \in \mathrm{hvars}(\tau)$ as required. □

Proposition 67 *Let $p \in P$ and $(x \mapsto t) \in \mathrm{Bind}$, where $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$. Let also $\sigma \in \gamma_P(p) \cap \mathrm{VSubst}$ and suppose that $\{r, r'\} = \{x, t\}$, $\mathrm{vars}(r) \subseteq \mathrm{hvars}(\sigma)$ and $\mathrm{lin}_p(r)$ holds. Then, for all $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$, we have*

$$\mathrm{hvars}(\sigma) \setminus \mathrm{share\_with}_p(r) \subseteq \mathrm{hvars}(\tau). \tag{B.58}$$

PROOF. If $\sigma \cup \{x = t\}$ is not satisfiable, the result is trivial. We therefore assume, for the rest of the proof, that $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$. It follows from Corollary 60 that we just have to show that there exists $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$ such that (B.58) holds.

By hypothesis, $\mathrm{vars}(r) \subseteq \mathrm{hvars}(\sigma)$. Hence, by Proposition 55, $\mathrm{rt}(r, \sigma) = r\sigma$ and

$$\mathrm{vars}(r\sigma) \cap \mathrm{dom}(\sigma) = \varnothing. \tag{B.59}$$

By hypothesis, $\mathrm{lin}_p(r)$ holds, so that, by Definition 8, $r\sigma$ is linear.

Let

$$\begin{aligned}
\{u_1, \dots, u_l\} &\stackrel{\mathrm{def}}{=} \mathrm{dom}(\sigma) \cap \Bigl(\mathrm{vars}(x\sigma) \cup \mathrm{vars}(t\sigma)\Bigr),\\
\bar{s} &\stackrel{\mathrm{def}}{=} (u_1, \dots, u_l, r\sigma),\\
\bar{t} &\stackrel{\mathrm{def}}{=} (u_1\sigma, \dots, u_l\sigma, r'\sigma).
\end{aligned}$$

Since $r\sigma$ is linear, it follows from (B.59) that $\bar{s}$ is linear. By Lemma 41 and the congruence axioms, $\sigma \cup \{x = t\} \implies \bar{s} = \bar{t}$. Thus, as $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$, we have $\mathrm{mgs}(\bar{s} = \bar{t}) \neq \varnothing$. Therefore, we can apply Lemma 62 so that there exists $\mu \in \mathrm{mgs}(\bar{s} = \bar{t})$ such that, for all $w \in \mathrm{dom}(\mu) \setminus \mathrm{vars}(\bar{s})$,

$$\mathrm{vars}(w\mu) \cap \mathrm{dom}(\mu) = \varnothing. \tag{B.60}$$

Note that, since $\sigma \in \mathrm{VSubst}$, for each $i = 1, \dots, l$, we have

$$\mathrm{vars}(u_i\sigma) \subseteq \mathrm{vars}(x\sigma) \cup \mathrm{vars}(t\sigma).$$

Thus

$$\mathrm{vars}(\mu) \subseteq \mathrm{vars}(x\sigma) \cup \mathrm{vars}(t\sigma). \tag{B.61}$$

Let

$$\nu \stackrel{\text{def}}{=} \left\{ z = z\sigma\mu \;\middle|\; z \in \mathrm{dom}(\sigma) \setminus \bigl(\mathrm{vars}(x\sigma) \cup \mathrm{vars}(t\sigma)\bigr) \right\},$$
$$\tau \stackrel{\text{def}}{=} \nu \cup \mu.$$

Then, as $\sigma, \mu \in \mathrm{RSubst}$, it follows from (B.61) that $\nu, \tau \in \mathrm{Eqs}$ have no identities or circular subsets so that $\nu, \tau \in \mathrm{RSubst}$. By Lemma 41, $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$.

Suppose $y \in \mathrm{hvars}(\sigma) \setminus \mathrm{share\_with}_p(r)$. Then we show that $y \in \mathrm{hvars}(\tau)$. As $y \in \mathrm{hvars}(\sigma)$, by Proposition 55, $\mathrm{rt}(y, \sigma) = y\sigma$ and

$$\mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) = \varnothing. \tag{B.62}$$

As $y \notin \mathrm{share\_with}_p(r)$, by Definition 8, $y \notin \mathrm{share\_same\_var}_p(y, r)$ so that, using the same definition,

$$\mathrm{vars}(y\sigma) \cap \mathrm{vars}(r\sigma) = \varnothing. \tag{B.63}$$

Therefore using (B.63) if $y \notin \mathrm{dom}(\sigma)$ and (B.59) if $y \in \mathrm{dom}(\sigma)$, it follows that

$$y \notin \mathrm{vars}(r\sigma). \tag{B.64}$$

We show that $\mathrm{vars}(y\tau) \cap \mathrm{dom}(\tau) = \varnothing$. Now, if $y \notin \mathrm{dom}(\tau)$, the result holds trivially. Suppose that $y \in \mathrm{dom}(\nu)$. Then $y\tau = y\sigma\mu$ and $y \in \mathrm{dom}(\sigma)$. It follows from (B.62) and (B.63) that $\mathrm{vars}(y\sigma) \cap \mathrm{vars}(\bar{s}) = \varnothing$. Let w be any variable in $\mathrm{vars}(y\sigma)$ so that $w \notin \mathrm{vars}(\bar{s})$. By (B.62), we have $w \notin \mathrm{dom}(\sigma)$. If $w \notin \mathrm{dom}(\mu)$, then we have $w = w\mu \notin \mathrm{dom}(\tau)$. If $w \in \mathrm{dom}(\mu)$, then $\mathrm{vars}(w\mu) \subseteq \mathrm{vars}(\mu)$ so that, by (B.61), $\mathrm{vars}(w\mu) \cap \mathrm{dom}(\nu) = \varnothing$. Moreover (B.60) applies so that $\mathrm{vars}(w\mu) \cap \mathrm{dom}(\mu) = \varnothing$. Therefore, $\mathrm{vars}(w\mu) \cap \mathrm{dom}(\tau) = \varnothing$. It follows that $\mathrm{vars}(y\nu) \cap \mathrm{dom}(\tau) = \varnothing$. Finally, suppose $y \in \mathrm{dom}(\mu)$. Then $y\tau = y\mu$ and, by (B.61), $\mathrm{vars}(y\mu) \cap \mathrm{dom}(\nu) = \varnothing$. Since $\sigma \in \mathrm{VSubst}$ and $y \in \mathrm{hvars}(\sigma)$, we have $y \notin \mathrm{dom}(\sigma) \cap \bigl(\mathrm{vars}(r\sigma) \cup \mathrm{vars}(r'\sigma)\bigr)$ and hence $y \notin \mathrm{vars}(\bar{s})$. Therefore (B.60) applies and $\mathrm{vars}(y\mu) \cap \mathrm{dom}(\mu) = \varnothing$. Thus $\mathrm{vars}(y\mu) \cap \mathrm{dom}(\tau) = \varnothing$.

Therefore, using Definition 12, we have that $y \in \mathrm{hvars}(\tau)$ as required. □

Proposition 68 *Let $p \in P$ and $(x \mapsto t) \in \mathrm{Bind}$, where $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$. Let also $\sigma \in \gamma_P(p) \cap \mathrm{VSubst}$. Then, for all $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$,*

$$\mathrm{hvars}(\sigma) \setminus \bigl(\mathrm{share\_with}_p(x) \cup \mathrm{share\_with}_p(t)\bigr) \subseteq \mathrm{hvars}(\tau). \tag{B.65}$$

PROOF. If $\sigma \cup \{x = t\}$ is not satisfiable, the result is trivial. We therefore assume, for the rest of the proof, that $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$. It follows from Corollary 60 that we just have to show that there exists $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$ such that (B.65) holds.

Let

$$\begin{aligned}
\{u_1, \dots, u_l\} &\stackrel{\mathrm{def}}{=} \mathrm{dom}(\sigma) \cap \bigl(\mathrm{vars}(x\sigma) \cup \mathrm{vars}(t\sigma)\bigr), \\
\bar{s} &\stackrel{\mathrm{def}}{=} (u_1, \dots, u_l, x\sigma), \\
\bar{t} &\stackrel{\mathrm{def}}{=} (u_1\sigma, \dots, u_l\sigma, t\sigma).
\end{aligned}$$

Note that, since $\sigma \in \mathrm{VSubst}$, for each $i = 1, \dots, l$, we have

$$\mathrm{vars}(u_i\sigma) \subseteq \mathrm{vars}(x\sigma) \cup \mathrm{vars}(t\sigma).$$

Thus, for any $\mu \in \mathrm{mgs}(\bar{s} = \bar{t})$, we have

$$\mathrm{vars}(\mu) \subseteq \mathrm{vars}(x\sigma) \cup \mathrm{vars}(t\sigma). \tag{B.66}$$

Let

$$\begin{aligned}
\nu &\stackrel{\mathrm{def}}{=} \Bigl\{\, z = z\sigma\mu \Bigm| z \in \mathrm{dom}(\sigma) \setminus \bigl(\mathrm{vars}(x\sigma) \cup \mathrm{vars}(t\sigma)\bigr) \,\Bigr\}, \\
\tau &\stackrel{\mathrm{def}}{=} \nu \cup \mu.
\end{aligned}$$

Then, as $\sigma, \mu \in \mathrm{RSubst}$, it follows from (B.66) that $\nu, \tau \in \mathrm{Eqs}$ have no identities or circular subsets so that $\nu, \tau \in \mathrm{RSubst}$. Thus, using Lemma 41 and the assumption that $\sigma \cup \{x = t\}$ is satisfiable in $\mathcal{RT}$, $\tau \in \mathrm{mgs}\bigl(\sigma \cup \{x = t\}\bigr)$.

Suppose that $y \in \mathrm{hvars}(\sigma) \setminus \bigl(\mathrm{share\_with}_p(x) \cup \mathrm{share\_with}_p(t)\bigr)$. We show that $y \in \mathrm{hvars}(\tau)$. As $y \in \mathrm{hvars}(\sigma)$, by Proposition 55, $\mathrm{rt}(y, \sigma) = y\sigma$ and

$$\mathrm{vars}(y\sigma) \cap \mathrm{dom}(\sigma) = \varnothing. \tag{B.67}$$

As $y \notin \mathrm{share\_with}_p(x) \cup \mathrm{share\_with}_p(t)$, it follows from Definition 8 that

$$y \notin \mathrm{share\_same\_var}_p(y, x) \cup \mathrm{share\_same\_var}_p(y, t)$$

so that, using the same definition with the result that $\mathrm{rt}(y, \sigma) = y\sigma$, we obtain

$$\mathrm{vars}(y\sigma) \cap \bigl(\mathrm{vars}(x\sigma) \cup \mathrm{vars}(t\sigma)\bigr) = \varnothing. \tag{B.68}$$

Therefore, using (B.68) if $y \notin \mathrm{dom}(\sigma)$ and using the fact that $\sigma \in \mathrm{VSubst}$, if $y \in \mathrm{dom}(\sigma)$, it follows that

$$y \notin \mathrm{vars}(x\sigma) \cup \mathrm{vars}(t\sigma). \tag{B.69}$$

We show that $\text{vars}(y\tau) \cap \text{dom}(\tau) = \varnothing$. Now, if $y \notin \text{dom}(\tau)$, the result holds trivially. Suppose that $y \in \text{dom}(\tau)$. Then, by (B.66) and (B.69), $y \notin \text{vars}(\mu)$ so that $y \notin \text{dom}(\mu)$ and $\text{vars}(y\mu) \cap \text{dom}(\mu) = \varnothing$. Thus we must have $y \in \text{dom}(\nu)$ and $y\tau = y\sigma$. Then, by (B.66) and (B.68), $\text{vars}(y\sigma) \cap \text{dom}(\mu) = \varnothing$. Moreover, by (B.67), $\text{vars}(y\sigma) \cap \text{dom}(\sigma) = \varnothing$. It follows that $\text{vars}(y\sigma) \cap \text{dom}(\tau) = \varnothing$ and hence, as $y\sigma = y\tau$, $\text{vars}(y\tau) \cap \text{dom}(\tau) = \varnothing$.

Therefore, using Definition 12, we have that $y \in \text{hvars}(\tau)$ as required. □

Proof of Theorem 19 on page 21. By hypothesis, $\sigma \in \gamma_P(p)$. By Theorem 50, there exists $\sigma' \in \text{VSubst}$ such that $\mathcal{RT} \vdash \forall(\sigma \leftrightarrow \sigma')$. By Proposition 59, as σ, σ' are satisfiable in $\mathcal{RT}$, we have that $\text{hvars}(\sigma) = \text{hvars}(\sigma')$. By Definition 7, $\sigma \in \gamma_P(p)$ if and only if $\sigma' \in \gamma_P(p)$. We therefore safely assume that $\sigma \in \text{VSubst}$.

By hypothesis, we have $\sigma \in \gamma_H(h)$. Therefore, it follows from Definition 16 that $h \subseteq \text{hvars}(\sigma)$. Similarly, by Definition 16, in order to prove $\tau \in \gamma_H(h')$, we just need to show that $h' \subseteq \text{hvars}(\tau)$ where h' is as defined in Definition 18. There are eight cases that have to be considered.

(1) $\text{hterm}_h(x) \wedge \text{ground}_p(x)$ holds.
As $\text{hterm}_h(x)$ holds, by Definition 18, $x \in h$. Hence, by Definition 16, we have $x \in \text{hvars}(\sigma)$. As $\text{ground}_p(x)$ holds, by Definition 8, $\text{rt}(x, \sigma) \in \text{GTerms}$. Therefore we can apply Proposition 63, where r is replaced by x and r' by t, to conclude that

$$\text{hvars}(\sigma) \cup \text{vars}(t) \subseteq \text{hvars}(\tau).$$

(2) $\text{hterm}_h(t) \wedge \text{ground}_p(t)$ holds.
As $\text{hterm}_h(t)$ holds, by Definition 18, $\text{vars}(t) \subseteq h$. Hence, by Definition 16, $\text{vars}(t) \subseteq \text{hvars}(\sigma)$. As $\text{ground}_p(t)$ holds, by Definition 8, $\text{rt}(t, \sigma) \in \text{GTerms}$. Therefore we can apply Proposition 63, where r is replaced by t and r' by x, to conclude that

$$\text{hvars}(\sigma) \cup \{x\} \subseteq \text{hvars}(\tau).$$

(3) $\text{hterm}_h(x) \wedge \text{hterm}_h(t) \wedge \text{ind}_p(x, t) \wedge \text{or\_lin}_p(x, t)$ holds.
As $\text{hterm}_h(x)$ and $\text{hterm}_h(t)$ hold, by Definition 18, $x \in h$ and $\text{vars}(t) \subseteq h$. Hence, by Definition 16, $x \in \text{hvars}(\sigma)$ and $\text{vars}(t) \subseteq \text{hvars}(\sigma)$. Therefore we can apply Proposition 64 to conclude that

$$\text{hvars}(\sigma) \subseteq \text{hvars}(\tau).$$

(4) $\text{hterm}_h(x) \wedge \text{hterm}_h(t) \wedge \text{gfree}_p(x) \wedge \text{gfree}_p(t)$ holds.

As $\mathrm{hterm}_h(x)$ and $\mathrm{hterm}_h(t)$ hold, by Definition 18, $x \in h$ and $\mathrm{vars}(t) \subseteq h$. Hence, by Definition 16, $x \in \mathrm{hvars}(\sigma)$ and $\mathrm{vars}(t) \subseteq \mathrm{hvars}(\sigma)$. Therefore we can apply Proposition 65 to conclude that

$$\mathrm{hvars}(\sigma) \subseteq \mathrm{hvars}(\tau).$$

(5) $\mathrm{hterm}_h(x) \wedge \mathrm{hterm}_h(t) \wedge \mathrm{share\_lin}_p(x,t) \wedge \mathrm{or\_lin}_p(x,t)$ holds.
As $\mathrm{hterm}_h(x)$ and $\mathrm{hterm}_h(t)$ hold, by Definition 18, $x \in h$ and $\mathrm{vars}(t) \subseteq h$. Hence, by Definition 16, $x \in \mathrm{hvars}(\sigma)$ and $\mathrm{vars}(t) \subseteq \mathrm{hvars}(\sigma)$. Therefore we can apply Proposition 66 to conclude that

$$\mathrm{hvars}(\sigma) \setminus \mathrm{share\_same\_var}_p(x,t) \subseteq \mathrm{hvars}(\tau).$$

(6) $\mathrm{hterm}_h(x) \wedge \mathrm{lin}_p(x)$ holds.
As $\mathrm{hterm}_h(x)$ holds, by Definition 18, $x \in h$. Hence, by Definition 16, we have $x \in \mathrm{hvars}(\sigma)$. Therefore we can apply Proposition 67 where r is replaced by x and r' by t, to conclude that

$$\mathrm{hvars}(\sigma) \setminus \mathrm{share\_with}_p(x) \subseteq \mathrm{hvars}(\tau).$$

(7) $\mathrm{hterm}_h(t) \wedge \mathrm{lin}_p(t)$ holds.
As $\mathrm{hterm}_h(t)$ holds, by Definition 18, $\mathrm{vars}(t) \subseteq h$. Hence, by Definition 16, $\mathrm{vars}(t) \subseteq \mathrm{hvars}(\sigma)$. Therefore we can apply Proposition 67 where r is replaced by t and r' by x, to conclude that

$$\mathrm{hvars}(\sigma) \setminus \mathrm{share\_with}_p(t) \subseteq \mathrm{hvars}(\tau).$$

(8) For all $(x \mapsto t) \in \mathrm{Bind}$ where $\{x\} \cup \mathrm{vars}(t) \subseteq \mathrm{VI}$, Proposition 68 applies so that

$$\mathrm{hvars}(\sigma) \setminus \Bigl(\mathrm{share\_with}_p(x) \cup \mathrm{share\_with}_p(t)\Bigr) \subseteq \mathrm{hvars}(\tau).$$

□

Proof of Theorem 21 on page 22. Suppose that $\tau \in \exists\, x \,.\, \{\sigma\}$. We need to show that $\tau \in \gamma_H\Bigl(\mathrm{proj}_H(h,x)\Bigr)$.

Let $\overline{V} = \mathrm{Vars} \setminus \mathrm{VI}$. Then, by Definition 5, $\mathcal{RT} \vdash \forall\Bigl(\exists \overline{V} \,.\, (\tau \leftrightarrow \exists x \,.\, \sigma)\Bigr)$. Thus we have

$$\mathcal{RT} \vdash \forall\biggl(\Bigl(\exists \overline{V} \,.\, \tau\Bigr) \leftrightarrow \Bigl(\exists \overline{V} \cup \{x\} \,.\, \sigma\Bigr)\biggr). \tag{B.70}$$

Suppose $v \in \overline{V} \setminus \mathrm{vars}(\sigma)$. As we assumed that Vars is denumerable and that VI is finite, such a v will exist. Moreover, as $x \in \mathrm{VI}$, we have $x \neq v$. Let $\sigma' \in \mathrm{RSubst}$ be obtained from σ by replacing every occurrence of x by v. Formally, if $\rho = \{x \mapsto v\}$, let

$$\sigma' \stackrel{\mathrm{def}}{=} \Bigl\{\, y \mapsto y\sigma\rho \Bigm| y \in \mathrm{dom}(\sigma) \setminus \{x\} \,\Bigr\} \cup \sigma'',$$

where $\sigma'' = \{v \mapsto x\sigma\rho\}$ if $x \in \operatorname{dom}(\sigma)$ and $\varnothing$ otherwise. Then $\sigma' \in \mathrm{RSubst}$ and

$$\mathcal{RT} \vdash \forall\Big(\big(\exists \overline{V} \,.\, \sigma'\big) \leftrightarrow \big(\exists \overline{V \cup \{x\}} \,.\, \sigma\big)\Big).$$

Thus, by (B.70), $\mathcal{RT} \vdash \forall\big((\exists \overline{V} \,.\, \tau) \leftrightarrow (\exists \overline{V} \,.\, \sigma')\big)$. Therefore, by Proposition 59,

$$\operatorname{hvars}(\tau) \cap \mathrm{VI} = \operatorname{hvars}(\sigma') \cap \mathrm{VI}. \tag{B.71}$$

As $\sigma' \in \mathrm{RSubst}$ and $x \notin \operatorname{dom}(\sigma')$, $\operatorname{rt}(x, \sigma') = x$ so that, by Proposition 12, $x \in \operatorname{hvars}(\sigma')$. Also, as σ' is obtained from σ by renaming x to the new variable v, $\operatorname{hvars}(\sigma') \supseteq \operatorname{hvars}(\sigma) \setminus \{v\}$. Since $v \notin \mathrm{VI}$, we have

$$\operatorname{hvars}(\sigma') \cap \mathrm{VI} \supseteq \big(\operatorname{hvars}(\sigma) \cup \{x\}\big) \cap \mathrm{VI}.$$

Therefore, by (B.71),

$$\operatorname{hvars}(\tau) \cap \mathrm{VI} \supseteq \big(\operatorname{hvars}(\sigma) \cup \{x\}\big) \cap \mathrm{VI}. \tag{B.72}$$

By hypothesis, $\sigma \in \gamma_H(h)$, so that, by Definition 16, $\operatorname{hvars}(\sigma) \supseteq h$. Therefore, by (B.72), $\operatorname{hvars}(\tau) \cap \mathrm{VI} \supseteq \big(h \cup \{x\}\big) \cap \mathrm{VI}$. Thus, by applying Definition 16, we can conclude that $\tau \in \gamma_H\big(h \cup \{x\}\big)$. $\square$

B.7 Finite-Tree Dependencies

The proof of Theorem 23 depends on the fact that finite-tree dependencies only capture permanent information and that the γ_F function is *meet-preserving.*

Proposition 69 *Let* $\sigma, \tau \in \mathrm{RSubst}$ *and* $\phi \in \mathrm{Bfun}$*, where* $\sigma \in \gamma_F(\phi)$ *and* $\tau \in \downarrow\sigma$*. Then* $\tau \in \gamma_F(\phi)$*.*

PROOF. By the hypothesis, $\tau \in \downarrow\sigma$, so that, for each $\upsilon \in \downarrow\tau$, $\upsilon \in \downarrow\sigma$. Therefore, as $\sigma \in \gamma_F(\phi)$, it follows from Definition 22 that, for all $\upsilon \in \downarrow\tau$, $\phi\big(\operatorname{hval}(\upsilon)\big) = 1$ and hence $\tau \in \gamma_F(\phi)$. $\square$

Proposition 70 *Let* $\phi_1, \phi_2 \in \mathrm{Bfun}$*. Then*

$$\gamma_F(\phi_1 \wedge \phi_2) = \gamma_F(\phi_1) \cap \gamma_F(\phi_2).$$

PROOF.

$$\gamma_F(\phi_1 \wedge \phi_2) = \Big\{\, \sigma \in \mathrm{RSubst} \;\Big|\; \forall \tau \in \downarrow\sigma : (\phi_1 \wedge \phi_2)\big(\operatorname{hval}(\tau)\big) = 1 \,\Big\}$$

$$
\begin{aligned}
&= \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\,\sigma : \forall i \in \{1,2\} : \phi_i\bigl(\mathrm{hval}(\tau)\bigr) = 1 \,\Bigr\} \\
&= \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\,\sigma : \phi_1\bigl(\mathrm{hval}(\tau)\bigr) = 1 \,\Bigr\} \\
&\qquad \cap \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\,\sigma : \phi_2\bigl(\mathrm{hval}(\tau)\bigr) = 1 \,\Bigr\} \\
&= \gamma_F(\phi_1) \cap \gamma_F(\phi_2).
\end{aligned}
$$

□

Proof of Theorem 23 on page 25. Assuming the hypothesis of the theorem, we will prove each relation separately.

(23a). Let $\sigma = \{x \mapsto t\}$ and suppose that $\tau \in {\downarrow}\,\sigma$. Then, by Proposition 2, $\mathcal{RT} \vdash \forall\bigl(\tau \rightarrow \sigma\bigr)$. It follows from Lemma 42 that $\mathrm{rt}(x,\tau) = \mathrm{rt}(t,\tau)$ and thus, by Proposition 13, $x \in \mathrm{hvars}(\tau)$ if and only if $\mathrm{vars}(t) \subseteq \mathrm{hvars}(\tau)$. This is equivalent to $\bigl(x \leftrightarrow \bigwedge \mathrm{vars}(t)\bigr)\bigl(\mathbf{0}\bigl[1/\mathrm{hvars}(\tau)\bigr]\bigr) = 1$ and, by Definition 22, to $\bigl(x \leftrightarrow \bigwedge \mathrm{vars}(t)\bigr)\bigl(\mathrm{hval}(\tau)\bigr) = 1$. As this holds for all $\tau \in {\downarrow}\,\sigma$, by Definition 22, $\sigma \in \gamma_F\bigl(x \leftrightarrow \bigwedge \mathrm{vars}(t)\bigr)$.

(23b). Let $\sigma = \{x \mapsto t\}$, where $x \in \mathrm{vars}(t)$. By Definition 12, $x \notin \mathrm{hvars}(\sigma)$. By case (15a) of Proposition 15, for all $\tau \in {\downarrow}\,\sigma$, we have $\mathrm{hvars}(\tau) \subseteq \mathrm{hvars}(\sigma)$. Thus $x \notin \mathrm{hvars}(\tau)$ and $(\neg x)\bigl(\mathrm{hval}(\tau)\bigr) = 1$. Therefore, by Definition 22, $\sigma \in \gamma_F(\neg x)$.

(23c). Let $\sigma \in \mathrm{RSubst}$ such that $x \in \mathrm{gvars}(\sigma) \cap \mathrm{hvars}(\sigma)$. By case (15b) of Proposition 15, we have $x \in \mathrm{hvars}(\tau)$ for all $\tau \in {\downarrow}\,\sigma$. So $(x)\bigl(\mathrm{hval}(\tau)\bigr) = 1$. Therefore, by Definition 22, $\sigma \in \gamma_F(x)$.

(23d). Let $\sigma_1 \in \Sigma_1$ and $\sigma_2 \in \Sigma_2$. Then, by hypothesis $\sigma_1 \in \gamma_F(\phi_1)$ and $\sigma_2 \in \gamma_F(\phi_2)$. Let $\tau \in \mathrm{mgs}(\sigma_1 \cup \sigma_2)$. By definition of mgs, $\mathcal{RT} \vdash \forall(\tau \rightarrow \sigma_1)$ and $\mathcal{RT} \vdash \forall(\tau \rightarrow \sigma_2)$. Thus, by Proposition 2, we have $\tau \in {\downarrow}\,\sigma_1 \cap {\downarrow}\,\sigma_2$. Therefore, by Proposition 69, $\tau \in \gamma_F(\phi_1) \cap \gamma_F(\phi_2)$. The result then follows by Proposition 70.

(23e). We have

$$
\begin{aligned}
\gamma_F(\phi_1 \vee \phi_2) &= \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\,\sigma : (\phi_1 \vee \phi_2)\bigl(\mathrm{hval}(\tau)\bigr) = 1 \,\Bigr\} \\
&= \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\,\sigma : \exists i \in \{1,2\} \,.\, \phi_i\bigl(\mathrm{hval}(\tau)\bigr) = 1 \,\Bigr\} \\
&\supseteq \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\,\sigma : \phi_1\bigl(\mathrm{hval}(\tau)\bigr) = 1 \,\Bigr\} \\
&\qquad \cup \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\,\sigma : \phi_2\bigl(\mathrm{hval}(\tau)\bigr) = 1 \,\Bigr\} \\
&= \gamma_F(\phi_1) \cup \gamma_F(\phi_2) \\
&\supseteq \Sigma_1 \cup \Sigma_2.
\end{aligned}
$$

(23f). Let $\sigma \in \Sigma$ and let $\sigma' \in \exists\, x \,.\, \{\sigma\}$. We will show that $\sigma' \in \gamma_F(\exists x \,.\, \phi)$.

Let $\tau' \in \downarrow \sigma'$. Then there exists $\sigma'_1 \in \mathrm{RSubst}$ such that $\mathcal{RT} \vdash \forall\bigl(\tau' \leftrightarrow (\sigma' \cup \sigma'_1)\bigr)$. Let $\sigma_1 \in \exists\, x \,.\, \{\sigma'_1\}$ and let $W \stackrel{\mathrm{def}}{=} (\mathrm{Vars} \setminus \mathrm{VI}) \cup \{x\}$. Then, by Definition 5, it follows $\mathcal{RT} \vdash \forall\bigl(\exists W \,.\, (\sigma' \leftrightarrow \sigma)\bigr)$ and $\mathcal{RT} \vdash \forall\bigl(\exists W \,.\, (\sigma'_1 \leftrightarrow \sigma_1)\bigr)$. As a consequence

$$\mathcal{RT} \vdash \forall\bigl(\exists W \,.\, (\sigma' \cup \sigma'_1) \leftrightarrow \exists W \,.\, (\sigma \cup \sigma_1)\bigr).$$

Therefore $\sigma \cup \sigma_1$ is satisfiable in $\mathcal{RT}$ so that, for some $\tau \in \mathrm{RSubst}$, $\mathcal{RT} \vdash \forall\bigl(\tau \leftrightarrow (\sigma \cup \sigma_1)\bigr)$. Thus $\mathcal{RT} \vdash \forall(\exists W \,.\, \tau \leftrightarrow \exists W \,.\, \tau')$. By Proposition 59, $\mathrm{hvars}(\tau') \setminus W = \mathrm{hvars}(\tau) \setminus W$ so that

$$\bigl(\mathrm{hvars}(\tau') \cap \mathrm{VI}\bigr) \cup \{x\} = (\mathrm{hvars}(\tau) \cap \mathrm{VI}) \cup \{x\}. \tag{B.73}$$

Let $c \stackrel{\mathrm{def}}{=} \mathrm{hval}(\tau)(x)$. Then, since $\tau \in \downarrow \sigma$ and, by hypothesis, $\sigma \in \gamma_F(\phi)$, we have the following chain of implications:

$$\begin{aligned}
\phi\bigl(\mathrm{hval}(\tau)\bigr) &= 1 && \text{[by Defn. 22]}\\
\phi\bigl(\mathrm{hval}(\tau)[c/x]\bigr) &= 1 && \text{[by Defn. 3]}\\
\phi\Bigl(\mathbf{0}\bigl[1/\,\mathrm{hvars}(\tau) \cap \mathrm{VI}\bigr][c/x]\Bigr) &= 1 && \text{[by Defn. 22]}\\
\phi\Bigl(\mathbf{0}\Bigl[1/\bigl(\mathrm{hvars}(\tau) \cap \mathrm{VI}\bigr) \cup \{x\}\Bigr][c/x]\Bigr) &= 1 && \text{[by Defn. 3]}\\
\phi\Bigl(\mathbf{0}\Bigl[1/\bigl(\mathrm{hvars}(\tau') \cap \mathrm{VI}\bigr) \cup \{x\}\Bigr][c/x]\Bigr) &= 1 && \text{[by (B.73)]}\\
\phi\Bigl(\mathbf{0}\bigl[1/\,\mathrm{hvars}(\tau') \cap \mathrm{VI}\bigr][c/x]\Bigr) &= 1 && \text{[by Defn. 3]}\\
\phi\bigl(\mathrm{hval}(\tau')[c/x]\bigr) &= 1 && \text{[by Defn. 22]}\\
\phi[c/x]\bigl(\mathrm{hval}(\tau')\bigr) &= 1. && \text{[by Defn. 4]}
\end{aligned}$$

From this last relation, since $\phi[c/x] \models \exists x \,.\, \phi$, it follows that

$$(\exists x \,.\, \phi)\bigl(\mathrm{hval}(\tau')\bigr) = 1.$$

As this holds for all $\tau' \in \downarrow \sigma'$, by Definition 22, $\sigma' \in \gamma_F(\exists x \,.\, \phi)$. $\square$

Proof of Theorem 25 on page 26. Since $h \subseteq h'$, by the monotonicity of γ_H we have $\gamma_H(h) \supseteq \gamma_H(h')$, whence one of the inclusions: $\gamma_H(h) \cap \gamma_F(\phi) \supseteq \gamma_H(h') \cap \gamma_F(\phi)$.

In order to establish the other inclusion, we now prove that $\sigma \in \gamma_H(h')$ assuming $\sigma \in \gamma_H(h) \cap \gamma_F(\phi)$. To this end, by Definition 16, it is sufficient to prove that $h' \subseteq \mathrm{hvars}(\sigma)$.

Let $z \in h'$ and let $\psi = \bigl(\phi \wedge \bigwedge h\bigr)$, so that, by hypothesis, $h' = \mathrm{true}(\psi)$. Therefore, we have $\psi \models z$. Consider now $\psi' = \bigl(\phi \wedge \bigwedge \mathrm{hvars}(\sigma)\bigr)$. Since $\sigma \in$

$\gamma_H(h)$, by Definition 16 we have $h \subseteq \mathrm{hvars}(\sigma)$, so that $\psi' \models \psi$ and thus $\psi' \models z$.

Since $\sigma \in \gamma_F(\phi)$, by Definition 22 we have $\phi\bigl(\mathrm{hval}(\sigma)\bigr) = 1$. Also note that $\bigl(\bigwedge \mathrm{hvars}(\sigma)\bigr)\bigl(\mathrm{hval}(\sigma)\bigr) = 1$. From these, by the definition of conjunction for Boolean formulas, we obtain $\psi'\bigl(\mathrm{hval}(\sigma)\bigr) = 1$. Thus we can observe that

$$\begin{aligned}\psi'\bigl(\mathrm{hval}(\sigma)\bigr) = 1 &\iff (\psi' \wedge z)\bigl(\mathrm{hval}(\sigma)\bigr) = 1\\ &\implies z \in \mathrm{hvars}(\sigma).\end{aligned}$$

□

Proof of Theorem 27 on page 27. Suppose there exists $\sigma \in \gamma_H(h) \cap \gamma_F(\phi)$. By Definition 22, since $\sigma \in \downarrow\sigma$, we have $\phi\bigl(\mathrm{hval}(\sigma)\bigr) = 1$; moreover, we have $\bigl(\bigwedge \mathrm{hvars}(\sigma)\bigr)\bigl(\mathrm{hval}(\sigma)\bigr) = 1$; therefore, by the definition of conjunction for Boolean formulas, we obtain

$$\Bigl(\phi \wedge \bigwedge h\Bigr)\bigl(\mathrm{hval}(\sigma)\bigr) = 1.$$

As a consequence, we also have

$$\mathrm{hvars}(\sigma) \cap \mathrm{false}\Bigl(\phi \wedge \bigwedge h\Bigr) = \varnothing;$$

by Definition 16, $h \subseteq \mathrm{hvars}(\sigma)$, so that we can conclude $h \cap \mathrm{false}\Bigl(\phi \wedge \bigwedge h\Bigr) = \varnothing$. □

B.8 Relation Between Groundness Dependencies and Finite-Tree Dependencies

As was the case for finite-tree dependencies, groundness dependencies only capture permanent information. Moreover, the γ_G function is *meet-preserving.*

Proposition 71 *Let $\sigma, \tau \in \mathrm{RSubst}$ and $\psi \in \mathrm{Pos}$, where we have $\sigma \in \gamma_G(\psi)$ and $\tau \in \downarrow\sigma$. Then $\tau \in \gamma_G(\psi)$.*

PROOF. By the hypothesis, $\tau \in \downarrow\sigma$, so that, for each $\upsilon \in \downarrow\tau$, $\upsilon \in \downarrow\sigma$. Therefore, as $\sigma \in \gamma_G(\psi)$, it follows from Definition 28 that, for all $\upsilon \in \downarrow\tau$, $\psi\bigl(\mathrm{gval}(\upsilon)\bigr) = 1$ and hence $\tau \in \gamma_G(\psi)$. □

Proposition 72 *Let $\psi_1, \psi_2 \in \mathrm{Pos}$. Then*

$$\gamma_G(\psi_1 \wedge \psi_2) = \gamma_G(\psi_1) \cap \gamma_G(\psi_2).$$

PROOF.

$$
\begin{aligned}
\gamma_G(\psi_1 \wedge \psi_2) &= \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\,\sigma : (\psi_1 \wedge \psi_2)\bigl(\mathrm{gval}(\tau)\bigr) = 1 \,\Bigr\} \\
&= \left\{\, \sigma \in \mathrm{RSubst} \;\middle|\; \begin{array}{c} \forall \tau \in {\downarrow}\,\sigma : \forall i \in \{1,2\} : \\ \psi_i\bigl(\mathrm{gval}(\tau)\bigr) = 1 \end{array} \right\} \\
&= \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\,\sigma : \psi_1\bigl(\mathrm{gval}(\tau)\bigr) = 1 \,\Bigr\} \\
&\qquad \cap \Bigl\{\, \sigma \in \mathrm{RSubst} \Bigm| \forall \tau \in {\downarrow}\,\sigma : \psi_2\bigl(\mathrm{gval}(\tau)\bigr) = 1 \,\Bigr\} \\
&= \gamma_G(\psi_1) \cap \gamma_G(\psi_2).
\end{aligned}
$$

□

Since non-ground terms can be made cyclic by instantiating their variables, those terms detected as definitely finite on Bfun are also definitely ground.

Proposition 73 *Let $x \in \mathrm{VI}$. Then $\gamma_F(x) \subseteq \gamma_G(x)$.*

PROOF. Suppose that $\sigma \in \gamma_F(x)$. Then, by Definition 22, $(x)\bigl(\mathrm{hval}(\tau)\bigr) = 1$ for all $\tau \in {\downarrow}\,\sigma$, so that $x \in \mathrm{hvars}(\tau)$; in particular, $x \in \mathrm{hvars}(\sigma)$. We prove $x \in \mathrm{gvars}(\sigma)$ by contradiction. That is, we show that if $x \in \mathrm{hvars}(\sigma) \setminus \mathrm{gvars}(\sigma)$, then there exists $\tau \in {\downarrow}\,\sigma$ for which $x \notin \mathrm{hvars}(\tau)$.

Suppose that $x \in \mathrm{hvars}(\sigma) \setminus \mathrm{gvars}(\sigma)$. Then, by Propositions 13 and 52, $\mathrm{rt}(x, \sigma) \in \mathrm{HTerms} \setminus \mathrm{GTerms}$. Hence, by Proposition 44, there exists $i \in \mathbb{N}$ such that $\mathrm{rt}(x, \sigma) = x\sigma^i$ and there exists $y \in \mathrm{vars}(x\sigma^i) \setminus \mathrm{dom}(\sigma)$. As we assumed that Sig contains a function symbol of non-zero arity, there exists $t \in \mathrm{HTerms} \setminus \{y\}$ for which $\{y\} = \mathrm{vars}(t)$. It follows that $\sigma' = \{y \mapsto t\} \in \mathrm{RSubst}$ and, by Definition 12, $y \notin \mathrm{hvars}(\sigma')$. Since $y \notin \mathrm{dom}(\sigma)$, by Lemma 39, $\tau = \sigma \cup \sigma' \in \mathrm{RSubst}$. Since $\tau \in {\downarrow}\,\sigma'$ then, by case (15a) of Proposition 15, we have $y \notin \mathrm{hvars}(\tau)$.

By Lemma 40, we have $\mathcal{RT} \vdash \forall\bigl(\sigma \to (x = x\sigma^i)\bigr)$. Thus, since we also have $\tau \in {\downarrow}\,\sigma$, we obtain $\mathcal{RT} \vdash \forall\bigl(\tau \to (x = x\sigma^i)\bigr)$. By applying Lemma 42, we have that $\mathrm{rt}(x, \tau) = \mathrm{rt}(x\sigma^i, \tau)$ and thus, by Proposition 13, we obtain $x \in \mathrm{hvars}(\tau)$ if and only if $\mathrm{vars}(x\sigma^i) \subseteq \mathrm{hvars}(\tau)$. However, as observed before, we know that $y \in \mathrm{vars}(x\sigma^i) \setminus \mathrm{hvars}(\tau)$, so that we also have $x \notin \mathrm{hvars}(\tau)$.

Therefore $x \in \mathrm{gvars}(\sigma) \cap \mathrm{hvars}(\sigma)$ and, by case (15b) of Proposition 15, for all $\tau \in {\downarrow}\,\sigma$, $x \in \mathrm{gvars}(\tau) \cap \mathrm{hvars}(\tau)$. As a consequence, for all $\tau \in {\downarrow}\,\sigma$, $(x)\bigl(\mathrm{gval}(\tau)\bigr) = 1$, so that, by Definition 28, we can conclude that $\sigma \in \gamma_G(x)$. □

Proof of Theorem 30 on page 28.

Proof of (30a). Since $\psi \wedge \psi' \models \psi$, the inclusion

$$\gamma_H(h) \cap \gamma_F(\phi) \cap \gamma_G(\psi) \supseteq \gamma_H(h) \cap \gamma_F(\phi) \cap \gamma_G(\psi \wedge \psi')$$

follows by the monotonicity of γ_G.

We now prove the reverse inclusion. Let us assume $\sigma \in \gamma_H(h) \cap \gamma_F(\phi) \cap \gamma_G(\psi)$. By Proposition 72 we have that $\gamma_G(\psi \wedge \psi') = \gamma_G(\psi) \cap \gamma_G(\psi')$. Therefore it is enough to show that $\sigma \in \gamma_G(\psi')$. By hypothesis, $\psi' = \mathrm{pos}(\exists \mathrm{VI} \setminus h \mathrel{.} \phi)$. Moreover, by Definition 22, $h \subseteq \mathrm{hvars}(\sigma)$. Thus, to prove the result, we will show, by contradiction, that $\sigma \in \gamma_G\Bigl(\mathrm{pos}\bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \mathrel{.} \phi\bigr)\Bigr)$.

Suppose therefore that $\sigma \notin \gamma_G\Bigl(\mathrm{pos}\bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \mathrel{.} \phi\bigr)\Bigr)$. Then there exists $\tau \in {\downarrow}\,\sigma$ such that

$$\mathrm{pos}\bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \mathrel{.} \phi\bigr)\bigl(\mathrm{gval}(\tau)\bigr) = 0. \tag{B.74}$$

Let $z \in \mathrm{hvars}(\sigma) \cap \mathrm{VI}$. By Proposition 13, $\mathrm{rt}(z, \sigma) \in \mathrm{HTerms}$. By Proposition 44, there exists $i \in \mathbb{N}$ such that $\mathrm{rt}(z, \sigma) = z\sigma^i$ and $\mathrm{vars}(z\sigma^i) \cap \mathrm{dom}(\sigma) = \varnothing$. Therefore, by Definition 12, $\mathrm{vars}(z\sigma^i) \subseteq \mathrm{hvars}(\sigma)$. Thus, we have

$$\mathrm{vars}(z\sigma^i) \subseteq \mathrm{hvars}(\sigma) \setminus \mathrm{dom}(\sigma). \tag{B.75}$$

By Lemma 40, as $\tau \in {\downarrow}\,\sigma$, $\mathcal{RT} \vdash \forall\bigl(\tau \to (z = z\sigma^i)\bigr)$. By Lemma 42, we have $\mathrm{rt}(z, \tau) = \mathrm{rt}(z\sigma^i, \tau)$ so that, by Proposition 52,

$$z \in \mathrm{gvars}(\tau) \iff \mathrm{vars}(z\sigma^i) \subseteq \mathrm{gvars}(\tau). \tag{B.76}$$

Take $t \in \mathrm{GTerms} \cap \mathrm{HTerms}$ and let

$$\upsilon_1 \stackrel{\mathrm{def}}{=} \Bigl\{\, y \mapsto t \Bigm| y \in \bigl(\mathrm{hvars}(\sigma) \cap \mathrm{gvars}(\tau)\bigr) \setminus \mathrm{dom}(\sigma) \,\Bigr\}.$$

As we assumed that Sig contains a function symbol of non-zero arity, for each $y \in \mathrm{Vars}$ there exists $t_y \in \mathrm{HTerms} \setminus \{y\}$ such that $\mathrm{vars}(t_y) = \{y\}$. Thus let

$$\upsilon_2 \stackrel{\mathrm{def}}{=} \left\{\, y \mapsto t_y \;\middle|\; \begin{array}{l} y \in \bigl(\mathrm{VI} \cup \mathrm{vars}(\sigma)\bigr) \cap \mathrm{hvars}(\sigma) \\ y \notin \mathrm{gvars}(\tau) \cup \mathrm{dom}(\sigma) \end{array} \right\}.$$

Note that $\upsilon_1, \upsilon_2 \in \mathrm{RSubst}$, $\mathrm{vars}(\upsilon_1) \cap \mathrm{vars}(\upsilon_2) = \varnothing$ and $\mathrm{vars}(\upsilon_i) \cap \mathrm{dom}(\sigma) = \varnothing$, for $i = 1, 2$. Thus, by Lemma 39, $\tau' \stackrel{\mathrm{def}}{=} (\sigma \cup \upsilon_1 \cup \upsilon_2) \in \mathrm{RSubst}$ is satisfiable in $\mathcal{RT}$.

We now show that

$$z \in \mathrm{gvars}(\tau) \iff z \in \mathrm{hvars}(\tau'). \tag{B.77}$$

Assume first that $z \in \mathrm{gvars}(\tau)$. Then, by (B.76), we have $\mathrm{vars}(z\sigma^i) \subseteq \mathrm{gvars}(\tau)$. From this, since also (B.75) holds, we obtain $\mathrm{vars}(z\sigma^i) \subseteq \mathrm{dom}(\upsilon_1)$ so that, by Definitions 9 and 12, $\mathrm{vars}(z\sigma^i) \subseteq \mathrm{gvars}(\upsilon_1) \cap \mathrm{hvars}(\upsilon_1)$. Since $\tau' \in {\downarrow}\, \upsilon_1$, by case (15b) of Proposition 15, $\mathrm{vars}(z\sigma^i) \subseteq \mathrm{gvars}(\tau') \cap \mathrm{hvars}(\tau')$. Thus, by Propositions 13 and 52, $\mathrm{rt}(z\sigma^i, \tau') \in \mathrm{GTerms} \cap \mathrm{HTerms}$. Now $\tau' \in {\downarrow}\, \sigma$ so that, by Lemma 40, $\mathcal{RT} \vdash \forall\bigl(\tau' \rightarrow (z = z\sigma^i)\bigr)$. By Lemma 42, $\mathrm{rt}(z\sigma^i, \tau') = \mathrm{rt}(z, \tau') \in \mathrm{GTerms} \cap \mathrm{HTerms}$ so that, by Proposition 13 and Proposition 52, $z \in \mathrm{hvars}(\tau')$.

We prove the other direction by contraposition, assuming that $z \notin \mathrm{gvars}(\tau)$. By (B.76), there exists $y \in \mathrm{vars}(z\sigma^i) \setminus \mathrm{gvars}(\tau)$. Also note that $y \in \mathrm{VI} \cup \mathrm{vars}(\sigma)$ and, by (B.75), $y \notin \mathrm{dom}(\sigma)$ so that $y \in \mathrm{dom}(\upsilon_2)$. By Definition 12, we have $y \notin \mathrm{hvars}(\upsilon_2)$ and, since $\tau' \in {\downarrow}\, \upsilon_2$, by case (15a) of Proposition 15, $y \notin \mathrm{hvars}(\tau')$. Thus, by Proposition 13, we have that $\mathrm{rt}(z\sigma^i, \tau') \notin \mathrm{HTerms}$. Moreover, as $\mathcal{RT} \vdash \forall\bigl(\tau' \rightarrow (z = z\sigma^i)\bigr)$, by Lemma 42 we have $\mathrm{rt}(z\sigma^i, \tau') = \mathrm{rt}(z, \tau') \notin \mathrm{HTerms}$ and therefore, by Proposition 13, $z \notin \mathrm{hvars}(\tau')$.

Since z was an arbitrary variable in $\mathrm{hvars}(\sigma) \cap \mathrm{VI}$, it follows from (B.74) and (B.77) that,

$$\mathrm{pos}\bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \;.\; \phi\bigr)\bigl(\mathrm{hval}(\tau')\bigr) = 0. \tag{B.78}$$

We have by hypothesis that $\sigma \in \gamma_F(\phi)$, so that, as $\tau' \in {\downarrow}\, \sigma$, by Definition 22 we have $\phi\bigl(\mathrm{hval}(\tau')\bigr) = 1$. Therefore, since $\phi \models \mathrm{pos}\bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \;.\; \phi\bigr)$, we obtain $\mathrm{pos}\bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \;.\; \phi\bigr)\bigl(\mathrm{hval}(\tau')\bigr) = 1$, which contradicts (B.78).

Proof of (30b). Since $\phi \wedge \phi' \models \phi$, the inclusion

$$\gamma_H(h) \cap \gamma_F(\phi) \cap \gamma_G(\psi) \supseteq \gamma_H(h) \cap \gamma_F(\phi \wedge \phi') \cap \gamma_G(\psi)$$

follows by the monotonicity of γ_F.

We now prove the reverse inclusion. Assume that $\sigma \in \gamma_H(h) \cap \gamma_F(\phi) \cap \gamma_G(\psi)$. By Proposition 70 we have that $\gamma_F(\phi \wedge \phi') = \gamma_F(\phi) \cap \gamma_F(\phi')$. Therefore it is enough to show that $\sigma \in \gamma_F(\phi')$. By hypothesis, $\phi' = \exists \mathrm{VI} \setminus h \;.\; \psi$. Moreover, by Definition 16, $h \subseteq \mathrm{hvars}(\sigma)$. Thus, to prove the result, we will show, by contradiction, that $\sigma \in \gamma_F\bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \;.\; \psi\bigr)$.

Suppose therefore that $\sigma \notin \gamma_F\bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \;.\; \psi\bigr)$. Then there exists $\tau \in {\downarrow}\, \sigma$ such that

$$\bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \;.\; \psi\bigr)\bigl(\mathrm{hval}(\tau)\bigr) = 0. \tag{B.79}$$

Take $t \in \mathrm{GTerms} \cap \mathrm{HTerms}$ and let

$$\upsilon \stackrel{\mathrm{def}}{=} \left\{ y \mapsto t \;\middle|\; y \in \mathrm{vars}(\sigma) \cap \Bigl(\mathrm{hvars}(\tau) \setminus \mathrm{dom}(\sigma)\Bigr) \right\}. \tag{B.80}$$

By Lemma 39, $\tau' \stackrel{\mathrm{def}}{=} \sigma \cup \upsilon \in \mathrm{RSubst}$ is satisfiable in $\mathcal{RT}$.

Let z be any variable in $\mathrm{hvars}(\sigma)$. By Proposition 13, $\mathrm{rt}(z, \sigma) \in \mathrm{HTerms}$. Then, by Proposition 44, there must exists $i \in \mathbb{N}$ such that $\mathrm{rt}(z, \sigma) = z\sigma^i$ and $\mathrm{vars}(z\sigma^i) \cap \mathrm{dom}(\sigma) = \varnothing$. Therefore, by Definition 12, $\mathrm{vars}(z\sigma^i) \subseteq \mathrm{hvars}(\sigma)$. Thus, we have

$$\mathrm{vars}(z\sigma^i) \subseteq \mathrm{hvars}(\sigma) \setminus \mathrm{dom}(\sigma). \tag{B.81}$$

By Lemma 40, as $\tau \in {\downarrow}\,\sigma$, $\mathcal{RT} \vdash \forall\bigl(\tau \rightarrow (z = z\sigma^i)\bigr)$. By Lemma 42, we have $\mathrm{rt}(z, \tau) = \mathrm{rt}(z\sigma^i, \tau)$ so that, by Proposition 13,

$$z \in \mathrm{hvars}(\tau) \iff \mathrm{vars}(z\sigma^i) \subseteq \mathrm{hvars}(\tau). \tag{B.82}$$

We now show that

$$\mathrm{hvars}(\tau) = \mathrm{hvars}(\sigma) \cap \mathrm{gvars}(\tau'). \tag{B.83}$$

Since $\tau \in {\downarrow}\,\sigma$, it follows from case (15a) of Proposition 15 that $\mathrm{hvars}(\tau) \subseteq \mathrm{hvars}(\sigma)$. Thus, as $z \in \mathrm{hvars}(\sigma)$, either $z \in \mathrm{hvars}(\tau)$ or $z \in \mathrm{hvars}(\sigma) \setminus \mathrm{hvars}(\tau)$. We consider these cases separately.

First, assume that $z \in \mathrm{hvars}(\tau)$. Then, by (B.82), $\mathrm{vars}(z\sigma^i) \subseteq \mathrm{hvars}(\tau)$. Also, by case (15a) of Proposition 15, we have $z \in \mathrm{hvars}(\sigma)$, so that we can apply (B.81) to derive $\mathrm{vars}(z\sigma^i) \cap \mathrm{dom}(\sigma) = \varnothing$. Therefore, $\mathrm{vars}(z\sigma^i) \subseteq \mathrm{dom}(\upsilon)$ and, by Definitions 9 and 12, $\mathrm{vars}(z\sigma^i) \subseteq \mathrm{gvars}(\upsilon) \cap \mathrm{hvars}(\upsilon)$. Since $\tau' \in {\downarrow}\,\upsilon$, by case (15b) of Proposition 15, we have $\mathrm{vars}(z\sigma^i) \subseteq \mathrm{gvars}(\tau') \cap \mathrm{hvars}(\tau')$. Thus, by Propositions 13 and 52, $\mathrm{rt}(z\sigma^i, \tau') \in \mathrm{GTerms} \cap \mathrm{HTerms}$. Now $\tau' \in {\downarrow}\,\sigma$ so that, by Lemma 40, we have $\mathcal{RT} \vdash \forall\bigl(\tau' \rightarrow (z = z\sigma^i)\bigr)$. Thus, by Lemma 42, $\mathrm{rt}(z\sigma^i, \tau') = \mathrm{rt}(z, \tau') \in \mathrm{GTerms} \cap \mathrm{HTerms}$ so that, by Propositions 13 and 52, $z \in \mathrm{hvars}(\tau') \cap \mathrm{gvars}(\tau')$. Hence, by case (15a) of Proposition 15, we can conclude $z \in \mathrm{hvars}(\sigma) \cap \mathrm{gvars}(\tau')$. Thus $\mathrm{hvars}(\tau) \subseteq \mathrm{hvars}(\sigma) \cap \mathrm{gvars}(\tau')$.

Secondly, assume that $z \in \mathrm{hvars}(\sigma) \setminus \mathrm{hvars}(\tau)$. Since $z \notin \mathrm{hvars}(\tau)$, by (B.82), there exists $y \in \mathrm{vars}(z\sigma^i) \setminus \mathrm{hvars}(\tau)$. Also, since $z \in \mathrm{hvars}(\sigma)$, by (B.81), we have $y \in \mathrm{hvars}(\sigma) \setminus \mathrm{dom}(\sigma)$ so that, by Definition 9, we have $y \notin \mathrm{gvars}(\sigma)$. By (B.80), since $y \notin \mathrm{dom}(\sigma) \cup \mathrm{hvars}(\tau)$, we have $y \notin \mathrm{dom}(\upsilon)$ so that $y \notin \mathrm{gvars}(\tau')$. Thus, by Proposition 52, we have $\mathrm{rt}(z\sigma^i, \tau') \notin \mathrm{GTerms}$. Moreover, since we have $\mathcal{RT} \vdash \forall\bigl(\tau' \rightarrow (z = z\sigma^i)\bigr)$, we obtain, by Lemma 42, $\mathrm{rt}(z\sigma^i, \tau') = \mathrm{rt}(z, \tau') \notin \mathrm{GTerms}$ and thus, by Proposition 52, $z \notin \mathrm{gvars}(\tau')$. Thus $\mathrm{hvars}(\tau) \supseteq \mathrm{hvars}(\sigma) \cap \mathrm{gvars}(\tau')$.

It follows from (B.79) and (B.83) that,

$$\Bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \mathrel{.} \psi\Bigr)\Bigl(\mathrm{gval}(\tau')\Bigr) = 0. \tag{B.84}$$

We have by hypothesis that $\sigma \in \gamma_G(\psi)$, so that, as $\tau' \in {\downarrow}\,\sigma$, by Definition 28 we have $\psi\Bigl(\mathrm{gval}(\tau')\Bigr) = 1$. Therefore, as $\psi \models \exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \mathrel{.} \psi$,

$$\Bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \mathrel{.} \psi\Bigr)\Bigl(\mathrm{gval}(\tau')\Bigr) = 1.$$

which contradicts (B.84). $\square$

Proof of Theorem 31 on page 28. Since $\psi \wedge \bigwedge \mathrm{true}(\phi) \models \psi$, the inclusion

$$\gamma_F(\phi) \cap \gamma_G(\psi) \supseteq \gamma_F(\phi) \cap \gamma_G\Bigl(\psi \wedge \bigwedge \mathrm{true}(\phi)\Bigr)$$

follows by the monotonicity of γ_G. To prove the inclusion

$$\gamma_F(\phi) \cap \gamma_G(\psi) \subseteq \gamma_F(\phi) \cap \gamma_G\Bigl(\psi \wedge \bigwedge \mathrm{true}(\phi)\Bigr)$$

we will show that $\gamma_F(\phi) \subseteq \gamma_G\Bigl(\bigwedge \mathrm{true}(\phi)\Bigr)$. The thesis will thus follow by Proposition 72. We have

$$\begin{aligned}
\gamma_F(\phi) &\subseteq \gamma_F\Bigl(\bigwedge \mathrm{true}(\phi)\Bigr) && [\text{since } \phi \models \textstyle\bigwedge \mathrm{true}(\phi)] \\
&= \bigcap \Bigl\{\, \gamma_F(x) \Bigm| x \in \mathrm{true}(\phi) \,\Bigr\} && [\text{by Proposition 70}] \\
&\subseteq \bigcap \Bigl\{\, \gamma_G(x) \Bigm| x \in \mathrm{true}(\phi) \,\Bigr\} && [\text{by Proposition 73}] \\
&= \gamma_G\Bigl(\bigwedge \mathrm{true}(\phi)\Bigr). && [\text{by Proposition 72}]
\end{aligned}$$

$\square$

Part of the proof of Theorem 34 relies on the following lemma.

Lemma 74 *Let $h \in H$ and $\phi \in \mathrm{Bfun}$ be such that $\gamma_H(h) \cap \gamma_F(\phi) \neq \varnothing$. Then $(\exists \mathrm{VI} \setminus h \mathrel{.} \phi) \in \mathrm{Pos}$.*

PROOF. By hypothesis, there exists $\sigma \in \gamma_H(h) \cap \gamma_F(\phi)$ so that, by Definitions 16 and 22, we have $h \subseteq \mathrm{hvars}(\sigma)$ and $\bigwedge \mathrm{hvars}(\sigma) \models \exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \mathrel{.} \phi$.

Towards a contradiction, suppose that $(\exists \mathrm{VI} \setminus h \mathrel{.} \phi) \notin \mathrm{Pos}$, i.e.,

$$(\exists \mathrm{VI} \setminus h \mathrel{.} \phi)(\mathbf{1}) = 0.$$

Since existential quantification is an extensive operator on Bfun and $h \subseteq \mathrm{hvars}(\sigma)$, we obtain $\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \,.\, \phi \models \exists \mathrm{VI} \setminus h \,.\, \phi$, so that

$$\Bigl(\exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \,.\, \phi\Bigr)(\mathbf{1}) = 0.$$

Moreover, since $\bigwedge \mathrm{hvars}(\sigma) \models \exists \mathrm{VI} \setminus \mathrm{hvars}(\sigma) \,.\, \phi$, we have

$$\Bigl(\bigwedge \mathrm{hvars}(\sigma)\Bigr)(\mathbf{1}) = 0.$$

which is a contradiction. Therefore, $(\exists \mathrm{VI} \setminus h \,.\, \phi) \in \mathrm{Pos}$ $\square$

Proof of Theorem 34 on page 31. Let us assume the hypotheses and prove each statement in turn.

Consider first the case where $i = 1$, which corresponds to the application of the abstract disjunction operator. Then, for the finiteness component h_1 we have:

$$\begin{aligned}
h_1 &= h \cap h' \\
&\supseteq \mathrm{true}\Bigl(\phi \wedge \bigwedge h\Bigr) \cap \mathrm{true}\Bigl(\phi' \wedge \bigwedge h'\Bigr) \\
&\supseteq \mathrm{true}\Bigl(\phi \wedge \bigwedge (h \cap h')\Bigr) \cap \mathrm{true}\Bigl(\phi' \wedge \bigwedge (h \cap h')\Bigr) \\
&= \mathrm{true}\Bigl(\phi \wedge \bigwedge (h \cap h') \vee \phi' \wedge \bigwedge (h \cap h')\Bigr) \\
&= \mathrm{true}\Bigl((\phi \vee \phi') \wedge \bigwedge (h \cap h')\Bigr) \\
&= \mathrm{true}\Bigl(\phi_1 \wedge \bigwedge h_1\Bigr).
\end{aligned}$$

For the finite-tree dependencies component ϕ_1, we have:

$$\begin{aligned}
\phi_1 &= \phi \vee \phi' \\
&\models (\exists \mathrm{VI} \setminus h \,.\, \psi) \vee (\exists \mathrm{VI} \setminus h' \,.\, \psi') \\
&\models \Bigl(\exists \mathrm{VI} \setminus (h \cap h') \,.\, \psi\Bigr) \vee \Bigl(\exists \mathrm{VI} \setminus (h \cap h') \,.\, \psi'\Bigr) \\
&= \exists \mathrm{VI} \setminus (h \cap h') \,.\, \psi \vee \psi' \\
&= \exists \mathrm{VI} \setminus h_1 \,.\, \psi_1.
\end{aligned}$$

For the groundness dependencies component ψ_1 we have:

$$\begin{aligned}
\psi_1 &= \psi \vee \psi' \\
&\models \mathrm{pos}(\exists \mathrm{VI} \setminus h \,.\, \phi) \vee \mathrm{pos}(\exists \mathrm{VI} \setminus h' \,.\, \phi') \\
&= \Bigl((\exists \mathrm{VI} \setminus h \,.\, \phi) \vee \bigwedge \mathrm{VI}\Bigr) \vee \Bigl((\exists \mathrm{VI} \setminus h' \,.\, \phi') \vee \bigwedge \mathrm{VI}\Bigr)
\end{aligned}$$

$$
\begin{aligned}
&= (\exists \mathrm{VI} \setminus h \,.\, \phi) \vee (\exists \mathrm{VI} \setminus h' \,.\, \phi') \vee \bigwedge \mathrm{VI} \\
&= \mathrm{pos}\Bigl((\exists \mathrm{VI} \setminus h \,.\, \phi) \vee (\exists \mathrm{VI} \setminus h' \,.\, \phi')\Bigr) \\
&\models \mathrm{pos}\Biggl(\Bigl(\exists \mathrm{VI} \setminus (h \cap h') \,.\, \phi\Bigr) \vee \Bigl(\exists \mathrm{VI} \setminus (h \cap h') \,.\, \phi'\Bigr)\Biggr) \\
&= \mathrm{pos}\Bigl(\exists \mathrm{VI} \setminus (h \cap h') \,.\, \phi \vee \phi'\Bigr) \\
&= \mathrm{pos}(\exists \mathrm{VI} \setminus h_1 \,.\, \phi_1).
\end{aligned}
$$

Consider now the case where $i = 2$, which corresponds to the application of the abstract projection operator. Then, for the finiteness component h_2 we have:

$$
\begin{aligned}
h_2 &= h \cup \{x\} \\
&\supseteq \mathrm{true}\Bigl(\phi \wedge \bigwedge h\Bigr) \cup \{x\} \\
&\supseteq \mathrm{true}\Bigl((\exists x \,.\, \phi) \wedge \bigwedge h\Bigr) \cup \{x\} \\
&= \mathrm{true}\Bigl((\exists x \,.\, \phi) \wedge \bigwedge\bigl(h \cup \{x\}\bigr)\Bigr) \\
&= \mathrm{true}\Bigl(\phi_2 \wedge \bigwedge h_2\Bigr).
\end{aligned}
$$

For the finite-tree dependencies component ϕ_2 we have:

$$
\begin{aligned}
\phi_2 &= \exists x \,.\, \phi \\
&\models \exists x \,.\, \exists \mathrm{VI} \setminus h \,.\, \psi \\
&= \exists \mathrm{VI} \setminus h \,.\, \exists x \,.\, \psi \\
&= \exists \mathrm{VI} \setminus \bigl(h \cup \{x\}\bigr) \,.\, \exists x \,.\, \psi \\
&= \exists \mathrm{VI} \setminus h_2 \,.\, \psi_2.
\end{aligned}
$$

By hypothesis, $\gamma_H(h) \cap \gamma_F(\phi) \neq \varnothing$ so that we also have $\gamma_H\bigl(h \cup \{x\}\bigr) \cap \gamma_F(\exists x \,.\, \phi) \neq \varnothing$. Thus, for the groundness dependencies component ψ_2 we have:

$$
\begin{aligned}
\psi_2 &= \exists x \,.\, \psi \\
&\models \exists x \,.\, \mathrm{pos}(\exists \mathrm{VI} \setminus h \,.\, \phi) \\
&= \exists x \,.\, \exists \mathrm{VI} \setminus h \,.\, \phi && \text{[by Lemma 74]} \\
&= \exists \mathrm{VI} \setminus h \,.\, \exists x \,.\, \phi \\
&= \exists \mathrm{VI} \setminus \bigl(h \cup \{x\}\bigr) \,.\, \exists x \,.\, \phi \\
&= \mathrm{pos}\Bigl(\exists \mathrm{VI} \setminus \bigl(h \cup \{x\}\bigr) \,.\, \exists x \,.\, \phi\Bigr) && \text{[by Lemma 74]} \\
&= \mathrm{pos}(\exists \mathrm{VI} \setminus h_2 \,.\, \phi_2).
\end{aligned}
$$

□