On the Theory of Structural Subtyping

Viktor Kuncak and Martin Rinard
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{vkuncak, rinard}@lcs.mit.edu

MIT-LCS-TR-879, January 2003

Abstract

We show that the first-order theory of structural subtyping of non-recursive types is decidable.

Let Σ be a language consisting of function symbols (representing type constructors) and C a decidable structure in the relational language L containing a binary relation $\leq$. C represents primitive types; $\leq$ represents a subtype ordering. We introduce the notion of Σ-*term-power* of C, which generalizes the structure arising in structural subtyping. The domain of the Σ-term-power of C is the set of Σ-terms over the set of elements of C.

We show that the decidability of the first-order theory of C implies the decidability of the first-order theory of the Σ-term-power of C. This result implies the decidability of the first-order theory of structural subtyping of non-recursive types.

Our decision procedure is based on quantifier elimination and makes use of quantifier elimination for term algebras and Feferman-Vaught construction for products of decidable structures.

We also explore connections between the theory of structural subtyping of recursive types and monadic second-order theory of tree-like structures. In particular, we give an embedding of the monadic second-order theory of infinite binary tree into the first-order theory of structural subtyping of recursive types.

Keywords: Structural Subtyping, Quantifier Elimination, Term Algebra, Decision Problem, Monadic Second-Order Logic

*This research was supported in part by DARPA Contract F33615-00-C-1692, NSF Grant CCR00-86154, NSF Grant CCR00-63513, and the Singapore-MIT Alliance.

†Draft of January 25, 2003, 10:54am, see http://www.mit.edu/~vkuncak/papers for later versions.

Contents

1 Introduction

Subtyping constraints are an important technique for checking and inferring program properties, used both in type systems and program analyses [34, 16, 13, 28, 23, 4, 3, 1, 2, 20, 41, 17, 54, 7, 8, 5, 42, 47, 19].

This paper presents a decision procedure for the first-order theory of structural subtyping of non-recursive types. This result solves (for the case of non-recursive types) a problem left open in [48]. [48] provides the decidability result for structural subtyping of only unary type constructors, whereas we solve the problem for any number of constructors of any arity. Furthermore, we do not impose any constraints on the subtyping relation $\leq$, it need not even be a partial order. The generality of our construction makes it potentially of independent interest in logic and model theory.

We approach the problem of structural subtyping using quantifier elimination and, to some extent, using monadic second-order logic of tree-like structures. This paper makes the contributions:

- we give a new presentation of Feferman-Vaught theorem for direct products using a multisorted logic (Section 3.3); for completeness we also include proof of quantifier-elimination for boolean algebras of sets (Section 3.2);
- we give a new presentation of decidability of the first-order theory of term algebras; the proof uses the language of both constructor and selector symbols (Section 3.4);
- as an introduction to main result, we show decidability of structural subtyping with one covariant binary constructor and two constants (Section 4), this result does not rely on Feferman-Vaught technique;
- we present a new construction, *term-power algebra* for creating tree-like theories based on existing theories (Section 5);
- as a central result, we prove that if the base theory is decidable, so is the theory of term-power with arbitrary variance of constructors; we give an effective decision procedure for quantifier elimination in term-power structure; the procedure combines elements of quantifier elimination in Feferman-Vaught theorem and quantifier elimination in term algebras (Sections 5, 6).
- we show the decidability of structural subtyping non-recursive types as a direct consequence of the main result;
- we give a simple embedding of monadic second-order theory of infinite binary tree into the theory of structural subtyping of recursive types with two primitive types (Section 7.1);
- we show that structural subtyping of recursive types where terms range over constant shapes is decidable (Section 7.4);

In addition to showing the decidability of structural subtyping, our hope is to promote the important technique of quantifier elimination, which forms the basis of our result.

Quantifier elimination [22, Section 2.7] is a fruitful technique that was used to show decidability and classification of boolean algebras [46, 51] decidability of term algebras [31, Chapter 23], [39, 30], with membership constraints [10] and with queues [43], decidability of products [35, 14], [31, Chapter 12], and algebraically closed fields [50],

The complexity of the decision problem for the first-order theory of structural subtyping has a non-elementary lower bound. This is a consequence of a general theorem about pairing functions [15, Theorem 1.2, Page 163] and applies to term algebras already, as observed in [39, 43].

2 Preliminaries

In this section we review some notions used in the this paper.

If w is a word over some alphabet, we write $|w|$ for the length of w. We write $w_1 \cdot w_2$ to denote the concatenation of words w_1 and w_2.

A node v in a directed graph is a *sink* if v has no outgoing edges. A node v in a directed graph is a *source* if v has no incoming edges.

We write $E_1 \equiv E_2$ to denote equality of syntactic entities E_1 and E_2.

We write $\bar{x}$ to denote some sequence of variables $x_1, \ldots, x_n$.

We assume that formulas are built from propositional connectives $\wedge$, $\vee$, $\neg$, the remaining connectives are defined as shorthands. Connective $\neg$ binds the strongest, followed by $\wedge$ and $\vee$.

A literal L is an atomic formula A or a negation of an atomic formula $\neg A$. We define complementation of a literal by $\overline{A} = \neg A$ and $\overline{\neg A} = A$.

A formula ψ is in prenex form if it is of the form

$$Q_1 x_1. \ldots Q_n x_n.\phi$$

where $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq n$ and ϕ is a quantifier free formula. We call ϕ a *matrix* of ψ.

If ϕ is a formula then $\mathsf{FV}(\phi)$ denotes the set of free variables in ϕ.

We write $[x_1 \mapsto a_1, \ldots, x_k \mapsto a_k]$ for the substitution σ such that $\sigma(x_i) = a_i$ for $1 \leq i \leq k$.

If ϕ is a formula and $t_1, \ldots, t_k$ terms, we write $\phi[x_1 := t_1, \ldots, x_k := t_k]$ for the result of simultaneously substituting free occurrences of variables x_i with term t_i, for $1 \leq i \leq k$.

We write $\mathsf{h}(t)$ for the height of term t. $\mathsf{h}(a) = 0$ if a is a constant, $\mathsf{h}(x) = 0$ if x is a variable. If $f(t_1, \ldots, t_k)$ is a term then

$$\mathsf{h}(f(t_1, \ldots, t_k)) = 1 + \max(\mathsf{h}(t_1), \ldots, \mathsf{h}(t_k))$$

We assume that all function symbols are of finite arity. If there are finitely many function symbols then for any non-negative integer k there is only a finite number of terms t such that $\mathsf{h}(t) \leq k$.

If $\phi(u)$ is a conjunction of literals, we say that ϕ' results from $\exists u.\phi(u)$ by *dropping quantified variable* u iff ϕ' is the result of eliminating from $\phi(u)$ all conjunctions containing u. More generally, if ψ is a formula of form

$$Q_1 x_1 \ldots Q u \ldots Q_k x_k.\ \psi_0$$

then the result of dropping u from ψ is

$$Q_1 x_1 \ldots Q_k x_k.\ \psi_0'$$

where ψ_0' is the result of dropping u from $\exists u.\phi_0$.

An *equality* is an atomic formula $t_1 = t_2$ where t_1 and t_2 are terms. A *disequality* is negation of an equality.

We use the usual Tarskian semantics of formulas. Unless otherwise stated $\phi \models \psi$ will denote that formula $\phi \Rightarrow \psi$ is true in a fixed relational structure that is under current consideration.

Occasionally we find it convenient to work with multi-sorted logic, where domain is union of disjoint sets called sorts, and arity specifies the sorts of all operations. Constants are operations with zero arguments. Relations are operations that return the result in a distinguished sort bool interpreted over the boolean lattice $\{\mathsf{false}, \mathsf{true}\}$ or over the distributive lattice of three-valued logic $\{\mathsf{false}, \mathsf{true}, \mathsf{undef}\}$ from Section 2.3).

A structure $\mathcal{C}$ of a given language L is a pair of domain C and the interpretation function $[\![\_]\!]^C$. Hence, we name operations of the structure using symbols of the language and the interpretation function. If C is clear from the context we write simply $[\![\_]\!]$ for $[\![\_]\!]^C$.

In Section 3.3 and Section 6 we use logic with several kinds of quantifiers. Our logic is first-order, but we give higher-order types to quantifiers. For example, a quantifier

$$Q :: (A \to B) \to B$$

denotes a quantifier that binds variables of A sort enclosed within an expression of B sort and returns an expression of B sort. If X and Y are sets then $X \to Y$ denotes the set of all functions from A to B. When specifying the semantics of the quantifier Q we specify a function

$$[\![Q]\!] : ([\![A]\!] \to [\![B]\!]) \to [\![B]\!]$$

The semantics of an expression M of sort B takes an environment σ which is a function from variable names to elements of A and produces an element of B, hence $[\![M]\!]\sigma \in [\![B]\!]$. We define the semantics of an expression $Qx.\ M$ by:

$$[\![Qx.\ M]\!]\sigma = [\![Q]\!]h$$

where $h : [\![A]\!] \to [\![B]\!]$ is the function

$$h(a) = [\![M]\!](\sigma[x := a])$$

Here

$$\sigma[x := a](y) = \begin{cases} \sigma(y), & \text{if } y \not\equiv x \\ a, & \text{if } y \equiv x \end{cases}$$

Specifying types for quantifiers allows to express more

Let σ_A be some arbitrary dummy global environment. If F is a formula without global variables we write $[\![F]\!]\sigma_A$ to denote the truth value of F; clearly $[\![F]\!]\sigma_A$ does not depend on σ_A and we denote it simply $[\![F]\!]$ when no ambiguity arises.

We use Hilbert's epsilon as a notational convenience in metatheory. If $P(x)$ is a unary predicate, then $\varepsilon x.P(x)$ denotes an arbitrary element d such that $P(d)$ holds, if such element exists, or an arbitrary object otherwise.

2.1 Term Algebra

We introduce the notion of term algebra [22, Page 14].

Let Nat be the set of natural numbers. Let the signature Σ be a finite set of function symbols and constants and let $\mathsf{ar} : \Sigma \to \mathsf{Nat}$ be a function specifying arity $\mathsf{ar}(f)$ for every function symbol or constant $f \in \Sigma$. Let $\mathsf{FT}(\Sigma)$ denote the set of finite ground terms over signature Σ. We assume that Σ contains at least one constant $c \in \Sigma$, $\mathsf{ar}(c) = 0$, and at least one function symbol $f \in \Sigma$, $\mathsf{ar}(f) > 0$. Therefore, $\mathsf{FT}(\Sigma)$ is countably infinite.

Let $\mathsf{Cons}(\Sigma)$ be the term algebra interpretation of signature Σ, defined as follows [22, Page 14]. For every $f \in \Sigma$ with $\mathsf{ar}(f) = k$ define $[\![f]\!] \in \mathsf{Cons}(\Sigma)$, with $[\![f]\!] : \mathsf{FT}(\Sigma)^k \to \mathsf{FT}(\Sigma)$ by

$$[\![f]\!](t_1, \ldots, t_k) = f(t_1, \ldots, t_k)$$

We will write f instead of $[\![f]\!]$ when it causes no confusion.

2.2 Terms as Trees

We define trees representing terms as follows.

We use sequences of nonegative integers to denote paths in the tree. Let Σ be a signature. A tree over Σ is a partial function t from the set Nat^* of paths to the set Σ of function symbols such that:

1. if $w \in \mathsf{Nat}^*$, $x \in \mathsf{Nat}$, and $t(w \cdot x)$ is defined, then $t(w)$ is defined as well;
2. if $t(w) = f$ with $\mathsf{ar}(f) = k$, then
$$\{i \mid t(w \cdot i) \text{ is defined }\} = \{1, \ldots, k\}$$

A *finite tree* is a tree with a finite domain.

2.3 First Order Structures with Partial Functions

We make use of partial functions in our quantifier elimination procedures. In this section we briefly describe the approach to partial functions we chose to use; other approaches would work as well, see e.g. [24].

A language of partial functions Σ_1 contains partial function symbols in addition to total function symbols and relation symbols. Consider a structure with the domain A interpreting a language with partial function symbols Σ_1. Given some environment σ, we have $[\![t]\!]\sigma \in A \cup \{\bot\}$ where $\bot \notin A$ is a special value denoting undefined results. We require the interpretations of total and partial function symbols to be strict in $\bot$, i.e. $f(a_1, \ldots, a_i, \bot, a_{i+2}, \ldots, a_k) = \bot$.

We interpret atomic formulas and their negations over the three-valued domain $\{\mathsf{false}, \mathsf{true}, \mathsf{undef}\}$ using strong Kleene's three-valued logic [26, 24, 44]. We require that $[\![R]\!](a_1, \ldots, a_i, \bot, a_{i+2}, \ldots, a_k) = \mathsf{undef}$ for every relational symbol R. Logical connectives in Kleene's strong three-valued logic are the strongest "regular" extension of the corresponding connectives on the two-valued domain [26]. The regularity requirement means that the three-valued logic is a sound approximation of two-valued logic in the following sense. We may obtain the truth tables for three-valued logic by considering the truth values $\mathsf{false}, \mathsf{true}, \mathsf{undef}$ as shorthands for sets $\{\mathsf{false}\}, \{\mathsf{true}\}, \{\mathsf{false}, \mathsf{true}\}$ and defining each logical operation $*$ by:

$$s_1\ [\![*]\!]\ s_2 = \{b_1 \circ b_2 \mid b_1 \in s_1 \land b_2 \in s_2\}$$

where $\circ$ denotes the corresponding operation in the two-valued logic. As in a call-by-value semantics of lambda calculus, variables in the environments (σ) do not range over $\bot$. We interpret quantifiers as ranging over the domain A or its subset if the logic is multisorted; the interpretation of quantifiers are similarly the best regular approximations of the corresponding two-valued interpretations.

These properties of Kleene's three-valued logic have the following important consequence. Suppose that we extend the definition of all partial functions to make them total functions on the domain A by assigning arbitrary values outside the original domain. Suppose that a formula ϕ evaluates to an element of $b \in \{\mathsf{false}, \mathsf{true}\}$ in Kleene's logic. Then ϕ evaluates to the *same* truth-value b in the new logic of total functions. This property of three-valued logic implies that the algorithms that we use to transform formulas with partial functions will apply even for the logic that makes all functions total by completing them with arbitrary elements of A.

We say that a formula ψ is *well-defined* iff its truth value is an element of $\{\mathsf{false}, \mathsf{true}\}$.

Example 1 Consider the domain of real numbers. The following formulas are not well-defined:

$$3 = 1/0$$
$$\forall x.\ 1/x > 0 \ \vee\ 1/x < 0 \ \vee\ 1/x = 0$$

The following formulas are well-defined:

$$\exists x.\ 1/x = 3$$
$$\forall x.\ 1/x \neq 3$$
$$x = 0 \ \vee\ 1/x > 0$$

♦

We say that a formula ϕ_1 is *equivalent* to a formula ϕ_2 and write $\phi_1 \cong \phi_2$ iff

$$[\![\phi_1]\!]\sigma = [\![\phi_2]\!]\sigma$$

for all valuations σ (including those for which $[\![\phi_1]\!]\sigma = \mathsf{undef}$).

Sections below perform equivalence-preserving transformations of formulas. This means that starting from a well-defined formula we obtain an equivalent well-defined formula.

When doing equivalence preserving transformations it is useful to observe that $\wedge, \vee$ still form a distributive lattice. The partial order of this lattice is the chain $\mathsf{false} \leq \mathsf{undef} \leq \mathsf{true}$. The element undef does not have a complement in the lattice; unary operation $\neg$ does not denote the lattice complement. However, the following laws still hold:

$$\neg(x \wedge y) \cong \neg x \vee \neg y$$
$$\neg(x \vee y) \cong \neg x \wedge \neg y$$
$$\neg\neg x \cong x$$

The properties of $\wedge, \vee, \neg$ are sufficient to transform any quantifier-free formula into disjunction of conjunctions of literals using the well-known straightforward technique. However, this straightforward technique in some cases yields conjunctions that are not well-defined, even though the formula as a whole is well-defined.

Example 2 Transforming a negation of well-defined formula:

$$\neg(x \neq 0 \wedge (y = 1/x \vee z = x + 1))$$

may yield the following disjunction of conjunctions:

$$x = 0 \vee (y \neq 1/x \wedge z \neq x + 1)$$

where $y \neq 1/x \wedge z \neq x + 1$ is not a well-defined conjunction for $x = 0$.

♦

To enable the transformation of each well-defined formula into a disjunction of well-defined conjunctions of literals, we enrich the language of function and relation symbols as follows. With each partial function symbol $f \in \Sigma_1$ of arity $k = \mathsf{ar}(f)$ we associate a *domain description* $D_f = \langle\langle x_1, \ldots, x_k\rangle, \phi\rangle$ specifying the domain of f. Here $x_1, \ldots, x_k$ are distinct variables and ϕ is an unnested conjunction of literals such that $\mathsf{FV}(\phi) \subseteq \{x_1, \ldots, x_k\}$. We require every interpretation of a first-order structure with partial function symbols to satisfy the following property:

$$[\![f]\!](a_1, \ldots, a_k) \neq \bot \iff [\![\phi]\!][x_1 \mapsto a_1, \ldots, x_k \mapsto a_k]$$

for all $a_1, \ldots, a_k \in A$. We henceforth assume that every structure with partial functions is equipped with a domain description D_f for every partial function symbol f.

The Proposition 8 below gives an algorithm for transforming a given well-defined formula into a disjunction of well-defined conjunctions. We first give some definitions and lemmas.

Definition 3 *If ψ is a formula with free variables, a* domain formula *for ψ is a formula ϕ not containing partial function symbols such that, for every valuation σ,*

$$[\![\psi]\!]\sigma \neq \mathsf{undef} \iff [\![\phi]\!]\sigma = \mathsf{true}$$

From Definition 3 we obtain the following Lemma 4.

Lemma 4 *Let ψ be a formula and ϕ a domain formula for ψ. Then*

$$\psi \cong (\psi \wedge \phi) \vee (\mathsf{undef} \wedge \neg\phi)$$

Proof. Let σ be arbitrary valuation. Let $v = [\![\psi]\!]\sigma$. If $v \in \{\mathsf{true}, \mathsf{false}\}$ then $[\![\phi]\!]\sigma = \mathsf{true}$ and

$$[\![(\psi \wedge \phi) \vee (\mathsf{undef} \wedge \neg\phi)]\!]\sigma =$$
$$(v \wedge \mathsf{true}) \vee (\mathsf{undef} \wedge \mathsf{false}) = v.$$

If $v = \mathsf{undef}$ then $[\![\phi]\!] = \mathsf{false}$, so

$$[\![(\psi \wedge \phi) \vee (\mathsf{undef} \wedge \neg\phi)]\!]\sigma =$$
$$(\mathsf{undef} \wedge \mathsf{false}) \vee (\mathsf{undef} \wedge \mathsf{true}) = \mathsf{undef}.$$

■

Observe that $\psi \wedge \phi$ in Lemma 4 is a well-defined conjunction. We use this property to construct domain formulas using partial function domain descriptions.

Let

$$D_f = \langle\langle x_1, \ldots, x_k\rangle, B_1^f \wedge \ldots \wedge B_{l^f}^f\rangle$$

for each partial function symbol $f \in \Sigma_1$ of arity k, where $B_1^f, \ldots, B_{l^f}^f$ are unnested literals. If $t_1, \ldots, t_k$ are terms, we write $B_i^f(t_1, \ldots, t_k)$ for $B_i^f[x_1 := t_1, \ldots, x_k := t_k]$. Let $\mathsf{subt}(t)$ denote the set of all subterms of term t.

For any literal $B(t_1, \ldots, t_n)$ where $B(t_1, \ldots, t_n) \equiv R(t_1, \ldots, t_n)$ or $B(t_1, \ldots, t_n) \equiv \neg R(t_1, \ldots, t_n)$, define

$$\mathsf{DomForm}(B(t_1, \ldots, t_n)) = \bigwedge_{\substack{f(s_1, \ldots, s_k) \in \cup_{1 \leq i \leq n} \mathsf{subt}(t_i) \\ 1 \leq j \leq l^f}} B_j^f(s_1, \ldots, s_k) \tag{1}$$

Lemma 5 *Let $B(t_1, \ldots, t_n)$ be a literal containing partial function symbols. Then* $\mathsf{DomForm}(B(t_1, \ldots, t_n))$ *is a domain formula for $B(t_1, \ldots, t_n)$.*

Proof. Let σ be a valuation. By strictness of interpretations of function and predicate symbols, $[\![B(t_1, \ldots, t_n)]\!]\sigma \neq \mathsf{undef}$ iff $[\![f(s_1, \ldots, s_k)]\!]\sigma \neq \perp$ for every subterm $f(s_1, \ldots, s_k)$ of every term t_i, iff $[\![B_j^f(s_1, \ldots, s_k)]\!]\sigma = \mathsf{true}$ for every $1 \leq j \leq l^f$ and every subterm $f(s_1, \ldots, s_k)$. ∎

Lemma 6 *Let B be a literal and let*

$$\mathsf{DomForm}(B) = F_1 \wedge \ldots \wedge F_m.$$

Then

$$B \cong (B \wedge F_1 \wedge \ldots \wedge F_m) \vee \bigvee_{1 \leq i \leq m} (\mathsf{undef} \wedge \neg F_i \wedge \mathsf{DomForm}(F_i))$$

Proof. If $[\![B]\!]\sigma \neq \mathsf{undef}$, then $[\![F_i]\!]\sigma = \mathsf{true}$ for every $1 \leq i \leq m$, and

$$[\![\mathsf{undef} \wedge \neg F_i \wedge \mathsf{DomForm}(F_i)]\!]\sigma = \mathsf{false}$$

so the right-hand side evaluates to $[\![B]\!]\sigma$ as well. Now consider the case when $[\![B]\!]\sigma = \mathsf{undef}$. Then there exists a term $f(s_1, \ldots, s_k)$ such that $[\![f(s_1, \ldots, s_k)]\!]\sigma = \mathsf{undef}$. Because $\sigma(x) \neq \perp$ for every variable x, there exists a term $f(s_1, \ldots, s_k)$ such that $[\![f(s_1, \ldots, s_k)]\!]\sigma = \mathsf{undef}$ and $[\![s_i]\!]\sigma \neq \mathsf{undef}$ for $1 \leq i \leq k$. Then there exists a formula F_p of form $B_j^f(s_1, \ldots, s_k)$ such that $[\![B_j^f(s_1, \ldots, s_k)]\!]\sigma = \mathsf{false}$, and

$$[\![\mathsf{undef} \wedge \neg F_p \wedge \mathsf{DomForm}(F_p)]\!]\sigma = \mathsf{undef}.$$

Because

$$[\![B \wedge F_1 \wedge \ldots \wedge F_m]\!]\sigma = \mathsf{false},$$

and for every q,

$$[\![\mathsf{undef} \wedge \neg F_q \wedge \mathsf{DomForm}(F_q)]\!]\sigma \in \{\mathsf{undef}, \mathsf{false}\},$$

the right-hand side evaluates to undef. ∎

Lemma 7 *Let $\phi_0(\bar{y})$ and $\phi_1(\bar{y})$ be well-defined formulas whose free variables are among $\bar{y}$ and let*

$$\psi(\bar{y}) \equiv (\mathsf{undef} \wedge \phi_0(\bar{y})) \vee \phi_1(\bar{y})$$

If $\psi(\bar{y})$ is well-defined for all values of variables $\bar{y}$, then

$$\psi(\bar{y}) \cong \phi_1(\bar{y})$$

Proof. Consider any valuation σ. Let

$$v = [\![\phi_1(\bar{y})]\!]\sigma$$

and

$$v' = [\![\psi(\bar{y})]\!]\sigma$$

We need to show $v = v'$. Because $\phi(\bar{y})$ and $\psi(\bar{y})$ are well-defined, $v, v' \in \{\mathsf{false}, \mathsf{true}\}$. We consider two cases.
Case 1. $v = \mathsf{true}$. Then also $v' = \mathsf{true}$.
Case 2. $v = \mathsf{false}$. Then $v' = \mathsf{undef} \wedge \phi_0(\bar{y})$. Because $v' \neq \mathsf{undef}$, we conclude $v' = \mathsf{false}$. ∎

Proposition 8 *Every well-defined quantifier-free formula ψ can be transformed into an equivalent disjunction ψ' of well-defined conjunctions of literals.*

Proof. Using the standard procedure, convert ψ to disjunction of conjunctions

$$C_1 \vee \ldots \vee C_n$$

Let $C_i = B \wedge C_i'$ where B is a literal and let $\mathsf{DomForm}(B) = F_1 \wedge \ldots \wedge F_m$. Replace $B \wedge C_i'$ by

$$(B \wedge F_1 \wedge \ldots \wedge F_m \wedge C_i') \vee \bigvee_{1 \leq i \leq m} (\mathsf{undef} \wedge \neg F_i \wedge \mathsf{DomForm}(F_i) \wedge C_i')$$

By Lemma 6 and distributivity, the result is an equivalent formula. Repeat this process for every literal in $C_1 \vee \ldots \vee C_n$. The result can be written in the form

$$(\mathsf{undef} \wedge \phi_1) \vee \ldots \vee (\mathsf{undef} \wedge \phi_p) \vee \phi_{p+1} \vee \ldots \vee \phi_{p+q} \quad (2)$$

where each ϕ_i for $1 \leq i \leq p+q$ is a well-defined conjunction. Formula (2) is equivalent to

$$(\mathsf{undef} \wedge (\phi_1 \vee \ldots \vee \phi_p)) \vee \phi_{p+1} \vee \ldots \vee \phi_{p+q} \quad (3)$$

and is equivalent to the well-defined formula ψ, so it is well-defined. Formulas $\phi_1 \vee \ldots \vee \phi_p$ and $\phi_{p+1} \vee \ldots \vee \phi_{p+q}$ are also well-defined. By Lemma 7, we conclude that formula (3) is equivalent to

$$\phi_{p+1} \vee \ldots \vee \phi_{p+q} \quad (4)$$

Because (4) is a disjunction of well-defined formulas, (4) is the desired result ψ'. ∎

The following proposition presents transformation to unnested form for the structures with equality and partial function symbols, building on Proposition 8. For a similar unnested form in the first-order logic containing only total function symbols, see [22, Page 58].

Proposition 9 *Every well-defined quantifier-free formula ψ in a language with equality can be effectively transformed into an equivalent formula ψ' where ψ' is a disjunction of existentially quantified well-defined conjunctions of the following kinds of literals:*

- *$R(x_1, \ldots, x_k)$ where R is some relational symbol of arity k and $x_1, \ldots, x_k$ are variables;*
- *$\neg R(x_1, \ldots, x_k)$ where R is some relational symbol of arity k and $x_1, \ldots, x_k$ are variables;*
- *$x_1 = x_2$ where x_1, x_2 are variables;*
- *$x = f(x_1, \ldots, x_k)$ where f is some partial or total function symbol of arity k and $x, x_1, \ldots, x_k$ are variables;*
- *$x_1 \neq x_2$ where x_1 and x_2 are variables.*

Proof. Transform the formula to disjunction of well-formed conjunctions of literals as in the proof of Proposition 8.

Then repeatedly perform the following transformation on each well-defined conjunction ϕ. Let $A(f(x_1, \ldots, x_k))$ be an atomic formula containing term $f(x_1, \ldots, x_k)$. Replace $\phi \wedge A(f(x_1, \ldots, x_k))$ with

$$\exists x_0.\ \phi \wedge x_0 = f(x_1, \ldots, x_k) \wedge A(x_0)$$

Replace $x \neq f(x_1, \ldots, x_k)$ with

$$x_0 = f(x_1, \ldots, x_k) \wedge x_0 \neq x$$

Repeat this process until the resulting conjunction ϕ' is in unnested form. ϕ' is clearly equivalent to the original conjunction ϕ when all partial functions are well-defined. When some partial function is not well-defined, then both ϕ and ϕ' evaluate to false, because by construction of ϕ in the proof of Proposition 8, each conjunction contains conjuncts that evaluate to false when some application of a function symbol is not well-defined. ■

Let a *left-strict* conjunction in Kleene logic be denoted by $\wedge'$ and defined by

$$p \wedge' q = (p \wedge q) \vee (p \wedge \neg p)$$

The correctness of the transformation to unnested form in Proposition 9 relies on the presence of conjuncts that ensure that the entire conjunction evaluates to false whenever some term is undefined. The following Lemma 10 enables transformation to unnested form in an arbitrary context, allowing the transformation to unnested form to be performed independently from ensuring well-definedness of conjuncts.

Lemma 10 *Let $\phi(x)$ be a formula with free variable x and let t be a term possibly containing partial function symbols. Then*

1. $\phi(t) \;\cong\; (\exists x.\ x = t \;\wedge\; \phi(x)) \;\vee\; (\mathsf{undef} \wedge \forall x. \neg\phi(x))$;
2. $\phi(t) \;\cong\; \exists x.\ x = t \wedge' \phi(x)$;
3. $\phi(t) \;\cong\; (\exists x.\ x = t \;\wedge\; \phi(x)) \;\vee\; (t \neq t)$.

Proof. Straightforward. ■

Proposition 13 below shows that a simplification similar to one in Lemma 7 can be applied even within the scope of quantifiers. To show Proposition 13 we first show two lemmas.

Lemma 11 *For all formulas $\phi_0(x, \bar{y})$ and $\phi_1(x, \bar{y})$,*

$$\exists x.\ (\mathsf{undef} \wedge \phi_0(x, \bar{y})) \vee \phi_1(x, \bar{y}) \;\cong$$
$$(\mathsf{undef} \wedge \exists x. \phi_0(x, \bar{y})) \vee \exists x. \phi_1(x, \bar{y})$$

Proof. By distributivity of quantifiers and propositional connectives in Kleene logic we have:

$$\exists x.\ (\mathsf{undef} \wedge \phi_0(x, \bar{y})) \vee \phi_1(x, \bar{y}) \;\cong$$
$$(\exists x. \mathsf{undef} \wedge \phi_0(x, \bar{y})) \vee \exists x. \phi_1(x, \bar{y}) \;\cong$$
$$(\mathsf{undef} \wedge \exists x. \phi_0(x, \bar{y})) \vee \exists x. \phi_1(x, \bar{y})$$

■

Lemma 12 *For all formulas $\phi_0(x, \bar{y})$ and $\phi_1(x, \bar{y})$,*

$$\forall x.\ (\mathsf{undef} \wedge \phi_0(x, \bar{y})) \vee \phi_1(x, \bar{y}) \;\cong$$
$$(\mathsf{undef} \wedge \forall x. \phi_0(x, \bar{y}) \vee \phi_1(x, \bar{y})) \vee \forall x. \phi_1(x, \bar{y})$$

Proof. The following sequence of equivalences holds.

$$\forall x.\ (\mathsf{undef} \wedge \phi_0(x, \bar{y})) \vee \phi_1(x, \bar{y}) \;\cong$$
$$\neg\exists x. \neg(\mathsf{undef} \wedge \phi_0(x, \bar{y})) \vee \phi_1(x, \bar{y}) \;\cong$$
$$\neg\exists x.\ (\mathsf{undef} \vee \neg\phi_0(x, \bar{y})) \wedge \neg\phi_1(x, \bar{y}) \;\cong$$
$$\neg\exists x.\ (\mathsf{undef} \wedge \neg\phi_1(x, \bar{y})) \vee (\neg\phi_0(x, \bar{y}) \wedge \neg\phi_1(x, \bar{y})) \;\cong$$
$$\neg\left((\mathsf{undef} \wedge \exists x. \neg\phi_1(x, \bar{y})) \vee (\exists x.\ \neg\phi_0(x, \bar{y}) \wedge \neg\phi_1(x, \bar{y}))\right) \;\cong$$
$$(\mathsf{undef} \vee \forall x. \phi_1(x, \bar{y})) \vee (\forall x.\ \phi_0(x, \bar{y}) \vee \phi_1(x, \bar{y})) \;\cong$$
$$(\mathsf{undef} \wedge \forall x. \phi_0(x, \bar{y}) \vee \phi_1(x, \bar{y})) \vee \forall x. \phi_1(x, \bar{y})$$

■

Proposition 13 *Let $\phi_0(\bar{x}, \bar{y})$ and $\phi_1(\bar{x}, \bar{y})$ be well-defined formulas whose free variables are among $\bar{y}$ and let*

$$\psi(\bar{y}) \;\equiv\; Q_1 x_1 \ldots Q_n x_n.\ (\mathsf{undef} \wedge \phi_0(\bar{x}, \bar{y})) \vee \phi_1(\bar{x}, \bar{y})$$

where $Q_1, \ldots, Q_n$ are quantifiers. If $\psi(\bar{y})$ is well-defined for all values of variables $\bar{y}$, then

$$\psi(\bar{y}) \;\cong\; Q_1 x_1 \ldots Q_n x_n.\ \phi_1(\bar{x}, \bar{y})$$

Proof. Applying successively Lemmas 11 and 12 to quantifiers $Q_n, \ldots, Q_1$, we conclude

$$\psi(\bar{y}) \;\cong\; (\mathsf{undef} \wedge \phi_2(\bar{y})) \vee Q_1 x_1 \ldots Q_n x_n.\ \phi_1(\bar{x}, \bar{y})$$

for some formula $\phi_2(\bar{y})$. Then by Lemma 7,

$$\psi(\bar{y}) \;\cong\; Q_1 x_1 \ldots Q_n x_n.\ \phi_1(\bar{x}, \bar{y}).$$

■

3 Some Quantifier Elimination Procedures

As a preparation for the proof of the decidability of term algebras of decidable theories, we present quantifier elimination procedures for some theories that are known to admit quantifier elimination. We use the results and ideas from this section to show the new results in Sections 4, 5, 6.

3.1 Quantifier Elimination

Our technique for showing decidability of structural subtyping of recursive types is based on quantifier elimination. This section gives some general remarks on quantifier elimination.

We follow [22] in describing quantifier elimination procedures. According to [22, Page 70, Lemma 2.7.4] it suffices to eliminate $\exists y$ from formulas of the form

$$\exists y. \bigwedge_{0 \leq i < n} \psi_i(\bar{x}, y) \tag{5}$$

where $\bar{x}$ is a tuple of variables and $\psi_i(\bar{x}, y)$ is a literal whose all variables are among $\bar{x}, y$. The reason why eliminating formulas of the form (5) suffices is the following. Suppose that the formula in prenex form and consider the innermost quantifier of a formula. Let ϕ be the subformula containing the quantifier and the subformula that is the scope of the quantifier. If ϕ is of the form $\forall x.\ \phi_0$ we may replace ϕ with $\neg\exists x. \neg\phi_0$. Hence, we may assume that ϕ is of the form

$\exists x.\ \phi_1$. We then transform ϕ_1 into disjunctive normal form and use the fact

$$\exists x.\ (\phi_2 \vee \phi_3) \iff (\exists x.\ \phi_2) \vee (\exists x.\ \phi_3) \tag{6}$$

We conclude that elimination of quantifiers from formulas of form (5) suffices to eliminate the innermost quantifier. By repeatedly eliminating innermost quantifiers we can eliminate all quantifiers from a formula.

We may also assume that y occurs in every literal ψ_i, otherwise we would place the literal outside the existential quantifier using the fact

$$\exists y.\ (A \wedge B) \iff (\exists y. A) \wedge B$$

for y not occurring in B.

To eliminate variables we often use the following identity of a theory with equality:

$$\exists x. x = t \wedge \phi(x) \iff \phi(t) \tag{7}$$

Section 2.3 presents analogous identities for partial functions.

Quantifier elimination procedures we give imply the decidability of the underlying theories. In this paper the interpretations of function and relation symbols on some domain A are effectively computable functions and relations on A. Therefore, the truth-value of every formula without variables is computable. The quantifier elimination procedures we present are all effective. To determine the truth value of a closed formula ϕ it therefore suffices to apply the quantifier elimination procedure to ϕ, yielding a quantifier free formula ψ, and then evaluate the truth value of ψ.

3.2 Quantifier Elimination for Boolean Algebras

This section presents a quantifier elimination procedure for finite boolean algebras. This result dates back at least to [46], see also [51, 27, 32, 6, 49], [22, Section 2.7 Exercise 3]. Note that the operations union, intersection and complement are definable in the first-order language of the subset relation. Therefore, quantifier elimination for the first-order theory of the boolean algebra of sets is no harder than the quantifier elimination for the first-order theory of the subset relation. However, the operations of boolean algebra are useful in the process of quantifier elimination, so we give the quantifier elimination procedure for the language containing boolean algebra operations.

Instead of the first-order theory of the subtype relation we could consider monadic second-order theory with no relation or function symbols. These two languages are equivalent because the first-order quantifiers can be eliminated from monadic second-order theory using the subset relation (see Section 7.1).

Finite boolean algebras are isomorphic to boolean algebras whose elements are all subsets of some finite set. We therefore use the symbols for the set operations as the language of boolean algebras. $t_1 \cap t_2$, $t_1 \cup t_2$, t_1^c, 0, 1, correspond to set intersection, set union, set complement, empty set, and full set, respectively. We write $t_1 \subseteq t_2$ for $t_1 \cap t_2 = t_1$, we write $t_1 \subset t_2$ for the conjunction $t_1 \subseteq t_2\ \wedge\ t_1 \neq t_2$.

For every nonnegative integer k we introduce formulas $|t| \geq k$ expressing that the set denoted by t has at least k elements, and formulas $|t| = k$ expressing that the set denoted by t has exactly k elements. These properties are first-order definable as follows.

$$\begin{aligned} |t| \geq 0 &\quad\equiv\quad \mathsf{true} \\ |t| \geq k{+}1 &\quad\equiv\quad \exists x.\ x \subset t\ \wedge\ |x| \geq k \\ |t| = k &\quad\equiv\quad |t| \geq k\ \wedge\ \neg|t| \geq k{+}1 \end{aligned}$$

We call a language which contains terms $|t| \geq k$ and $|t| = k$ the language of boolean algebras with finite cardinality constraints. Because finite cardinality constraints are first-order definable, the language with finite cardinality constraints is equally expressive as the language of boolean algebras.

Every inequality $t_1 \subseteq t_2$ is equivalent to the equality $t_1 \cap t_2 = t_1$, and every equality $t_3 = t_4$ is equivalent to the cardinality constraint

$$|(t_3 \cap t_4^c) \cup (t_4 \cap t_3^c)| = 0$$

It is therefore sufficient to consider the first-order formulas whose only atomic formulas are of the form $|t| = 0$. For the purpose of quantifier elimination we will additionally consider formulas that contain atomic formulas $|t|{=}k$ for all $k \geq 1$, as well as $|t|{\geq}k$ for $k \geq 0$.

Note that we can eliminate negative literals as follows:

$$\begin{aligned} \neg|t| = k &\iff |t| = 0\ \vee \cdots \vee\ |t| = k{-}1\ \vee |t| \geq k{+}1 \\ \neg|t| \geq k &\iff |t| = 0\ \vee \cdots \vee\ |t| = k{-}1 \end{aligned} \tag{8}$$

Every formula in the language of boolean algebras can therefore be written in prenex normal form where the matrix of the formulas is a disjunction of conjunctions of atomic formulas of the form $|t| = k$ and $|t| \geq k$, with no negative literals.

Note that if a term t contains at least one operation of arity one or more, we may assume that the constants 0 and 1 do not appear in t, because 0 and 1 can be simplified away. Furthermore, the expression $|0|$ denotes the integer zero, so all terms of form $|0| = k$ or $|0| \geq k$ evaluate to true or false. We can therefore simplify every nontrivial term t so that it either t contains no occurrences of constants 0 and 1, or $t \equiv 1$.

We next describe a quantifier elimination procedure for finite boolean algebras.

We first transform the formula into prenex normal form and then repeatedly eliminate the innermost quantifier. As argued in Section 3.1, it suffices to show that we can eliminate an existential quantifier from any existentially quantified conjunction of literals. Consider therefore an arbitrary existentially quantified conjunction of literals

$$\exists y. \bigwedge_{1 \leq i \leq n} \psi_i(\bar{x}, y)$$

where ψ_i is of the form $|t| = k$ or of the form $|t| \geq k$. We assume that y occurs in every formula ψ_i. It follows that no ψ_i contains $|0|$ or $|1|$.

Let $x_1, \ldots, x_m, y$ be the set of variables occurring in formulas ψ_i for $1 \leq i \leq n$.

First consider the more general case $m \geq 1$. Let for $i_1, \ldots, i_m \in \{0, 1\}$,

$$t_{i_1 \ldots i_m} = x_1^{i_1} \cap \cdots \cap x_m^{i_m}$$

where $t^0 = t$ and $t^1 = t^c$. The terms in the set

$$P = \{t_{i_1 \ldots i_m} \mid i_1, \ldots, i_m \in \{0, 1\}\}$$

original formula	eliminated form
$\exists y.\ \lvert s\cap y\rvert \geq k \wedge \lvert s\cap y^c\rvert \geq l$	$\lvert s\rvert \geq k+l$
$\exists y.\ \lvert s\cap y\rvert = k \wedge \lvert s\cap y^c\rvert \geq l$	$\lvert s\rvert \geq k+l$
$\exists y.\ \lvert s\cap y\rvert \geq k \wedge \lvert s\cap y^c\rvert = l$	$\lvert s\rvert \geq k+l$
$\exists y.\ \lvert s\cap y\rvert = k \wedge \lvert s\cap y^c\rvert = l$	$\lvert s\rvert = k+l$

Figure 1: Rules for Eliminating Quantifiers

form a partition; moreover every boolean algebra expression whose variables are among x_i can be written as a disjoint union of some elements of the partition P. Any boolean algebra expression containing y can be written, for some $p, q \geq 0$ as

$$(s_1 \cap y) \cup \cdots \cup (s_p \cap y) \cup$$
$$(t_1 \cap y^c) \cup \cdots \cup (t_q \cap y^c)$$

where $s_1, \ldots, s_p \in P$ are pairwise distinct elements from the partition and $t_1, \ldots, t_q \in P$ are pairwise distinct elements from the partition. Because

$$|(s_1 \cap y) \cup \cdots \cup (s_p \cap y) \cup (t_1 \cap y^c) \cup \cdots \cup (t_q \cap y^c)| =$$
$$|s_1 \cap y| + \cdots + |s_p \cap y| + |t_1 \cap y^c| + \cdots + |t_q \cap y^c|$$

the constraint of form $|t| = k$ can be written as

$$\bigvee_{k_1,\ldots,k_p,l_1,\ldots,l_q} \begin{array}{l} |s_1 \cap y| = k_1 \wedge \cdots \wedge |s_p \cap y| = k_p \wedge \\ |t_1 \cap y^c| = l_1 \wedge \cdots \wedge |t_q \cap y^c| = l_p \end{array}$$

where the disjunction ranges over nonnegative integers $k_1, \ldots, k_p, l_1, \ldots, l_q \geq 0$ that satisfy

$$k_1 + \cdots + k_p + l_1 + \cdots + l_q = k$$

From (8) it follows that we can perform a similar transformation for constraints of form $|t| \geq k$. After performing this transformation, we bring the formula into disjunctive normal form and continue eliminating the existential quantifier separately for each disjunct, as argued in Section 3.1. We may therefore assume that all conjuncts ψ_i are of one of the forms: $|s \cap y| = k$, $|s \cap y^c| = k$, $|s \cap y| \geq k$, and $|s \cap y^c| \geq k$ where $s \in P$.

If there are two conjuncts both of which contain $|s \cap y|$ for the same s, then either they are contradictory or one implies the other. We therefore assume that for any $s \in P$, there is at most one conjunct ψ_i containing $|s \cap y|$. For analogous reasons we assume that for every $s \in P$ there is at most one conjunct ψ_i containing $|s \cap y^c|$. The result of eliminating the variable y is then given in Figure 1. The case when a literal containing $|s \cap y|$ does not occur is covered by the case $|s \cap y| \geq k$ for $k = 0$, similarly for a literal containing $|s \cap y^c|$.

It remains to consider the case $m = 0$. Then y is the only variable occurring in conjuncts ψ_i. Every cardinality expression t containing only y reduces to one of $|y|$ or $|y^c|$. If there are multiple literals containing $|y|$, they are either contradictory or one implies the others. We may therefore assume there is at most one literal containing $|y|$ and at most one literal containing $|y^c|$. We eliminate quantifier by applying rules in Figure 1 putting formally $s = 1$ where 1 is the universal set.

This completes the description of quantifier elimination from an existentially quantified conjunction. By repeating this process for all quantifiers we arrive at a quantifier-free formula ψ. Hence we have the following theorem.

Theorem 14 *For every first-order formula ϕ in the language of boolean algebras with finite cardinality constraints there exists a quantifier-free formula ψ such that ψ is a disjunction of conjunctions of literals of form $|t| \geq k$ and $|t| = k$ where t are terms of boolean algebra, the free variables of ψ are a subset of the free variables of ϕ, and ψ is equivalent to ϕ on all algebras of finite sets.*

Remark 15 Now consider the case when formula ϕ has no free variables. By Theorem 14, ϕ is equivalent to ψ where ψ contains only terms without variables. A term without variables in boolean algebra can always be simplified to 0 or 1. Because $|0| = 0$, the literals with $|0|$ reduce to true or false, so we may simplify them away. The expression $|1|$ evaluates to the number of elements in the boolean algebra. We call literals $|1| = k$ and $|1| \geq k$ *domain cardinality constraints.* A quantifier-free formula ψ can therefore be written as a propositional combination of domain cardinality constraints. We can simplify ψ into a disjunction of conjunctions of domain cardinality constraints and transform each conjunction so that it contains at most one literal. The result ψ' is a single disjunction of domain cardinality constraints. We may further assume that the disjunct of form $|1| \geq k$ occurs at most once. Therefore, the truth value of each closed boolean algebra formula is characterized by a set C of possible cardinalities of the domain. If ψ' does not contain any $|1| \geq k$ literals, the set C is finite. Otherwise, $C = C_0 \cup \{k, k+1, \ldots\}$ for some k where C_0 is a finite subset of $\{1, \ldots, k-1\}$.

3.3 Feferman-Vaught Theorem

> *The Feferman-Vaught technique is a way of discovering the first-order theories of complex structures by analyzing their components. This description is a little vague, and in fact the Feferman-Vaught technique itself has something of a floating identity. It works for direct products, as we shall see. Clever people can make it work in other situations too.* — *[22], page 458*

We next review Feferman-Vaught theorem for direct products [14] which implies that the products of structures with decidable first-order theories have decidable first-order theories.

The result was first obtained for strong and weak powers of theories in [35]; [35] also suggests the generalization to products. Our sketch here mostly follows [14] and [35], see also [31, Chapter 12] as well as [22, Section 9.6]. Somewhat specific to our presentation is the fact that we use a multisorted logic and build into the language the correspondence between formulas interpreted over C and the cylindric algebra of sets of positions.

Let L_C be a relational language. Let further I be some nonempty finite or countably infinite index set. For each $i \in I$ let $\mathcal{C}_i = \langle C_i, [\![\_]\!]^{\mathcal{C}_i} \rangle$ be a decidable structure interpreting the language L_C.

We define *direct product* of the family of structures $\mathcal{C}_i$, $i \in I$, as the structure

$$\mathcal{P} = \Pi_{i \in I} \mathcal{C}_i$$

where $\mathcal{P} = \langle P, [\![\_]\!]^P \rangle$. P is the set of all functions t such that $t(i) \in C_i$ for $i \in I$, and $[\![\_]\!]^P$ is defined by

$$[\![r]\!]^P(t_1, \ldots, t_k) \;=\; \forall i.\; [\![r]\!]^{\mathcal{C}_i}(t_1(i), \ldots, t_k(i))$$

for each relation symbol $r \in L_C$.

$$\begin{array}{rcl}
\multicolumn{3}{c}{\textbf{inner formula relations for } r \in L_C} \\
r & :: & \mathsf{tuple}^k \to \mathsf{indset} \\
\multicolumn{3}{c}{\textbf{inner logical connectives}} \\
\wedge^{\mathsf{I}}, \vee^{\mathsf{I}} & :: & \mathsf{indset} \times \mathsf{indset} \to \mathsf{indset} \\
\neg^{\mathsf{I}} & :: & \mathsf{indset} \to \mathsf{indset} \\
\mathsf{true}^{\mathsf{I}}, \mathsf{false}^{\mathsf{I}} & :: & \mathsf{indset} \\
\multicolumn{3}{c}{\textbf{inner formula quantifiers}} \\
\exists^{\mathsf{I}}, \forall^{\mathsf{I}} & :: & (\mathsf{tuple} \to \mathsf{indset}) \to \mathsf{indset} \\
\multicolumn{3}{c}{\textbf{index set equality}} \\
=^{\mathsf{I}} & :: & \mathsf{indset} \times \mathsf{indset} \to \mathsf{bool} \\
\multicolumn{3}{c}{\textbf{logical connectives}} \\
\wedge, \vee & :: & \mathsf{bool} \times \mathsf{bool} \to \mathsf{bool} \\
\neg & :: & \mathsf{bool} \to \mathsf{bool} \\
\mathsf{true}, \mathsf{false} & :: & \mathsf{bool} \\
\multicolumn{3}{c}{\textbf{index set quantifiers}} \\
\exists^{\mathsf{L}}, \forall^{\mathsf{L}} & :: & (\mathsf{indset} \to \mathsf{bool}) \to \mathsf{bool} \\
\multicolumn{3}{c}{\textbf{tuple quantifiers}} \\
\exists, \forall & :: & (\mathsf{tuple} \to \mathsf{bool}) \to \mathsf{bool}
\end{array}$$

Figure 2: Operations in product structure

For the purpose of quantifier elimination we consider a richer language of statements about product structure $\mathcal{P}$. Figure 2 shows this richer language. The corresponding structure $\mathcal{P}_2 = \langle P_2, [\![\_]\!]^{P_2} \rangle$ contains, in addition to the function space P, a copy of the boolean algebra 2^I of subsets of the index set I. We interpret a relation $r \in L_C$ by

$$[\![r]\!]^{P_2}(t_1, \ldots, t_k) = \{\, i \mid [\![r]\!]^{\mathcal{C}_i}(t_1(i), \ldots, t_k(i)) \,\}$$

We let $[\![\mathsf{true}^{\mathsf{I}}]\!]^{P_2} = I$ and write

$$r(t_1, \ldots, t_k) =^{\mathsf{I}} \mathsf{true}^{\mathsf{I}}$$

to express $[\![r]\!]^P(t_1, \ldots, t_k)$. Hence P_2 is at least as expressive as $\mathcal{P}$.

Note that Figure 2 does not contain an equality relation between tuples. If we need to express the equality between tuples, we assume that some binary relation $r_0 \in L_C$ in the base structure is interpreted as equality, and express the equality between tuples t_1 and t_2 using the formula:

$$r_0(t_1, t_2) =^{\mathsf{I}} \mathsf{true}^{\mathsf{I}}.$$

Figure 3 shows the semantics of the language in Figure 2. (The logic has no partial functions, so we interpret the sort bool over the set $\{\mathsf{true}, \mathsf{false}\}$.)

$$\begin{array}{rcl}
\multicolumn{3}{c}{\textbf{inner formula relations for } r \in L_C} \\
[\![r]\!]^{P_2}(t_1, \ldots, t_k) & = & \{\, i \mid [\![r]\!]^{\mathcal{C}_i}(t_1(i), \ldots, t_k(i)) \,\} \\
\multicolumn{3}{c}{\textbf{inner logical connectives}} \\
[\![\wedge^{\mathsf{I}}]\!]^{P_2}(A_1, A_2) & = & A_1 \wedge A_2 \\
[\![\vee^{\mathsf{I}}]\!]^{P_2}(A_1, A_2) & = & A_1 \cup A_2 \\
[\![\neg^{\mathsf{I}}]\!]^{P_2}(A) & = & I \setminus A \\
[\![\mathsf{true}^{\mathsf{I}}]\!]^{P_2} & = & I \\
[\![\mathsf{false}^{\mathsf{I}}]\!]^{P_2} & = & \emptyset \\
\multicolumn{3}{c}{\textbf{inner formula quantifiers}} \\
[\![\exists^{\mathsf{I}}]\!]^{P_2} f & = & \bigcup_{t \in P} f(t) \\
[\![\forall^{\mathsf{I}}]\!]^{P_2} f & = & \bigcap_{t \in P} f(t) \\
\multicolumn{3}{c}{\textbf{index set equality}} \\
[\![=^{\mathsf{I}}]\!]^{P_2}(A_1, A_2) & = & (A_1 = A_2) \\
\multicolumn{3}{c}{\textbf{logical connectives}} \\
\multicolumn{3}{c}{\text{(interpreted as usual)}} \\
\multicolumn{3}{c}{\textbf{index set quantifiers}} \\
[\![\exists^{\mathsf{L}}]\!]^{P_2} f & = & \bigcup_{A \in 2^I} f(A) \\
[\![\forall^{\mathsf{L}}]\!]^{P_2} f & = & \bigcap_{A \in 2^I} f(A) \\
\multicolumn{3}{c}{\textbf{tuple quantifiers}} \\
[\![\exists]\!]^{P_2} f & = & \exists t \in P.\; f(t) \\
[\![\forall]\!]^{P_2} f & = & \forall t \in P.\; f(t)
\end{array}$$

Figure 3: Semantics of operations in product structure $\mathcal{P}_2$

We let $A_1 \subseteq^I A_2$ stand for $A_1 \wedge^{\mathsf{I}} A_2 =^{\mathsf{I}} A_2$.

Note that the interpretations of $\wedge^{\mathsf{I}}$, $\vee^{\mathsf{I}}$, $\neg^{\mathsf{I}}$, $\mathsf{true}^{\mathsf{I}}$, $\mathsf{false}^{\mathsf{I}}$, $=^{\mathsf{I}}$, $\exists^{\mathsf{L}}$, $\forall^{\mathsf{L}}$ form a first-order structure of boolean algebras of subsets of the set I. We call formulas in this boolean algebra sublanguage *index-set algebra formulas.*

On the other hand, relations r for $r \in L_C$, together with $\wedge^{\mathsf{I}}$, $\vee^{\mathsf{I}}$, $\neg^{\mathsf{I}}$, $\exists^{\mathsf{I}}$, $\forall^{\mathsf{I}}$ form the signature of first-order logic with relation symbols. We call formulas built only from these operations *inner formulas.*

Let ϕ be a an inner formula with free tuple variables $t_1, \ldots, t_m$ and no free indset variables. Then ϕ specifies a relation $\rho \subseteq D^m$. Consider the corresponding first-order formula ϕ' interpreted in the base structure $\mathcal{C}$; formula ϕ' specifies a relation $\rho' \subseteq C^m$. The following property follows from the semantics in Figure 3:

$$\rho(t_1, \ldots, t_m) = \{\, i \in I \mid \rho'(t_1(i), \ldots, t_m(i)) \,\} \tag{9}$$

Sort constraints imply that quantifiers $\exists^{\mathsf{I}}, \forall^{\mathsf{I}}$ are only applied to inner formulas. Let ϕ be a formula of sort bool. By labelling subformulas of sort indset with variables $A_1, \ldots, A_n$, we can write ϕ in form ϕ^1:

$$\begin{aligned}
&\exists^{\mathsf{L}} A_1, \ldots, A_n. \\
&\quad A_1 =^{\mathsf{I}} \phi_1 \ \wedge \ldots \wedge \ A_n =^{\mathsf{I}} \phi_n \ \wedge \\
&\quad \psi(A_1, \ldots, A_n)
\end{aligned}$$

where

$$\phi = \psi(\phi_1, \ldots, \phi_n)$$

Furthermore, by defining $B_1, \ldots, B_m$ to be the partition of $\mathsf{true}^{\mathsf{I}}$ consisting of terms of form

$$A_1^{p_1} \ \wedge^{\mathsf{I}} \ldots \wedge^{\mathsf{I}} A_n^{p_n}$$

for $p_1, \ldots, p_n \in \{0, 1\}$, we can find a formula ψ' and formulas $\phi'_1, \ldots, \phi'_m$ such that ϕ^1 is equivalent to ϕ^2:

$$\begin{aligned}
&\exists^{\mathsf{L}} B_1, \ldots, B_m. \\
&\quad B_1 =^{\mathsf{I}} \phi'_1 \ \wedge \ldots \wedge \ B_m =^{\mathsf{I}} \phi'_m \ \wedge \\
&\quad \psi'(B_1, \ldots, B_n)
\end{aligned} \tag{10}$$

and where $\phi'_1, \ldots, \phi'_m$ evaluate to sets that form partition of $\mathsf{true}^{\mathsf{I}}$ for all values of free variables. (By partition of $\mathsf{true}^{\mathsf{I}}$ we here mean a family of pairwise disjoint sets whose union is $\mathsf{true}^{\mathsf{I}}$, but we do not require the sets to be non-empty.)

Now consider a formula of form $\exists t.\phi$ where ϕ is without $\exists, \forall$ quantifiers (but possibly contains $\exists^{\mathsf{I}}, \forall^{\mathsf{I}}$ and $\exists^{\mathsf{L}}, \forall^{\mathsf{L}}$ quantifiers). We transform ϕ into ϕ^2 as described, and then replace

$$\begin{aligned}
&\exists t.\ \exists^{\mathsf{L}} B_1, \ldots, B_m. \\
&\quad B_1 =^{\mathsf{I}} \phi'_1 \ \wedge \ldots \wedge \ B_m =^{\mathsf{I}} \phi'_m \ \wedge \\
&\quad \psi'(B_1, \ldots, B_n)
\end{aligned} \tag{11}$$

with

$$\begin{aligned}
&\exists^{\mathsf{L}} D_1, \ldots, D_m.\ \exists^{\mathsf{L}} B_1, \ldots, B_m. \\
&\quad D_1 =^{\mathsf{I}} (\exists^{\mathsf{I}} t.\phi'_1) \ \wedge \ldots \wedge \ D_m =^{\mathsf{I}} (\exists^{\mathsf{I}} t.\phi'_m) \ \wedge \\
&\quad B_1 \subseteq^I D_1 \ \wedge \ldots \wedge \ B_m \subseteq^I D_m \ \wedge \\
&\quad \mathsf{partition}(B_1, \ldots, B_n) \wedge \psi'(B_1, \ldots, B_n)
\end{aligned} \tag{12}$$

where $\mathsf{partition}(B_1, \ldots, B_n)$ denotes a boolean algebra expression expressing that sets $B_1, \ldots, B_n$ form the partition of $\mathsf{true}^{\mathsf{I}}$.

It is easy to see that 11 and 12 are equivalent.

By repeating this construction we eliminate all term quantifiers from a formula. We then eliminate all set quantifiers as in Section 3.2. For that purpose we extend the language with cardinality constraints.

As the result we obtain cardinality constraints on inner formulas. Closed inner formulas evaluate to $\mathsf{true}^{\mathsf{I}}$ or $\mathsf{false}^{\mathsf{I}}$ depending on their truth value in base structure $\mathcal{C}$. Hence, if $\mathcal{C}$ is decidable, so is $\mathcal{P}_2$.

Theorem 16 (Feferman-Vaught) *Let $\mathcal{C}$ be a decidable structure. Then every formula in the language of Figure 2 is equivalent on the structure P_2 to a propositional combination of cardinality constraints of the index-set boolean algebra i.e. formulas of form $|\phi| \geq k$ and $|\phi| = k$ where ϕ is an inner formula.*

Example 17 Let $r \in L_C$ be a binary relation on structure C. Let us eliminate quantifier $\exists t$ from the formula $\phi(t_1, t_2)$:

$$\begin{aligned}
&\exists t. \exists^{\mathsf{L}} A_1, A_2, A_3. \\
&\quad A_1 =^{\mathsf{I}} r(t, t_1) \ \wedge \ A_2 =^{\mathsf{I}} r(t_1, t) \ \wedge \ A_3 =^{\mathsf{I}} r(t_2, t) \wedge \\
&\quad |\neg^{\mathsf{I}} A_1| = 0 \ \wedge \ |\neg^{\mathsf{I}} A_2| = 0 \ \wedge |\neg^{\mathsf{I}} A_3| \geq 1
\end{aligned}$$

We first introduce sets $B_0, \ldots, B_7$ that form partition of $\mathsf{true}^{\mathsf{I}}$. The formula is then equivalent to ϕ_1:

$$\begin{aligned}
&\exists t. \exists^{\mathsf{L}} B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7. \\
&\quad B_0 =^{\mathsf{I}} r(t, t_1) \wedge^{\mathsf{I}} r(t_1, t) \wedge^{\mathsf{I}} r(t_2, t) \ \wedge \\
&\quad B_1 =^{\mathsf{I}} \neg^{\mathsf{I}} r(t, t_1) \wedge^{\mathsf{I}} r(t_1, t) \wedge^{\mathsf{I}} r(t_2, t) \ \wedge \\
&\quad B_2 =^{\mathsf{I}} r(t, t_1) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_1, t) \wedge^{\mathsf{I}} r(t_2, t) \ \wedge \\
&\quad B_3 =^{\mathsf{I}} \neg^{\mathsf{I}} r(t, t_1) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_1, t) \wedge^{\mathsf{I}} r(t_2, t) \ \wedge \\
&\quad B_4 =^{\mathsf{I}} r(t, t_1) \wedge^{\mathsf{I}} r(t_1, t) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_2, t) \ \wedge \\
&\quad B_5 =^{\mathsf{I}} \neg^{\mathsf{I}} r(t, t_1) \wedge^{\mathsf{I}} r(t_1, t) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_2, t) \ \wedge \\
&\quad B_6 =^{\mathsf{I}} r(t, t_1) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_1, t) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_2, t) \ \wedge \\
&\quad B_7 =^{\mathsf{I}} \neg^{\mathsf{I}} r(t, t_1) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_1, t) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_2, t) \ \wedge \\
&\quad \phi_0
\end{aligned}$$

where

$$\begin{aligned}
\phi_0 \ \equiv \ & \\
& |B_1| = 0 \ \wedge \ |B_2| = 0 \ \wedge \\
& |B_3| = 0 \ \wedge \ |B_5| = 0 \ \wedge \\
& |B_6| = 0 \ \wedge \ |B_7| = 0 \ \wedge \\
& |B_4| \geq 1
\end{aligned}$$

We now eliminate the quantifier $\exists t$ from the formula ϕ_1, obtaining formula ϕ_2:

$$
\begin{aligned}
&\exists^{\mathsf{L}} D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7.\\
&\quad D_0 =^{\mathsf{I}} \exists^{\mathsf{I}} t.\ r(t,t_1) \wedge^{\mathsf{I}} r(t_1,t) \wedge^{\mathsf{I}} r(t_2,t)\ \wedge\\
&\quad D_1 =^{\mathsf{I}} \exists^{\mathsf{I}} t.\ \neg^{\mathsf{I}} r(t,t_1) \wedge^{\mathsf{I}} r(t_1,t) \wedge^{\mathsf{I}} r(t_2,t)\ \wedge\\
&\quad D_2 =^{\mathsf{I}} \exists^{\mathsf{I}} t.\ r(t,t_1) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_1,t) \wedge^{\mathsf{I}} r(t_2,t)\ \wedge\\
&\quad D_3 =^{\mathsf{I}} \exists^{\mathsf{I}} t.\ \neg^{\mathsf{I}} r(t,t_1) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_1,t) \wedge^{\mathsf{I}} r(t_2,t)\ \wedge\\
&\quad D_4 =^{\mathsf{I}} \exists^{\mathsf{I}} t.\ r(t,t_1) \wedge^{\mathsf{I}} r(t_1,t) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_2,t)\ \wedge\\
&\quad D_5 =^{\mathsf{I}} \exists^{\mathsf{I}} t.\ \neg^{\mathsf{I}} r(t,t_1) \wedge^{\mathsf{I}} r(t_1,t) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_2,t)\ \wedge\\
&\quad D_6 =^{\mathsf{I}} \exists^{\mathsf{I}} t.\ r(t,t_1) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_1,t) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_2,t)\ \wedge\\
&\quad D_7 =^{\mathsf{I}} \exists^{\mathsf{I}} t.\ \neg^{\mathsf{I}} r(t,t_1) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_1,t) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_2,t)\ \wedge\\
&\quad \phi_3
\end{aligned}
$$

where

$$
\begin{aligned}
\phi_3 \ \equiv\ & \exists^{\mathsf{L}} B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7.\\
& B_0 \subseteq^I D_0\ \wedge \ldots \wedge\ B_7 \subseteq^I D_7\ \wedge\\
& \phi_0
\end{aligned}
$$

We next apply quantifier elimination for boolean algebras to formula ϕ_3 and obtain formula ϕ_3':

$$\phi_3' \ \equiv\ |D_4| \geq 1\ \wedge\ |\neg^{\mathsf{I}} D_0 \wedge^{\mathsf{I}} \neg^{\mathsf{I}} D_4| = 0$$

Hence $\phi(t_1,t_2)$ is equivalent to

$$
\begin{aligned}
&\exists^{\mathsf{L}} D_0, D_4.\\
&\quad D_0 =^{\mathsf{I}} \exists^{\mathsf{I}} t.\ r(t,t_1) \wedge^{\mathsf{I}} r(t_1,t) \wedge^{\mathsf{I}} r(t_2,t)\ \wedge\\
&\quad D_4 =^{\mathsf{I}} \exists^{\mathsf{I}} t.\ r(t,t_1) \wedge^{\mathsf{I}} r(t_1,t) \wedge^{\mathsf{I}} \neg^{\mathsf{I}} r(t_2,t)\ \wedge\\
&\quad |D_4| \geq 1 \wedge |\neg^{\mathsf{I}} D_0 \wedge^{\mathsf{I}} \neg^{\mathsf{I}} D_4| = 0
\end{aligned}
$$

After substituting the definitions of D_0 and D_4, formula $\phi(t_1,t_2)$ can be written without quantifiers $\exists, \forall, \exists^{\mathsf{L}}, \forall^{\mathsf{L}}$.

♦

3.4 Term Algebras

In this section we present a quantifier elimination procedure for term algebras (see Section 2.1). A quantifier elimination procedure for term algebras implies that the first-order theory of term algebras is decidable. In the sections below we build on the procedure in this section to define quantifier elimination procedures for structural subtyping.

The decidability of the first-order theory of term algebras follows from Mal'cev's work on locally free algebras [31, Chapter 23]. [39] also gives an argument for decidability of term algebra and presents a unification algorithm based on congruence closure [38]. Infinite trees are studied in [12]. [30] presents a complete axiomatization for algebra of finite, infinite and rational trees. A proof in the style of [22] for an extension of free algebra with queues is presented in [43]. Decidability of an extension of term algebras with membership tests is presented in [10] in the form of a terminating term rewriting system. Unification and disunification problems are special cases of decision problem for first-order theory of term algebras, for a survey see e.g. [45, 9].

We believe that our proof provides some insight into different variations of quantifier elimination procedures for term algebras. Like [22] we use selector language symbols, but retain the usual constructor symbols as well. The advantage of the selector language is that $\exists y.\ z = f(x,y)$ is equivalent to a quantifier-free formula $x = f_1(z)\ \wedge\ \mathsf{Is}_f(z)$. On the other hand, constructor symbols also increase the set of relations on terms definable via quantifier-free formulas, which can slightly simplify quantifier-elimination procedure, as will be seen by comparing Proposition 34 and Proposition 38. Compared to [22, Page 70], we find that the termination of our procedure is more evident and the extension to the term-power algebra in Section 6 easier. Our base formulas somewhat resemble formulas arising in other quantifier elimination procedures [31, 11, 30]. Our terminology also borrows from congruence closure graphs like those of [39, 38], although we are not primarily concerned with efficiency of the algorithm described. Term algebra is an example of a theory of *pairing functions*, and [15] shows that non-empty family of theories of pairing functions as non-elementary lower bound on time complexity.

3.4.1 Term Algebra in Selector Language

To facilitate quantifier elimination we use a *selector language* $\mathsf{Sel}(\Sigma)$ for term algebra [22, Page 61]. We define term algebra in selector language as a first-order structure with partial functions.

The set $\mathsf{Sel}(\Sigma)$ contains, for every function symbol $f \in \Sigma$ of arity $\mathsf{ar}(f) = k$, a unary predicate $\mathsf{Is}_f \subseteq \mathsf{FT}(\Sigma)$ and functions $f_1, \ldots, f_k : \mathsf{FT}(\Sigma) \to \mathsf{FT}(\Sigma)$ such that

$$
\begin{aligned}
\mathsf{Is}_f(t) &\iff \exists t_1, \ldots, t_k.\ t = f(t_1, \ldots, t_k) && (13)\\
f_i(f(t_1, \ldots, t_k)) &= t_i, \quad 1 \leq i \leq k && (14)\\
f_i(t) &= \bot, \quad \neg \mathsf{Is}_f(t) && (15)
\end{aligned}
$$

For every $f \in \Sigma$ and $1 \leq i \leq \mathsf{ar}(f)$, expression $f_i(t)$ defined iff $\mathsf{Is}_f(t)$ holds, so we let $D_f = \langle x, \mathsf{Is}_f(x) \rangle$.

As a special case, if d is a constant, then $\mathsf{ar}(d) = 0$ and $\mathsf{Is}_d(t) \iff t = d$.

Proposition 18 *For every formula ϕ_1 in the language $\mathsf{Cons}(\Sigma)$ there exists an equivalent formula ϕ_2 in the selector language.*

Proof Sketch. Because of the presence of equality symbol, every formula in language $\mathsf{Cons}(\Sigma)$ can be written in unnested form such that every atomic formula is of two forms: $x_1 = x_2$, or $f(x_1, \ldots, x_k) = y$, where y and x_i are variables. We keep every formula $x_1 = x_2$ unchanged and transform each formula

$$f(x_1, \ldots, x_k) = y$$

into the well-defined conjunction

$$x_1 = f_1(y)\ \wedge \cdots \wedge\ x_k = f_k(y)\ \wedge\ \mathsf{Is}_f(y)$$

■

Note that predicates Is_f form a partition of the set of all terms i.e. the following formulas are valid:

$$
\begin{aligned}
&\forall x.\ \bigvee_{f \in \Sigma} \mathsf{Is}_f(x)\\
&\forall x.\ \neg(\mathsf{Is}_f(x) \wedge \mathsf{Is}_g(x)), \qquad \text{for } f \not\equiv g
\end{aligned} \qquad (16)
$$

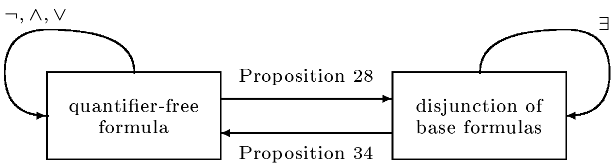

Figure 4: Quantifier Elimination for Term Algebra

A *constructor-selector* language contains both constructor symbols $f \in \mathsf{Cons}(\Sigma)$ and selector symbols $f_i \in \mathsf{Sel}(\Sigma)$.

3.4.2 Quantifier Elimination

We proceed to quantifier elimination for term algebra. A schematic view of our proof is in Figure 4. The basic insight is that any quantifier-free formula can be written in a particular unnested form, as a disjunction of *base formulas.* Base formulas trivially permit elimination of an existential quantifier, yet every base formula can be converted back to a quantifier-free formula.

A semi-base formula is almost the base formula, except that it may be cyclic. We introduce cyclicity after explaining the graph representation of a semi-base formula.

Definition 19 (Semi-Base Formula) *A semi-base formula β with*

- *free variables $x_1, \ldots, x_m$,*
- *internal non-parameter variables $u_1, \ldots, u_p$, and*
- *internal parameter variables $u_{p+1}, \ldots, u_{p+q}$*

is a formula of form

$$\begin{aligned} &\exists u_1, \ldots, u_n \\ &\qquad \mathsf{distinct}(u_1, \ldots, u_n) \ \wedge \\ &\qquad \mathsf{structure}(u_1, \ldots, u_n) \ \wedge \\ &\qquad \mathsf{labels}(u_1, \ldots, u_n; x_1, \ldots, x_m) \end{aligned}$$

$\mathsf{distinct}(u_1, \ldots, u_n)$ *enforces that variables are distinct*

$$\mathsf{distinct}(u_1, \ldots, u_n) \equiv \bigwedge_{1 \le i < j \le n} u_i \neq u_j \ .$$

$\mathsf{structure}(u_1, \ldots, u_n)$ *specifies relationships between terms denoted by variables:*

$$\mathsf{structure}(u_1, \ldots, u_n) \equiv$$
$$\bigwedge_{i=1}^{p} u_i = t_i(u_1, \ldots, u_n)$$

where each $t_i(u_1, \ldots, u_n)$ is a term of form $f(u_{l_1}, \ldots, u_{l_k})$ for $f \in \Sigma$, $k = \mathsf{ar}(f)$.

$\mathsf{labels}(u_1, \ldots, u_n; x_1, \ldots, x_m)$ *identifies some free variables with some parameter and non-parameter variables:*

$$\mathsf{labels}(u_1, \ldots, u_n; x_1, \ldots, x_m) \equiv \bigwedge_{1 \le i \le m} x_i = u_{j_i}$$

for some function $j : \{1, \ldots, m\} \to \{1, \ldots, n\}$.

We require each semi-base formula to satisfy the following congruence closure property: there are no two distinct variables u_i and $u_{i'}$ such that both $u_i = f(u_{l_1}, \ldots, u_{l_k})$ and $u_{i'} = f(u_{l_1}, \ldots, u_{l_k})$ occur as conjuncts ϕ_j in formula $\mathsf{structure}$.

We denote by U the set of internal variables of a given semi-base formula, $U = \{u_1, \ldots, u_n\}$.

Definition 20 *A semi-base formula in selector language is obtained from the base formula in constructor language by replacing every conjunct of form*

$$u_i = f(u_{l_1}, \ldots, u_{l_k})$$

with the well-defined conjunction

$$\mathsf{Is}_f(u_i) \ \wedge \ u_{l_1} = f_1(u_i) \ \wedge \cdots \wedge \ u_{l_k} = f_k(u_i)$$

A semi-base formula in selector language is clearly a well-formed conjunction of literals. All atomic formulas in a semi-base formula are unnested, in both constructor and selector language.

We can represent a base formula as a labelled directed graph with the set of nodes U; we call this graph *graph associated with a semi-base formula.* Nodes of the graph are in a bijection with internal variables of the semi-base formula. We call nodes corresponding to parameter variables $u_{p+1}, \ldots, u_{p+q}$ *parameter nodes*; nodes $u_1, \ldots, u_p$ are *non-parameter nodes.* Each non-parameter node is labelled by a function symbol $f \in \Sigma$ and has exactly $\mathsf{ar}(f)$ successors, with edge from u_k to u_l labelled by the positive integer i iff $f_i(u_k) = u_l$ occurs in the semi-base formula written in selector language. A *constant node* is a node labelled by some constant symbol $c \in \Sigma$, $\mathsf{ar}(c) = 0$. A constant node is a sink in the graph; every sink is either a constant or a parameter node. In addition to the labelling by function symbols, each node $u \in U$ of the graph is labelled by zero or more free variables x such that equation $x = u$ occurs in the semi-base formula.

Definition 21 (Base Formula) *A semi-base formula ϕ is a* base formula *iff the graph associated with ϕ is acyclic.*

A semi-base formula whose associated graph is cyclic is unsatisfiable in the term algebra of finite terms. Checking the cyclicity of a base formula corresponds to occur-check in unification algorithms (see e.g. [29, 11]).

Definition 22 *By* height $\mathcal{H}(u)$ *of a node u in the acyclic graph we mean the length of the longest path starting from u.*

A node u is sink iff $\mathcal{H}(u) = 0$.

Definition 23 *We say that an internal variable u_l is a* source variable *of a base formula β iff u_l is represented by a node that is source in the directed acyclic graph corresponding to β. Equivalently, if β is written in the selector language, then u_l is a source variable iff β contains no equations of form $u_l = f_i(u_k)$.*

Definition 24 *If u_i and u_j are internal variables, we write $u_i \rightarrowtail^* u_j$ if there is a path in the underlying graph from node u_i to node u_j. Equivalently, $u_i \rightarrowtail^* u_j$ iff there exists a term $t(u_i)$ in the selector language such that $\models \beta \Rightarrow u_j = t(u_i)$.*

Relation $\rightarrowtail^*$ is a partial order on internal variables of β.

The following Lemma 25 is similar to the Independence of Disequations Lemma in e.g. [10, Page 178].

Lemma 25 *Let β be a base formula of the form*

$$\exists u_1, \ldots, u_p, u_{p+1}, \ldots, u_{p+q}.\ \beta_0$$

where $u_{p+1}, \ldots, u_{p+q}$ are parameter variables of β, and β_0 is quantifier-free. Let $S_{p+1}, \ldots, S_{p+q}$ be infinite sets of terms. Then there exists a valuation σ such that $[\![\beta_0]\!]\sigma = \mathsf{true}$ and $[\![u_i]\!]\sigma \in S_i$ for $p+1 \le i \le p+q$.

Proof. To construct σ assign first the values to parameter variables, as follows. Let h_G be the length of the longest path in the graph associated with β. Pick $\sigma(u_{p+1}) \in S_{p+1}$ so that $\mathsf{h}(\sigma(u_{p+1})) > h_G$, and for each i where $p+2 \le i \le p+q$ pick $\sigma(u_i) \in S_i$ so that $\mathsf{h}(\sigma(u_i)) > \mathsf{h}(\sigma(u_{i-1})) + h_G$. The set of heights of an infinite set of terms is infinite, so it is always possible to choose such $\sigma(u_i)$.

Next consider internal nodes $u_1, \ldots, u_{p+q}$ in some topological order. For each non-parameter node u_i such that $u_i = f(u_{l_1}, \ldots, u_{l_k})$ occurs in β_0, let $\sigma(u_i) = f(\sigma(u_{l_i}), \ldots, \sigma(u_{l_k}))$.

Finally assign the values to free variables by $\sigma(x) = \sigma(u)$ where $x = u$ occurs in β_0.

By construction, $[\![\mathsf{structure}]\!]\sigma = \mathsf{true}$ and $[\![\mathsf{labels}]\!]\sigma = \mathsf{true}$. It remains to show $[\![\mathsf{distinct}]\!]\sigma = \mathsf{true}$ i.e. $\sigma(u_i) \neq \sigma(u_j)$ for $1 \le i, j \le p+q$, $i \neq j$. We show this property of σ by induction on $m = \min(\mathcal{H}(u_i), \mathcal{H}(u_j))$. Without loss of generality we assume $\mathcal{H}(u_i) \le \mathcal{H}(u_j)$.

Consider first the case $m = 0$. Then u_i is a parameter or a constant node.

If u_i is a constant and u_j is a non-parameter variable then u_i and u_j are labelled by different function symbols so $\sigma(u_i) \neq \sigma(u_j)$.

If u_i is a constant and u_j is a parameter variable then $\mathsf{h}(\sigma(u_i)) = 0$ whereas $\mathsf{h}(\sigma(u_j)) > h_G \ge 0$.

Consider the case where u_i is a parameter variable and u_j is a non-parameter variable. Let

$$J = \{j_1 \mid u_{j_1} \text{ is a parameter variable s.t. } u_j \rightarrowtail^* u_{j_1}\}$$

If $J = \emptyset$, then β_0 uniquely specifies $\sigma(u_j)$, and

$$\mathsf{h}(\sigma(u_j)) = \mathcal{H}(u_j) \le h_G < \mathsf{h}(\sigma(u_i))$$

Let $J \neq \emptyset$ and $j_0 = \max J$. If $i \le j_0$, then

$$\mathsf{h}(\sigma(u_i)) \le \mathsf{h}(\sigma(u_{j_0})) < \mathsf{h}(\sigma(u_j))$$

If $j_0 < i$ then

$$\mathsf{h}(\sigma(u_j)) \le \mathsf{h}(\sigma(u_{j_0})) + h_G < \mathsf{h}(\sigma(u_{j_0+1})) \le \mathsf{h}(\sigma(u_i))$$

Now consider the case $m > 0$. u_i and u_j are non-parameter nodes, so let $u_i = f(u_{i_1}, \ldots, u_{i_k})$ and $u_j = g(u_{j_1}, \ldots, u_{j_l})$. If $f \neq g$ then clearly $\sigma(u_i) \neq \sigma(u_j)$. Otherwise, by congruence closure property of base formulas, there exists d such that $u_{i_d} \neq u_{j_d}$. Then by induction hypothesis $\sigma(u_{i_d}) \neq \sigma(u_{j_d})$, so $\sigma(u_i) \neq \sigma(u_j)$. ■

Corollary 26 *Every base formula is satisfiable.*

Proposition 27 (Quantification of Base Formula) *If β is a base formula and x a free variable in β, then there exists a base formula β_1 equivalent to $\exists x.\beta$.*

Proof. Consider a formula $\exists x.\beta$ where β is a base formula. The only place where x occurs in β is $x = u_{s_1}$ in the subformula labels. By dropping the conjunct $x = u_{s_1}$ from β we obtain a base formula β_1 where β_1 is equivalent to $\exists x.\beta$. ■

Proposition 28 (Quantifier-Free to Base) *Every well-defined quantifier-free formula in constructor-selector language can be written as* true, false, *or a disjunction of base formulas.*

Proof Sketch. Let ϕ be a well-defined quantifier-free formula in constructor-selector language. By Proposition 8 we can transform ϕ into an equivalent formula in disjunctive normal form

$$\psi_1 \vee \cdots \vee \psi_p$$

where each ψ_i is a well-defined conjunction of literals. Consider an arbitrary ψ_i. There exists an unnested quantifier-free formula ψ_i' with additional fresh free variables $x_1, \ldots, x_q$ such that ψ_i is equivalent to

$$\exists x_1, \ldots, x_q.\ \psi_i'$$

By distributivity and (6) it suffices to transform each conjunction of unnested formulas into disjunction of base formulas. In the sequel we will assume transformations based on distributivity and (6) are applied whenever we transform conjunction of literals into a formula containing disjunction. We also assume that every equation $f(x_1, \ldots, x_n) = y$ is replaced by the equivalent one $y = f(x_1, \ldots, x_n)$ and every equation $f_i(x) = y$ is replace by $y = f_i(x)$.

Because of our assumption that Σ is finite, we can eliminate every literal of form $\neg\mathsf{Is}_f(x)$ using the equivalence

$$\neg\mathsf{Is}_f(x) \iff \bigvee_{g \in \Sigma\setminus\{f\}} \mathsf{Is}_g(x) \tag{17}$$

which follows from (16). We then transform formula back into disjunctive normal form and propagate the existential quantifiers to the conjunctions of literals. We may therefore assume that there are no literals of form $\neg\mathsf{Is}_f(x)$ in the conjunction. Furthermore, $\mathsf{Is}_f(x) \wedge \mathsf{Is}_g(x) \iff \mathsf{false}$ for $f \not\equiv g$, so we may assume that for variable x there is at most one literal $\mathsf{Is}_f(x)$ for some f. If $f_i(x)$ occurs in the conjunction, because the conjunction is well-defined, we may always add the conjunct $\mathsf{Is}_f(x)$. This way we ensure that exactly one literal of form $\mathsf{Is}_f(x)$ occurs in the conjunction.

We next ensure that every variable has either none or all of its components named by variables. If the conjunction contains literal $\mathsf{Is}_f(x)$ but does not contain $x = f(x_1, \ldots, x_n)$ and does not contain an equation of form $y = f_i(x)$ for every i, $1 \le i \le \mathsf{ar}(f)$, we introduce a fresh existentially quantified variable for each i such that a term of form $y = f_i(x)$ does not appear in the conjunction. At this point we may transform the entire conjunction into constructor language by replacing

$$\mathsf{Is}_f(u_i)\ \wedge\ v_{l_1} = f_1(u_i)\ \wedge \cdots \wedge\ v_{l_k} = f_k(u_i)$$

with $u_i = f(v_{l_1}, \ldots, v_{l_k})$ for $k = \mathsf{ar}(f)$.

We next ensure that for every two variables x_1 and x_2 occurring in the conjunction exactly one of the conjunct $x_1 = x_2$ or $x_1 \neq x_2$ is present. Namely if both conjuncts $x_1 = x_2$ and $x_1 \neq x_2$ are present, the conjunction is false. If none of the conjuncts is present, we insert the disjunction

$x_1 = x_2 \vee x_1 \neq x_2$ as one of the conjuncts and transform the result into disjunction of existentially quantified conjunctions.

We next perform congruence closure for finite terms [38] on the resulting conjunction, using the fact that equality is reflexive, symmetric, transitive and congruent with respect to free operations $f \in \mathsf{Cons}(\Sigma)$ and that $t(x) \neq x$ for every term $t \not\equiv x$. Syntactically, the result of congruence closure can be viewed as adding new equations to the conjunction. If the congruence closure procedure establishes that the formula is unsatisfiable, the result is false. Otherwise, all variables are grouped into equivalence classes. If a $u_1 = u_2$ occurs in the conjunction where both u_1 and u_2 are internal variables, we replace u_1 with u_2 in the formula and eliminate the existential quantifier. If for some free variable x there is no internal variable u such that conjunction $x = u$ occurs, we introduce a new existentially quantified variable and a conjunct $x = u$. These transformations ensure that for every equivalence class there exists exactly one internal variable in the formula. It is now easy to pick representative conjuncts from the conjunction to obtain conjunction of the syntactic form in Definition 19 of semi-base formula. The resulting formula is a base formula because congruence closure algorithm ensures that the associated graph is acyclic. ■

We next turn to the problem of transforming a base formula into a quantifier-free formula. We will present two constructions. The first construction yields a quantifier-free formula in *constructor-selector language* and is sufficient for the purpose of quantifier elimination. The second construction yields a quantifier-free formula in *selector language* and is slightly more involved; we present it to provide additional insight into the quantifier elimination approach to term algebras.

We first introduce notions of *covered* and *determined* variables of a base formula β. The basic idea behind these notions is that β implies a functional dependence from the free variables of β to each of the determined variables.

In both constructions we use the notion of a a *covered variable*, which denotes a component of a term denoted by some free variable. In the first construction we also use the notion of *determined variable*, which includes covered variables as well as variables constructed from covered variables using constructor operations $f \in \mathsf{Cons}(\Sigma)$.

Definition 29 *Consider an arbitrary base formula β. We say that an internal variable u is* covered by a free variable x *iff $x = u'$ occurs in β for some u' such that $u \rightarrowtail^* u'$. An internal variable u is* covered *iff u is covered by x for some free variable x (in particular, if $x = u$ occurs in β then u is covered). Let* $\mathsf{covered}$ *denote the set of covered internal variables of base formula, and let* $\mathsf{uncovered} = U \setminus \mathsf{covered}$ *where U is the set of all internal variables of β.*

Lemma 30 (Covered Base to Selector) *Every base formula without uncovered variables is equivalent to a quantifier free formula in selector language.*

Proof. Consider a base formula β where every variable is covered. Consider an arbitrary quantified variable u. Because u is covered, there exists variable x free in β such that $u = t(x)$ for some term t in the selector language. Replace every occurrence of u in the matrix of β by $t(x)$ and eliminate the quantification over u. Repeating this process for every variable u we obtain a quantifier-free formula equivalent to β. ■

Definition 31 *Let β be a base formula. The set* $\mathsf{determined}$ *of determined variables of β is the smallest set S that contains the set* $\mathsf{covered}$ *and satisfies the following condition: if u is a non-parameter node and all successors $u_1, \ldots, u_k$ ($k \geq 0$) of u in the associated graph are in S, then u is also in S.*

In particular, every constant node is determined. A parameter node w is determined iff w is covered.

Lemma 32 *If a node u is not determined, then there exists an uncovered parameter node v such that $u \rightarrowtail^* v$.*

Proof. The proof is by induction on $\mathcal{H}(()u)$. If $\mathcal{H}(()u) = 0$ then u has no successors, and u cannot be a constant node because it is not determined. Therefore, u is a parameter node, so we may let $v \equiv u$. Assume that the statement holds for for every node u' such that $\mathcal{H}(()u') = k$ and let $\mathcal{H}(()u) = k + 1$. Because u is not determined, there exists a successor u' of u such that u' is not determined, so by induction hypothesis there exists an uncovered parameter node v such that $u' \rightarrowtail^* v$. Hence $u \rightarrowtail^* u' \rightarrowtail^* v$. ■

Lemma 33 *Every base formula β is equivalent to a base formula β' obtained from β by eliminating all nodes that are not determined.*

Proof. Construct β' from β by eliminating all terms containing a variable $u \in U \setminus \mathsf{determined}$ and eliminating the corresponding existential quantifiers. Then all variables in β' are determined. β' has fewer conjuncts than β, so $\models \beta \Rightarrow \beta'$. To show $\models \beta' \Rightarrow \beta$, let σ be any assignment of terms to determined variables of β such that β evaluate to true under σ. As in the proof of Lemma 25, define the extension σ' of σ as follows. Choose sufficiently large values $\sigma'(v)$ for every uncovered sink variable v, so that σ'' defined as the unique extension of σ' to the remaining undetermined variables assigns different terms to different variables. This is possible because the term model is infinite. The resulting assignment σ'' satisfies the matrix of the base formula β. Therefore, $\models \beta' \Rightarrow \beta$, so β and β' are equivalent base formulas. ■

First Construction

Proposition 34 (Base to Constructor-Selector) *Every base formula β is equivalent to a quantifier-free formula ϕ in constructor-selector language.*

Proof. By Lemma 33 we may assume that all variables in β are determined. To every variable u we assign a term $\tau(u)$. Term $\tau(u)$ is in constructor-selector language and the variables of $\tau(u)$ are among the free variables of β. If $u \in \mathsf{covered}$, we assign $\tau(u)$ as in the proof of Lemma 30. If $u_1, \ldots, u_k$ are the successors of a determined node u, we put

$$\tau(u) = f(\tau(u_1), \ldots, \tau(u_k))$$

where f is the label of node u. This definition uniquely determines $\tau(u)$ for all $u \in \mathsf{determined}$. We obtain the quantifier-free formula ϕ by replacing every variable u with $\tau(u)$ and eliminating all quantifiers.

For every u we have $\models \beta \Rightarrow u = \tau(u)$, so $\models \beta \Rightarrow \phi$. Conversely, if ϕ is satisfied then τ defines an assignment for u variables which makes the matrix of β true. Therefore β and ϕ are equivalent. ■

Second Construction The reason for using constructor symbols $f \in \mathsf{Cons}(\Sigma)$ in the first construction is to preserve the constraints of form $u \neq v$ when eliminating node u with successors $u_1, \ldots, u_k$. Using constructor symbols we would obtain the constraint $f(u_1, \ldots, u_k) \neq v$. Our second construction avoids introducing constructor operations by decomposing $f(u_1, \ldots, u_k) \neq v$ into disjunction of inequalities of form $u_i \neq f_i(v)$. When v is a parameter node, the presence of term $f_i(v)$ potentially requires introducing a new node in the associated graph, we call this process *parameter expansion.* Parameter expansion may increase the total number of nodes in the graph, but it decreases the number of uncovered nodes, so the process of converting a base formula to a quantifier-free formula in the selector language terminates.

Lemma 35 *Let β be an arbitrary base formula.*

1. *If u is covered and $u \rightarrowtail^* u'$ then u' is covered as well.*
2. *If u' is uncovered and u' is not a source, then there exists $u \not\equiv u'$ such that $u \rightarrowtail^* u'$ and u is also uncovered.*
3. *If β contains an uncovered variable then β contains an uncovered variable that is a source.*

Proof. By definition. ■

Parameter Expansion We define the operation of expanding a parameter node in a base formula as follows. Let β be an arbitrary base formula and w a parameter variable in β. The result of expansion of w is a disjunction of base formulas β' generated by applying (13) to w. In each of the resulting formulas β' variable w is not a parameter any more. Each β' contains $\mathsf{Is}_f(w)$ for some $f \in \Sigma$ and node w has successors $u_1, \ldots, u_k$ for $k = \mathsf{ar}(f)$. Each successor u_i is either an existing internal variable or a fresh variable. For a given β, sink expansion generates disjunction of formulas β' for every choice of $f \in \Sigma$ and every choice of successors u_i, subject to congruence closure so that β' is a base formula: we discard the choices of successors of w that yield formulas β' violating congruence of equality. (This process is similar to converting quantifier-free formulas into disjunction of base formulas in the proof of Proposition 28.) The following lemma shows the correctness of parameter expansion.

Lemma 36 (Parameter expansion soundness) *Let $\Delta = \beta_1' \vee \cdots \beta_k'$ be the disjunction generated by parameter expansion of a base formula β. Then Δ is equivalent to β.*

Lemma 36 justifies the use of parameter expansion in the following Lemma 37.

Lemma 37 *Every base formula β can be written as a disjunction of base formulas without uncovered variables.*

Proof Sketch. By Lemma 33 we may assume that all variables of β are determined. Suppose β contains an uncovered variable. Then by Lemma 35, β contains an uncovered variable u_0 such that u_0 is a source. Because u_0 is uncovered and determined, it is not a parameter node. We show how to eliminate u_0 without introducing new uncovered variables.

Our goal is to eliminate u_0 from the associated graph. We need to preserve information that u_0 is distinct from variables $u \in U \setminus \{u_0\}$ in the graph. We consider two cases.

If u is not a parameter node, then by congruence closure either u_0 and u are labelled by different function symbols, or they are labelled by the same function symbol $f \in \Sigma$ with $\mathsf{ar}(f) = k$ and there exists i, $1 \leq i \leq k$ and variables $u_i = f_i(u_0)$ and $u_i' = f_i(u')$ such that $u_i \not\equiv u_i'$. Hence the constraint $u_0 \neq u$ is deducible from the inequalities of other variables in β and we can eliminate u_0 without changing the truth value of β.

Next consider the case when u is a parameter node. By assumption u is determined, and because it is parameter, it is covered. We then perform parameter node expansion as described above. The result of elimination of u_0 in β is a disjunction of base formulas β', in each β' every parameter node is expanded. If u is a parameter node in β then the constraint $u_0 \neq u$ is preserved in each β' because u is not a parameter node in β' so the previous argument applies.

Because the parameter nodes being expanded are covered, so are their successor nodes introduced by parameter expansion. Therefore, by repeatedly applying elimination of uncovered variables for every uncovered variable u_0, we obtain a disjunction Δ of formulas β' where each β' has no uncovered variables, and Δ is equivalent to β. ■

Proposition 38 (Base to Selector) *For every base formula β there exists an equivalent quantifier-free formula ψ in selector language.*

Proof. By Lemma 37, β is equivalent to a disjunction $\beta_1 \vee \cdots \vee \beta_n$ where each β_i has no uncovered variables. By Lemma 30, each β_i is equivalent to some quantifier free formula ψ_i, so β is equivalent to the quantifier-free formula $\psi_1 \vee \cdots \vee \psi_n$. ■

The final theorem in this section summarizes quantifier elimination for term algebra.

Theorem 39 (Term Algebra Quantifier Elimination) *There exist algorithms A, B, C such that for a given formula ϕ in constructor-selector language of term algebras:*

a) *A produces a quantifier-free formula ϕ' in constructor-selector language*

b) *B produces a quantifier-free formula ϕ' in selector language*

c) *C produces a disjunction ϕ' of base formulas*

Proof. *a*): Transform formula ϕ into prenex form

$$Q_1 x_1 \ldots Q_{n-1} x_{n-1} Q_n x_n . \phi^*$$

where ϕ^* is quantifier free, as in Section 3.1. We eliminate the innermost quantifier Q_n as follows.

Suppose first that Q_n is $\exists$. Transform the matrix ϕ^* into disjunctive normal form $C_1 \vee \cdots \vee C_n$. By Proposition 28, transform $C_1 \vee \cdots \vee C_n$ into disjunction $\beta_1 \vee \cdots \vee \beta_m$ of base formulas. Then propagate $\exists$ into individual disjuncts, using

$$\exists x_n .\ \beta_1 \vee \cdots \vee \beta_m \iff (\exists x_n . \beta_1) \vee \cdots \vee (\exists x_n . \beta_m)$$

By Proposition 27, an existentially quantified base formula is again a base formula, so $\exists x_n.\beta_i \iff \beta'_i$ for some β'_i. We thus obtain the

$$Q_1 x_1 \dots Q_{n-1} x_{n-1}.\ \beta'_1 \vee \dots \vee \beta'_m \tag{18}$$

By Proposition 34, every base formula is equivalent to a quantifier-free formula in selector language, so 18 is equivalent to

$$Q_1 x_1 \dots Q_{n-1} x_{n-1}.\psi$$

where ψ is a quantifier free formula. Hence, we have eliminated the innermost existential quantifier.

Next consider the case when Q_n is $\forall$. Then ϕ is equivalent to

$$Q_1 x_1 \dots Q_{n-1} x_{n-1} \neg \exists x_n. \neg \phi^*$$

Apply the procedure for eliminating x_n to $\neg\phi^*$. The result is formula of form

$$Q_1 x_1 \dots Q_{n-1} x_{n-1}.\ \neg\psi \tag{19}$$

where ψ is quantifier free. But $\neg\psi$ is also quantifier free, so we have eliminated the innermost universal quantifier. By repeating this process we eliminate all quantifiers, yielding the desired formula ϕ'.

The direct construction for showing b) is analogous to a), but uses Proposition 38 in place of Proposition 34. To show c), apply e.g. construction a) to obtain a quantifier-free formula ψ and then transform ψ into disjunction of base formulas using Proposition 28. ∎

This completes our description of quantifier elimination for term algebras.

We remark that there are alternative ways to define base formula. In particular the requirement on disequality of all variables is not necessary. This requirement may lead to unnecessary case analysis when converting a quantifier-free formula to disjunction of base formulas, but we believe that it simplifies the correctness argument.

4 The Pair Constructor and Two Constants

In this section we give a quantifier elimination procedure for structural subtyping of non-recursive types with two constant symbols and one covariant binary constructor. Two constants corresponds to two primitive types; one binary covariant constructor corresponds to the pair constructor for building products of types.

The construction in this section is an introduction to the more general construction in Section 5, where we give a quantifier elimination procedure for any number of constant symbols and relations between them. The construction in this section demonstrates the interaction between the term and boolean algebra components of the structural subtyping. We therefore believe the construction captures the essence of the general result of Section 5.

The basic observation behind the quantifier elimination procedure for two constant symbols is that the structure of terms in this language is isomorphic to a disjoint union of boolean algebras with some additional term structure connecting elements from different boolean algebras. As we argue below, the structural subtyping structure contains one copy of boolean algebra for every equivalence class of terms that have the same "shape" i.e. are same up to the constants in the leaves.

Consider a signature $\Sigma = \{a, b, g\}$ where a and b are constant symbols and g is a function symbol of arity 2. We define a partial order $\leq$ on the set $\mathsf{FT}(\Sigma)$ of ground terms over Σ as the least reflexive partial order relation ρ satisfying

1. $a \,\rho\, b$;
2. $(s_1 \,\rho\, t_1) \wedge (s_2 \,\rho\, t_2) \Rightarrow g(s_1, s_2) \,\rho\, g(t_1, t_2)$.

The structure with equality in the language $\{a, b, g, \leq\}$, where $\leq$ is interpreted as above and a, b, g are interpreted as free operations on term algebra corresponds to the structural subtyping with two base types a and b and one binary type constructor g, with g covariant in both arguments. We denote this structure by BS. We proceed to show that BS admits quantifier elimination and is therefore decidable.

4.1 Boolean Algebras on Equivalent Terms

In preparation for the quantifier elimination procedure we define certain operations and relations on terms. We also establish some fundamental properties of the structure BS.

Define a new signature $\Sigma_0 = \{c^{\mathsf{s}}, g^{\mathsf{s}}\}$ as an abstraction of signature $\Sigma = \{a, b, g\}$. Define function $\mathsf{shapified} : \Sigma \to \Sigma_0$ by

$$\mathsf{shapified}(a) = c^{\mathsf{s}}$$
$$\mathsf{shapified}(b) = c^{\mathsf{s}}$$
$$\mathsf{shapified}(g) = g^{\mathsf{s}}$$

Let $\mathsf{ar}(\mathsf{shapified}(f)) = \mathsf{ar}(f)$ for each $f \in \Sigma$; in this case c^{s} is a constant and g^{s} is a binary function symbol. Let $\mathsf{FT}(\Sigma_0)$ be the set of ground terms over the signature Σ_0. Define *shape* of a term t, as the function $\mathsf{sh} : \mathsf{FT}(\Sigma) \to \mathsf{FT}(\Sigma_0)$, by letting

$$\mathsf{sh}(f(t_1, \dots, t_k)) = \mathsf{shapified}(f)(\mathsf{sh}(t_1), \dots, \mathsf{sh}(t_k))$$

for $k = \mathsf{ar}(f)$. In this case we have

$$\begin{aligned} \mathsf{sh}(a) &= c^{\mathsf{s}} \\ \mathsf{sh}(b) &= c^{\mathsf{s}} \\ \mathsf{sh}(g(t_1, t_2)) &= g^{\mathsf{s}}(\mathsf{sh}(t_1), \mathsf{sh}(t_2)) \end{aligned}$$

Define $t_1 \sim t_2$ iff $\mathsf{sh}(t_1) = \mathsf{sh}(t_2)$. Then $\sim$ is the smallest equivalence relation ρ such that

1. $a \,\rho\, b$;
2. $(s_1 \,\rho\, t_1) \wedge (s_2 \,\rho\, t_2) \Rightarrow g(s_1, s_2) \,\rho\, g(t_1, t_2)$.

For every term t define the word $\mathsf{tCont}(t) \in \{0, 1\}^*$ by letting

$$\begin{aligned} \mathsf{tCont}(a) &= 0 \\ \mathsf{tCont}(b) &= 1 \\ \mathsf{tCont}(f(t_1, t_2)) &= \mathsf{tCont}(t_1) \cdot \mathsf{tCont}(t_2) \end{aligned}$$

The set of all words $w \in \{0, 1\}^n$ is isomorphic the boolean algebra of B_n of all subsets of some finite sets of cardinality n, so we write $w_1 \cap w_2$, $w_1 \cup w_2$, w^c for operations corresponding to intersection, union, and set complement in the set of words $w \in \{0, 1\}^n$. We write $w_1 \subseteq w_2$ for $w_1 \cap w_2 = w_1$.

Define function δ by

$$\delta(t) = \langle \mathsf{sh}(t), \mathsf{tCont}(t) \rangle$$

For term t in any language containing constant symbols, let $\mathsf{tLen}(t)$ denote the number of occurrences of constant symbols in t. If w is a sequence of elements of some set, let $\mathsf{sLen}(w)$ denote the length of the sequence. Observe that $\mathsf{sLen}(\mathsf{tCont}(t)) = \mathsf{tLen}(t)$ and $\mathsf{tLen}(\mathsf{sh}(t)) = \mathsf{tLen}(t)$. Moreover, $t_1 \sim t_2$ implies $\mathsf{sLen}(\mathsf{tCont}(t_1)) = \mathsf{sLen}(\mathsf{tCont}(t_2))$. Define the set B by

$$B = \{\langle s, w\rangle \mid s \in \mathsf{FT}(\Sigma_0), w \in \{0,1\}^*, \mathsf{tLen}(s) = \mathsf{sLen}(w)\}$$

Function δ is a bijection from the set $\mathsf{FT}(\Sigma)$ to the set B. For $b_1, b_2 \in B$ define $b_1 \leq b_2$ iff $\delta^{-1}(b_1) \leq \delta^{-1}(b_2)$. From the definitions it follows

$$\langle s_1, w_1\rangle \leq \langle s_2, w_2\rangle \iff s_1 = s_2 \wedge w_1 \subseteq w_2$$

If g is defined on B via isomorphism δ we also have

$$g(\langle s_1, w_1\rangle, \langle s_2, w_2\rangle) = \langle g^{\mathsf{s}}(s_1, s_2), w_1 \cdot w_2\rangle$$

For any fixed $s \in \mathsf{FT}(\Sigma_0)$, the set

$$B(s_0) = \{\langle s, w\rangle \in B \mid s = s_0\} \tag{20}$$

is isomorphic to the boolean algebra B_n, where $n = \mathsf{tLen}(s)$. Accordingly, we introduce on each $B(s)$ the set operations $t_1 \cap_s t_2$, $t_1 \cup_s t_2$, t_{1s}^c. Expressions $t_1 \cap_s t_2$ and $t_1 \cup_s t_2$ are defined iff $\mathsf{sh}(t_1) = s$ and $\mathsf{sh}(t_2) = s$, whereas expression t_{1s}^c is defined iff $\mathsf{sh}(t_1) = s$.

We also introduce cardinality expressions as in Section 3.2. If t denotes a term, then the expression $|t|_s$ denotes the number of elements of the set corresponding to t. Here we require $s = \mathsf{sh}(t)$. We use expressions $|t|_s = k$ and $|t|_s \geq k$ as atomic formulas for constant integer $k \geq 0$. Note that

$$t_1 \leq t_2 \iff \mathsf{sh}(t_1) = \mathsf{sh}(t_2) \ \wedge \ |t_1 \cap t_2^c|_{\mathsf{sh}(t_1)} = 0 \tag{21}$$

$$\begin{aligned} t_1 = t_2 \iff \ & \mathsf{sh}(t_1) = \mathsf{sh}(t_2) \ \wedge \\ & |(t_1 \cap t_2^c) \cup (t_1^c \cap t_2)|_{\mathsf{sh}(t_1)} = 0 \end{aligned} \tag{22}$$

Let $\mathsf{sh}(t_1) = s_1$, $\mathsf{sh}(t_2) = s_2$, and $s = g^{\mathsf{s}}(s_1, s_2)$. Then

$$|g(t_1, t_2)|_s = |t_1|_{s_1} + |t_2|_{s_2} \tag{23}$$

Equation 23 allows decomposing formulas of form $|g(t_1, t_2)|_s \geq k$ into propositional combinations of formulas of form $|t_1|_{s_1} \geq k$ and $|t_2|_{s_2} \geq k$.

Note further that the following equations hold:

$$\begin{aligned} g(t_1, t_2) \cap g(t_1', t_2') &= g(t_1 \cap t_1', t_2 \cap t_2') \\ g(t_1, t_2) \cup g(t_1', t_2') &= g(t_1 \cup t_1', t_2 \cup t_2') \\ g(t_1, t_2)^c &= g(t_1^c, t_2^c) \end{aligned}$$

If $E(x_1, \ldots, x_n)$ denotes an expression consisting only of operations of boolean algebra, then from (4.1) by induction follows that

$$E(g(t_1^1, t_1^2), \ldots, g(t_n^1, t_n^2)) = g(E(t_1^1, \ldots, t_n^1), E(t_1^2, \ldots, t_n^2)) \tag{24}$$

Equations (24) and (23) imply

$$|E(g(t_1^1, t_1^2), \ldots, g(t_n^1, t_n^2))| = |E(t_1^1, \ldots, t_n^1)| + |E(t_1^2, \ldots, t_n^2)| \tag{25}$$

Boolean algebra $B(g^{\mathsf{s}}(s_1, s_2))$ is isomorphic to the product of boolean algebras $B(s_1)$ and $B(s_2)$; the constructor g acts as union of disjoint sets.

a, b	::	term
g	::	term $\times$ term $\to$ term
Is_g	::	term $\to$ bool
g_1, g_2	::	term $\to$ term
$=$	::	term $\times$ term $\to$ bool
$\leq$	::	term $\times$ term $\to$ bool
c^{s}	::	shape
g^{s}	::	shape $\times$ shape $\to$ shape
$\mathsf{Is}_{g^{\mathsf{s}}}$	::	shape $\to$ bool
$g_1^{\mathsf{s}}, g_2^{\mathsf{s}}$	::	shape $\to$ shape
sh	::	term $\to$ shape
$=^{\mathsf{s}}$	::	shape $\times$ shape $\to$ bool
$\cap_{\_}, \cup_{\_}$	::	shape $\times$ term $\times$ term $\to$ term
$\_^c_{\_}$	::	shape $\times$ term $\to$ term
$1_{\_}, 0_{\_}$	::	shape $\to$ term
$\lvert\_\rvert_{\_} \geq k$, $\lvert\_\rvert_{\_} = k$	::	shape $\times$ term $\to$ bool

Figure 5: Operations and relations in structure FT_2

4.2 A Multisorted Logic

To show the decidability of structure BS, we give a quantifier elimination procedure for an extended structure, denoted FT_2. We use a first-order two-sorted logic with sorts term and shape interpreted over FT_2.

The domain of structure FT_2 is $\mathsf{FT}(\Sigma) \cup \mathsf{FT}(\Sigma_0)$ with elements $\mathsf{FT}(\Sigma)$ having sort term and elements $\mathsf{FT}(\Sigma_0)$ having sort shape. Variables in Var have term sort, variables in $\mathsf{Var}^{\mathsf{s}}$ have shape sort. In general, if t denotes an element of FT_2, we write t^S to indicate that the element has sort shape.

Figure 5 shows operations and relations in FT_2 with their sort declarations. The signature is infinite because operations $|t|_s \geq k$ and $|t|_s = k$ are parameterized by a nonnegative integer k.

We require all terms to be well-sorted. Functions g_1 and g_2 are interpreted as partial selector functions in the term constructor-selector language, so $D_{g_1} = D_{g_2} = \langle\langle x\rangle, \mathsf{Is}_g(x)\rangle$. Similarly, g_1^{s} and g_2^{s} are partial selector functions in the shape constructor-selector language, so $D_{g_1^{\mathsf{s}}} = D_{g_2^{\mathsf{s}}} = \langle\langle x\rangle, \mathsf{Is}_{g^{\mathsf{s}}}(x)\rangle$. The expressions $t_1 \cap_s t_2$ and $t_1 \cup_s t_2$ are defined iff $\mathsf{sh}(t_1) = \mathsf{sh}(t_2) = s$, and t_s^c is defined iff $\mathsf{sh}(t) = s$. We therefore let

$$\begin{aligned} & D_{\cap_s} = D_{\cup_s} = \\ & \quad \langle\langle y^{\mathsf{s}}, x_1, x_2\rangle, \mathsf{sh}(x_1) = y^{\mathsf{s}} \wedge \mathsf{sh}(x_2) = y^{\mathsf{s}}\rangle \end{aligned}$$

and

$$D_{\_^c_{\_}} = \langle\langle y^{\mathsf{s}}, x\rangle, \mathsf{sh}(x) = y^{\mathsf{s}}\rangle$$

For atomic formulas $|t|_s \geq k$ and $|t|_s = k$ we require atomic formula $\mathsf{sh}(t) = s$ to ensure well-definedness:

$$D_{|\_|\_=k} = D_{|\_|\_\geq k} = \langle\langle y^{\mathsf{s}}, x\rangle, \mathsf{sh}(x) = y^{\mathsf{s}}\rangle$$

Note that the language of Figure 5 subsumes the language $\{a, b, g, \leq\}$ for the structural subtyping structure. The quantifier-elimination procedure we present in Section 4.3 is therefore sufficient for quantifier elimination in the first-order logic interpreted over the structural subtyping structure FT_2.

4.3 Quantifier Elimination for Two Constants

We are now ready to present a quantifier elimination procedure for the structure FT_2. The quantifier elimination procedure is based on the quantifier elimination for term algebras of Section 3.4 as well as the quantifier elimination for boolean algebras of Section 3.2.

We first define an auxiliary notion of a u^{s}-term as a term formed starting from shape u^{s} term variables and shape u^{s} constants, using operations $\cap_{u^{\mathsf{s}}}$, $\cup_{u^{\mathsf{s}}}$, and $\_^{c}_{u^{\mathsf{s}}}$.

Definition 40 (u^{s}-terms) *Let $u^{\mathsf{s}} \in \mathsf{Var}^{\mathsf{s}}$ be a shape variable. The set of u^{s}-terms $\mathsf{Term}(u^{\mathsf{s}})$ is the least set such that:*

1. $\mathsf{Var} \subseteq \mathsf{Term}(u^{\mathsf{s}})$
2. $0_{u^{\mathsf{s}}}, 1_{u^{\mathsf{s}}} \in \mathsf{Term}(u^{\mathsf{s}})$
3. *if $t, t' \in \mathsf{Term}(u^{\mathsf{s}})$, then also*

$$t \cap_{u^{\mathsf{s}}} t' \in \mathsf{Term}(u^{\mathsf{s}}),$$
$$t \cup_{u^{\mathsf{s}}} t' \in \mathsf{Term}(u^{\mathsf{s}}), \text{ and}$$
$$t^{c}_{u^{\mathsf{s}}} \in \mathsf{Term}(u^{\mathsf{s}})$$

Similarly to base formulas of Section 3.4, we define *structural base formulas* for FT_2 structure. A structural base formula contains a copy of a base formula for the shape sort (shapeBase), a base formula for the term sort without term disequalities (termBase), a formula expressing mapping of term variables to shape variables (hom), and cardinality constraints on term parameter nodes of the term base formula (cardin).

Definition 41 (Structural Base Formula) A structural base formula *with:*

- *free term variables $x_1, \ldots, x_m$;*
- *internal non-parameter term variables $u_1, \ldots, u_p$;*
- *internal parameter term variables $u_{p+1}, \ldots, u_{p+q}$;*
- *free shape variables $x^{\mathsf{s}}_1, \ldots, x^{\mathsf{s}}_{m^{\mathsf{s}}}$;*
- *internal non-parameter shape variables $u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{p^{\mathsf{s}}}$;*
- *internal parameter shape variables $u^{\mathsf{s}}_{p^{\mathsf{s}}}, \ldots, u^{\mathsf{s}}_{p^{\mathsf{s}}+q^{\mathsf{s}}}$*

is a formula of form:

$$\begin{aligned}
&\exists u_1, \ldots, u_n, u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}.\\
&\quad \mathsf{shapeBase}(u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}, x^{\mathsf{s}}_1, \ldots, x^{\mathsf{s}}_{m^{\mathsf{s}}}) \wedge\\
&\quad \mathsf{termBase}(u_1, \ldots, u_n, x_1, \ldots, x_m) \wedge\\
&\quad \mathsf{hom}(u_1, \ldots, u_n, u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}) \wedge\\
&\quad \mathsf{cardin}(u_{p+1}, \ldots, u_n, u^{\mathsf{s}}_{p^{\mathsf{s}}+1}, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}})
\end{aligned}$$

where $n = p + q$, $n^{\mathsf{s}} = p^{\mathsf{s}} + q^{\mathsf{s}}$, and formulas shapeBase, termBase, hom, *and* cardin *are defined as follows.*

$$\begin{aligned}
&\mathsf{shapeBase}(u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}, x^{\mathsf{s}}_1, \ldots, x^{\mathsf{s}}_{m^{\mathsf{s}}}) =\\
&\quad \bigwedge_{i=1}^{p^{\mathsf{s}}} u^{\mathsf{s}}_i = t_i(u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_n) \;\wedge\; \bigwedge_{i=1}^{m^{\mathsf{s}}} x^{\mathsf{s}}_i = u^{\mathsf{s}}_{j_i}\\
&\quad \wedge\ \mathsf{distinct}(u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_n)
\end{aligned}$$

where each t_i is a shape term of form $f(u^{\mathsf{s}}_{i_1}, \ldots, u^{\mathsf{s}}_{i_k})$ for some $f \in \Sigma_0$, $k = \mathsf{ar}(f)$, and $j : \{1, \ldots, m^{\mathsf{s}}\} \to \{1, \ldots, n^{\mathsf{s}}\}$ is a function mapping indices of free shape variables to indices of internal shape variables.

$$\begin{aligned}
&\mathsf{termBase}(u_1, \ldots, u_n, x_1, \ldots, x_m) =\\
&\quad \bigwedge_{i=1}^{p} u_i = t_i(u_1, \ldots, u_n) \;\wedge\; \bigwedge_{i=1}^{m} x_i = u_{j_i}
\end{aligned}$$

where each t_i is a term of form $f(u_{i_1}, \ldots, u_{i_k})$ for some $f \in \Sigma$, $k = \mathsf{ar}(f)$, and $j : \{1, \ldots, m\} \to \{1, \ldots, n\}$ is a function mapping indices of free term variables to indices of internal term variables.

$$\mathsf{hom}(u_1, \ldots, u_n, u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}) = \bigwedge_{i=1}^{n} \mathsf{sh}(u_i) = u^{\mathsf{s}}_{j_i}$$

where $j : \{1, \ldots, n\} \to \{1, \ldots, n^{\mathsf{s}}\}$ is some function such that $\{j_1, \ldots, j_p\} \subseteq \{1, \ldots, p^{\mathsf{s}}\}$ and $\{j_{p+1}, \ldots, j_{p+q}\} \subseteq \{p^{\mathsf{s}} + 1, \ldots, p^{\mathsf{s}} + q^{\mathsf{s}}\}$ (a term variable is a parameter variable iff its shape is a parameter shape variable).

$$\mathsf{cardin}(u_{p+1}, \ldots, u_{p+q}, u^{\mathsf{s}}_{p^{\mathsf{s}}+1}, \ldots, u^{\mathsf{s}}_{p^{\mathsf{s}}+q^{\mathsf{s}}}) = \psi_1 \wedge \cdots \wedge \psi_r$$

where each ψ_i is of form

$$|t(u_{p+1}, \ldots, u_{p+q})|_{u^{\mathsf{s}}} = k$$

or

$$|t(u_{p+1}, \ldots, u_{p+q})|_{u^{\mathsf{s}}} \geq k$$

for some u^{s}-term $t(u_{p+1}, \ldots, u_{p+q})$ that contains no variables other than some of the variables $u_{p+1}, \ldots, u_{p+q}$, and the following condition holds:

> *If a variable u_{p+j} occurs in term $t(u_{p+1}, \ldots, u_{p+q})$, then $\mathsf{sh}(u_{p+j}) = u^{\mathsf{s}}$ occurs in formula* hom. (26)

We require each structural base formula to satisfy the following conditions:

P0) the graph associated with shape base formula

$$\exists u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}.\ \mathsf{shapeBase}(u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}, x^{\mathsf{s}}_1, \ldots, x^{\mathsf{s}}_{m^{\mathsf{s}}})$$

is acyclic (compare to Definition 21);

P1) congruence closure property for shapeBase *subformula: there are no two distinct variables u^{s}_i and u^{s}_j such that both $u^{\mathsf{s}}_i = f(u^{\mathsf{s}}_{l_1}, \ldots, u^{\mathsf{s}}_{l_k})$ and $u^{\mathsf{s}}_j = f(u^{\mathsf{s}}_{l_1}, \ldots, u^{\mathsf{s}}_{l_k})$ occur as conjuncts in formula* shapeBase*;*

P2) congruence closure property for termBase *subformula: there are no two distinct variables u_i and u_j such that both $u_i = f(u_{l_1}, \ldots, u_{l_k})$ and $u_j = f(u_{l_1}, \ldots, u_{l_k})$ occur as conjuncts in formula* termBase*;*

P3) homomorphism property of sh*: for every non-parameter term variable u such that $u = f(u_{i_1}, \ldots, u_{i_k})$ occurs in* termBase*, if the conjunct $\mathsf{sh}(u) = u^{\mathsf{s}}$ occurs in* hom*, then for some shape variables $u^{\mathsf{s}}_{j_1}, \ldots, u^{\mathsf{s}}_{j_k}$ the term $u^{\mathsf{s}} = f^{\mathsf{s}}(u^{\mathsf{s}}_{j_1}, \ldots, u^{\mathsf{s}}_{j_k})$ occurs in* shapeBase *where $f^{\mathsf{s}} = \mathsf{shapified}(f)$ and for every r where $1 \leq r \leq k$, conjunct $\mathsf{sh}(u_{i_r}) = u^{\mathsf{s}}_{j_r}$ occurs in* hom.

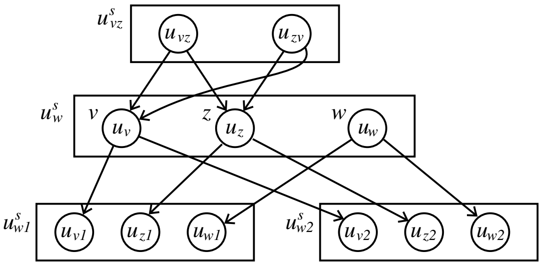

Figure 6: One of the Base Formulas Resulting from (28)

According to Definition 41 a structural base formula contains no selector function symbols. Formulation using selector symbols is also possible, as in Definition 20. The only partial function symbols occurring in a structural base formula of Definition 41 are in cardin subformula. Condition (26) therefore ensures that functions in cardin and thus the entire base formula are well-defined.

Note that acyclicity of shape base formula $\mathsf{shapeBase}$ (condition P0) implies acyclicity of term base formula as well. Namely, condition P3 ensures that any cycle in $\mathsf{termBase}$ implies a cycle in $\mathsf{shapeBase}$.

As in Section 3.4 we proceed to show that each quantifier-free formula can be written as a disjunction of base formulas and each base formula can be written as a quantifier-free formula.

We strongly encourage the reader to study the following example because it illustrates the idea behind our quantifier-elimination decision procedure.

Example 42 The following sentence is true in structure FT_2.

$$\begin{aligned}
&\forall x, y.\ x \leq y \Rightarrow \\
&\quad \exists z.\ z \leq x \wedge z \leq y \wedge \\
&\quad\quad \forall w.\ w \leq x \wedge w \leq y \Rightarrow \\
&\quad\quad\quad \forall v.\ g(v,z) \leq g(z,v) \wedge \mathsf{ls}_g(v) \wedge \mathsf{ls}_g(w) \Rightarrow g_1(w) \leq g_1(v)
\end{aligned} \tag{27}$$

An informal proof of sentence (27) is as follows. Suppose that $x \leq y$. Then $\mathsf{sh}(x) = \mathsf{sh}(y) = x^{\mathsf{s}}$. Let $z = x \cap_{x^{\mathsf{s}}} y$. Now consider some w such that $w \leq x$ and $w \leq y$. Then $\mathsf{sh}(w) = x^{\mathsf{s}}$, so $w \leq z$. Suppose that v is such that $g(v,z) \leq g(z,v)$. Then by covariance of g we have $z \leq v$, so $w \leq v$. If we assume $\mathsf{ls}_g(w)$ and $\mathsf{ls}_g(v)$, then $g_1(w)$ and $g_1(v)$ are well defined and by covariance of g we conclude $g_1(w) \leq g_1(v)$, as desired.

We now give an alternative argument that shows that sentence (27) is true. This alternative argument illustrates the idea behind our quantifier-elimination decision procedure. For the sake of brevity we perform some additional simplifications along the way that are not part of the procedure we present (although they could be incorporated to improve efficiency), and we skip consideration of some uninteresting cases during the case analyses.

Let us first eliminate the quantifier from formula

$$\forall v.\ g(v,z) \leq g(z,v) \wedge \mathsf{ls}_g(v) \wedge \mathsf{ls}_g(w) \Rightarrow g_1(w) \leq g_1(v) \tag{28}$$

Formula (28) is equivalent to $\neg\exists v.\phi_1$ where

$$\phi_1 \equiv g(v,z) \leq g(z,v) \wedge \mathsf{ls}_g(v) \wedge \mathsf{ls}_g(w) \wedge \neg(g_1(w) \leq g_1(v)) \tag{29}$$

We next use (21) to eliminate atomic formulas $t_1 \leq t_2$ and replace them with cardinality constraints, resulting in formula ϕ_2 equivalent to ϕ_1:

$$\phi_2 \equiv \phi_{2,1} \wedge \phi_{2,2}$$

where

$$\begin{aligned}
\phi_{2,1} \equiv\ & |g(v,z) \cap g(z,v)^c|_{\mathsf{sh}(g(v,z))} = 0\ \wedge \\
& \mathsf{sh}(g(v,z)) = \mathsf{sh}(g(z,v))\ \wedge \\
& \mathsf{ls}_g(v) \wedge \mathsf{ls}_g(w)
\end{aligned} \tag{30}$$

and

$$\begin{aligned}
&\phi_{2,2} \equiv \\
&\quad \neg\left(|g_1(w) \cap g_1(v)^c|_{\mathsf{sh}(g_1(w))} = 0\ \wedge\ \mathsf{sh}(g_1(w)) = \mathsf{sh}(g_1(v))\right)
\end{aligned} \tag{31}$$

Here we have written e.g.

$$|g(v,z) \cap g(z,v)^c|_{\mathsf{sh}(g(v,z))} = 0$$

as a shorthand for

$$|g(v,z) \cap_{\mathsf{sh}(g(v,z))} g(z,v)^c_{\mathsf{sh}(g(v,z))}|_{\mathsf{sh}(g(v,z))} = 0$$

(In general, we omit term shape arguments for boolean algebra operations if the arguments are identical to the enclosing term shape argument of the cardinality constraint.)

We next transform ϕ_2 into disjunction of well-defined conjunctions. Following the ideas in Proposition 8, we transform $\phi_{2,2}$ into $\phi_{3,1} \vee \phi_{3,2}$ where

$$\begin{aligned}
&\phi_{3,1} \equiv \\
&\quad |g_1(w) \cap g_1(v)^c|_{\mathsf{sh}(g_1(w))} \geq 1\ \wedge\ \mathsf{sh}(g_1(w)) = \mathsf{sh}(g_1(v))
\end{aligned} \tag{32}$$

and

$$\phi_{3,2}\ \equiv\ \mathsf{sh}(g_1(w)) \neq \mathsf{sh}(g_1(v))$$

and then transform $\phi_{2,1} \wedge \phi_{2,2}$ into

$$(\phi_{2,1} \wedge \phi_{3,1}) \vee (\phi_{2,1} \wedge \phi_{3,2})$$

For the sake of brevity we ignore the case $\phi_{2,1} \wedge \phi_{3,2}$; it is possible to show that $\phi_{2,1} \wedge \phi_{3,2}$ is equivalent to false in the context of the entire formula.

We transform $\phi_{2,1} \wedge \phi_{3,1}$ into unnested form, introducing fresh existentially quantified variables u_{vz}, u_{zv}, u_{w1}, u_{v1}, u^{s}_{vz}, u^{s}_{w1} that denote terms occurring in $\phi_{2,1} \wedge \phi_{3,1}$. The result is formula ϕ_4 where

$$\begin{aligned}
\phi_4\ \equiv\ & \exists u_{vz}, u_{zv}, u_{w1}, u_{v1}, u^{\mathsf{s}}_{vz}, u^{\mathsf{s}}_{w1}. \\
& u_{vz} = g(v,z) \wedge u_{zv} = g(z,v)\ \wedge \\
& u_{w1} = g_1(w) \wedge u_{v1} = g_1(v)\ \wedge \\
& u^{\mathsf{s}}_{vz} = \mathsf{sh}(u_{vz}) \wedge u^{\mathsf{s}}_{w1} = \mathsf{sh}(u_{w1})\ \wedge \\
& \mathsf{sh}(u_{zv}) = u^{\mathsf{s}}_{vz} \wedge \mathsf{sh}(u_{v1}) = u^{\mathsf{s}}_{w1}\ \wedge \\
& \mathsf{ls}_g(v) \wedge \mathsf{ls}_g(w)\ \wedge \\
& |u_{vz} \cap u^c_{zv}|_{u^{\mathsf{s}}_{vz}} = 0\ \wedge\ |u_{w1} \cap u^c_{v1}|_{u^{\mathsf{s}}_{w1}} \geq 1
\end{aligned} \tag{33}$$

To transform ϕ_4 into disjunction of structural base formulas we keep introducing new existentially quantified variables and adding derived conjuncts to satisfy the invariants of Definition 41.

Because $\mathsf{Is}_g(v)$ and $\mathsf{Is}_g(w)$ appear in the conjunct, we give names to the remaining successors of v, w, by introducing $u_{w2} = g_2(w)$, $u_{v2} = g_2(v)$. We may now write the constraints in constructor language, using e.g. conjunct $v = g(u_{v1}, u_{v2})$ instead of

$$\mathsf{Is}_g(v) \;\wedge\; u_{v1} = g_1(v) \;\wedge\; u_{v2} = g_2(v)$$

To ensure that every term variable has an associated shape variable, we introduce fresh variables u^{s}_v, u^{s}_w, u^{s}_z, u^{s}_{w2}, u^{s}_{v2} with conjuncts $u^{\mathsf{s}}_v = \mathsf{sh}(v)$, $u^{\mathsf{s}}_w = \mathsf{sh}(w)$, $u^{\mathsf{s}}_z = \mathsf{sh}(z)$, $u^{\mathsf{s}}_{w2} = \mathsf{sh}(u_{w2})$, $u^{\mathsf{s}}_{v2} = \mathsf{sh}(u_{v2})$.

Note that base formula contains $\mathsf{distinct}(u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_n)$ subformula. In the case when the current conjunction is not strong enough to entail the disequality between shape variables u^{s}_i and u^{s}_j, we perform case analysis, considering the case $u^{\mathsf{s}}_i = u^{\mathsf{s}}_j$ (then u^{s}_i can be replaced by u^{s}_j), and the case $u^{\mathsf{s}}_i \neq u^{\mathsf{s}}_j$. This case analysis will lead to a disjunction of structural base formulas (unless some of the formulas is shown contradictory in the transformation process). In contrast to *shape* variables, we do not not perform case analysis for disequality of *term* variables, because $\mathsf{termBase}$ in Definition 41 does not contain a $\mathsf{distinct}$ subformula.

In this example we perform case analysis on whether $u^{\mathsf{s}}_w = u^{\mathsf{s}}_z$ and $u^{\mathsf{s}}_w = u^{\mathsf{s}}_v$ should hold. For the sake of the example let us consider the case when $u^{\mathsf{s}}_w = u^{\mathsf{s}}_z = u^{\mathsf{s}}_v$, $u^{\mathsf{s}}_{v2} = u^{\mathsf{s}}_{w2}$ and $u^{\mathsf{s}}_{vz}, u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}$ are all distinct. In that case shape variables $u^{\mathsf{s}}_w, u^{\mathsf{s}}_z, u^{\mathsf{s}}_v$ denote the same shape, so let us replace e.g. u^{s}_z and u^{s}_v with u^{s}_w. Similarly, we replace u^{s}_{v2} with u^{s}_{w2}. We obtain conjuncts $\mathsf{sh}(v) = u^{\mathsf{s}}_w, \mathsf{sh}(z) = u^{\mathsf{s}}_w, \mathsf{sh}(u_{v2}) = u^{\mathsf{s}}_{w2}$.

We next ensure homomorphism property P3 in Definition 41. From conjuncts $u_{vz} = g(v, z)$, $\mathsf{sh}(u_{vz}) = u^{\mathsf{s}}_{vz}$, and $\mathsf{sh}(v) = u^{\mathsf{s}}_w$, we conclude

$$\begin{aligned} u^{\mathsf{s}}_{vz} = \;& \mathsf{sh}(u_{vz}) = \\ & \mathsf{sh}(g(v, z)) = \\ & g^{\mathsf{s}}(\mathsf{sh}(v), \mathsf{sh}(z)) = \\ & g^{\mathsf{s}}(u^{\mathsf{s}}_w, u^{\mathsf{s}}_w) \end{aligned}$$

so we add the conjunct $u^{\mathsf{s}}_{vz} = g^{\mathsf{s}}(u^{\mathsf{s}}_w, u^{\mathsf{s}}_w)$ to the formula. Similarly, from $w = g(u_{w1}, u_{w2})$, $\mathsf{sh}(w) = u^{\mathsf{s}}_w$, $\mathsf{sh}(u_{w1}) = u^{\mathsf{s}}_{w1}$, $\mathsf{sh}(u_{w2}) = u^{\mathsf{s}}_{w2}$ we conclude $u^{\mathsf{s}}_w = g(u_{w1}, u_{w2})$ and add this conjunct to the formula. Adding these two conjuncts makes property P3 hold. (Note that, had we decided to consider the case where $\mathsf{sh}(v) \neq \mathsf{sh}(z)$ we would have arrived at a contradiction due to $\mathsf{sh}(u^{\mathsf{s}}_{vz}) = \mathsf{sh}(u^{\mathsf{s}}_{zv})$.)

We next apply rule (25) to reduce all cardinality constraints into cardinality constraints on parameter nodes (nodes u for which there there is no conjunct of form $u = f(u_{i_1}, \ldots, u_{i_k})$). We replace $|u_{vz} \cap u^c_{zv}|_{u^{\mathsf{s}}_{vz}} = 0$ with

$$|u_v \cap u^c_z|_{u^{\mathsf{s}}_w} = 0 \wedge |u_z \cap u^c_v|_{u^{\mathsf{s}}_w} = 0 \tag{34}$$

Variable v is a parameter variable, but z is not, which prevents application of (25). We therefore introduce u_{z1} and u_{z2} such that $z = g(u_{z1}, u_{z2})$. Because $\mathsf{sh}(z) = u^{\mathsf{s}}_w$, we have $\mathsf{sh}(u_{z1}) = u^{\mathsf{s}}_{w1}$ and $\mathsf{sh}(u_{z2}) = u^{\mathsf{s}}_{w2}$ by homomorphism property. We can now continue applying rule (25) to (34). The result is:

$$\begin{aligned} & |u_{v1} \cap u^c_{z1}|_{u^{\mathsf{s}}_{w1}} = 0 \;\wedge\; |u_{z1} \cap u^c_{v1}|_{u^{\mathsf{s}}_{w1}} = 0 \;\wedge \\ & |u_{v2} \cap u^c_{z2}|_{u^{\mathsf{s}}_{w2}} = 0 \;\wedge\; |u_{z2} \cap u^c_{v2}|_{u^{\mathsf{s}}_{w2}} = 0 \end{aligned}$$

To make the formula conform to Definition 41 we introduce internal variables u_v, u_z, u_w corresponding to free variables v, z, w, respectively. The resulting structural base formula is

$$\begin{aligned} & \exists u_{vz}, u_{zv}, u_v, u_z, u_w, u_{v1}, u_{v2}, u_{z1}, u_{z2}, u_{w1}, u_{w2}, \\ & \quad u^{\mathsf{s}}_{vz}, u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}. \\ & \quad\; \mathsf{shapeBase}_1 \wedge \mathsf{termBase}_1 \wedge \\ & \quad\; \mathsf{hom}_1 \wedge \mathsf{cardin}_1 \end{aligned} \tag{35}$$

where

$$\begin{aligned} \mathsf{shapeBase}_1 = \;& u^{\mathsf{s}}_{vz} = g^{\mathsf{s}}(u^{\mathsf{s}}_w, u^{\mathsf{s}}_w) \wedge u^{\mathsf{s}}_w = g^{\mathsf{s}}(u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}) \wedge \\ & \mathsf{distinct}(u^{\mathsf{s}}_{vz}, u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}) \\ \mathsf{termBase}_1 = \;& u_{vz} = g(u_v, u_z) \wedge u_{zv} = g(u_z, u_v) \wedge \\ & u_v = g(u_{v1}, u_{v2}) \wedge u_z = g(u_{z1}, u_{z2}) \wedge \\ & u_w = g(u_{w1}, u_{w2}) \wedge \\ & v = u_v \wedge z = u_z \wedge w = u_w \\ \mathsf{hom}_1 = \;& \\ & \mathsf{sh}(u_{vz}) = u^{\mathsf{s}}_{vz} \wedge \mathsf{sh}(u_{zv}) = u^{\mathsf{s}}_{vz} \wedge \\ & \mathsf{sh}(u_v) = u^{\mathsf{s}}_w \wedge \mathsf{sh}(u_z) = u^{\mathsf{s}}_w \wedge \mathsf{sh}(u_w) = u^{\mathsf{s}}_w \wedge \\ & \mathsf{sh}(u_{v1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{sh}(u_{z1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{sh}(u_{w1}) = u^{\mathsf{s}}_{w1} \wedge \\ & \mathsf{sh}(u_{v2}) = u^{\mathsf{s}}_{w2} \wedge \mathsf{sh}(u_{z2}) = u^{\mathsf{s}}_{w2} \wedge \mathsf{sh}(u_{w2}) = u^{\mathsf{s}}_{w2} \\ \mathsf{cardin}_1 = \;& |u_{v1} \cap u^c_{z1}|_{u^{\mathsf{s}}_{w1}} = 0 \;\wedge\; |u_{z1} \cap u^c_{v1}|_{u^{\mathsf{s}}_{w1}} = 0 \wedge \\ & |u_{v2} \cap u^c_{z2}|_{u^{\mathsf{s}}_{w2}} = 0 \;\wedge\; |u_{z2} \cap u^c_{v2}|_{u^{\mathsf{s}}_{w2}} = 0 \wedge \\ & |u_{w1} \cap u^c_{v1}|_{u^{\mathsf{s}}_{w1}} \geq 1 \end{aligned}$$

Figure 6 shows a graph representation of the subformulas $\mathsf{shapeBase}_1$, $\mathsf{termBase}_1$, and hom_1 of the resulting structural base formula.

Recall that we are eliminating the quantification over v from $\neg\exists v.\phi_1$. We can now existentially quantify over v. As in Proposition 27, we simply remove the conjunct $v = u_v$ from $\mathsf{termBase}$ and the quantifier $\exists v$.

As in Figure 4 of Section 3.4 the structural base formula form allows us to eliminate an existential quantifier, whereas the quantifier-free form allows us to eliminate a negation. We transform the structural base formula (35) into a quantifier-free formula as follows.

We first use rule (7) to eliminate variable u_{vz}, replacing it with $g(v, z)$. In the resulting formula $g(v, z)$ occurs only in hom_1 in the form

$$\mathsf{sh}(g(u_v, u_z)) = u^{\mathsf{s}}_{vz} \tag{36}$$

But (36) is a consequence of conjuncts $u^{\mathsf{s}}_{vz} = g^{\mathsf{s}}(u^{\mathsf{s}}_w, u^{\mathsf{s}}_w)$, $\mathsf{sh}(u_v) = u^{\mathsf{s}}_w$ and $\mathsf{sh}(u_w) = u^{\mathsf{s}}_w$, so we omit (36) from the formula. In analogous way we eliminate variable u_{zv} and the conjuncts that contain it. We also eliminate u_v, analogously to u_{vz} and u_{zv}. In the resulting formula u^{s}_{vz} occurs only in $\mathsf{distinct}$ subformula of $\mathsf{shapeBase}$. Conjuncts $u^{\mathsf{s}}_{vz} \neq u^{\mathsf{s}}_w$, $u^{\mathsf{s}}_{vz} \neq u^{\mathsf{s}}_{w1}$, and $u^{\mathsf{s}}_{vz} \neq u^{\mathsf{s}}_{w2}$ follow from the remaining conjuncts in $\mathsf{shapeBase}$ by acyclicity. Hence we may replace $\mathsf{distinct}(u^{\mathsf{s}}_{vz}, u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2})$ by $\mathsf{distinct}(u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2})$. Now u^{s}_{vz} does not occur in the matrix of the formula, so we may eliminate $\exists u^{\mathsf{s}}_{vz}$ altogether.

The resulting formula is:

$$
\begin{aligned}
\phi_5 \;\equiv\; & \exists u_z, u_w, u_{v1}, u_{v2}, u_{z1}, u_{z2}, u_{w1}, u_{w2}, u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}. \\
& u^{\mathsf{s}}_w = g^{\mathsf{s}}(u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}) \;\wedge\; \mathsf{distinct}(u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}) \;\wedge \\
& u_z = g(u_{z1}, u_{z2}) \;\wedge\; u_w = g(u_{w1}, u_{w2}) \;\wedge \\
& z = u_z \wedge w = u_w \;\wedge \\
& \mathsf{sh}(u_z) = u^{\mathsf{s}}_w \wedge \mathsf{sh}(u_w) = u^{\mathsf{s}}_w \;\wedge \\
& \mathsf{sh}(u_{v1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{sh}(u_{z1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{sh}(u_{w1}) = u^{\mathsf{s}}_{w1} \;\wedge \\
& \mathsf{sh}(u_{v2}) = u^{\mathsf{s}}_{w2} \wedge \mathsf{sh}(u_{z2}) = u^{\mathsf{s}}_{w2} \wedge \mathsf{sh}(u_{w2}) = u^{\mathsf{s}}_{w2} \;\wedge \\
& |u_{v1} \cap u^c_{z1}|_{u^{\mathsf{s}}_{w1}} = 0 \;\wedge\; |u_{z1} \cap u^c_{v1}|_{u^{\mathsf{s}}_{w1}} = 0 \;\wedge \\
& |u_{v2} \cap u^c_{z2}|_{u^{\mathsf{s}}_{w2}} = 0 \;\wedge\; |u_{z2} \cap u^c_{v2}|_{u^{\mathsf{s}}_{w2}} = 0 \;\wedge \\
& |u_{w1} \cap u^c_{v1}|_{u^{\mathsf{s}}_{w1}} \geq 1
\end{aligned} \tag{37}
$$

We next eliminate u_{v1}. It suffices to eliminate it from conjuncts where it occurs, so we consider formula $\phi_{5,1}$:

$$
\begin{aligned}
\phi_{5,1} \;\equiv\; & \exists u_{v1}. \\
& \mathsf{sh}(u_{v1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{sh}(u_{z1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{sh}(u_{w1}) = u^{\mathsf{s}}_{w1} \;\wedge \\
& |u_{v1} \cap u^c_{z1}|_{u^{\mathsf{s}}_{w1}} = 0 \;\wedge\; |u_{z1} \cap u^c_{v1}|_{u^{\mathsf{s}}_{w1}} = 0 \;\wedge \\
& |u_{w1} \cap u^c_{v1}|_{u^{\mathsf{s}}_{w1}} \geq 1
\end{aligned} \tag{38}
$$

Note that all variables from $\phi_{5,1}$ belong to $B(s)$ where s is the value of shape variable u^{s}_{w1} (see (20)). This means that we can apply quantifier elimination for boolean algebra (Section 3.2) to eliminate u_{v1}. The result is

$$
\begin{aligned}
\phi_{5,2} \;\equiv\; & \mathsf{sh}(u_{z1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{sh}(u_{w1}) = u^{\mathsf{s}}_{w1} \;\wedge \\
& |u_{w1} \cap u^c_{z1}|_{u^{\mathsf{s}}_{w1}} \geq 1
\end{aligned} \tag{39}
$$

Similarly, to eliminate u_{v2} we consider formula $\phi_{5,3}$:

$$
\begin{aligned}
\phi_{5,3} \;\equiv\; & \exists u_{v2}. \\
& \mathsf{sh}(u_{v2}) = u^{\mathsf{s}}_{w2} \wedge \mathsf{sh}(u_{z2}) = u^{\mathsf{s}}_{w2} \wedge \mathsf{sh}(u_{w2}) = u^{\mathsf{s}}_{w2} \;\wedge \\
& |u_{v2} \cap u^c_{z2}|_{u^{\mathsf{s}}_{w2}} = 0 \;\wedge\; |u_{z2} \cap u^c_{v2}|_{u^{\mathsf{s}}_{w2}} = 0
\end{aligned} \tag{40}
$$

The result of boolean algebra quantifier elimination on $\phi_{5,3}$ is true (indeed, one may let $u_{v2} = u_{z2}$). The resulting base formula with u_{v1} and u_{v2} eliminated is ϕ_6 :

$$
\begin{aligned}
\phi_6 \;\equiv\; & \exists u_z, u_w, u_{z1}, u_{z2}, u_{w1}, u_{w2}, u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}. \\
& u^{\mathsf{s}}_w = g^{\mathsf{s}}(u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}) \;\wedge\; \mathsf{distinct}(u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}) \;\wedge \\
& u_z = g(u_{z1}, u_{z2}) \;\wedge\; u_w = g(u_{w1}, u_{w2}) \;\wedge \\
& z = u_z \wedge w = u_w \;\wedge \\
& \mathsf{sh}(u_z) = u^{\mathsf{s}}_w \wedge \mathsf{sh}(u_w) = u^{\mathsf{s}}_w \;\wedge \\
& \mathsf{sh}(u_{z1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{sh}(u_{w1}) = u^{\mathsf{s}}_{w1} \;\wedge \\
& \mathsf{sh}(u_{z2}) = u^{\mathsf{s}}_{w2} \wedge \mathsf{sh}(u_{w2}) = u^{\mathsf{s}}_{w2} \;\wedge \\
& |u_{w1} \cap u^c_{z1}|_{u^{\mathsf{s}}_{w1}} \geq 1
\end{aligned} \tag{41}
$$

Observe that the equalities in ϕ_6 are sufficient to express all variables bound in ϕ_6 in terms of free variables (all internal variables are "covered"):

$$
\begin{aligned}
& u_z = z && u_w = w \\
& u_{z1} = g_1(z) && u_{z2} = g_2(z) \\
& u_{w1} = g_1(w) && u_{w2} = g_2(w) \\
& u^{\mathsf{s}}_w = \mathsf{sh}(w) && \\
& u^{\mathsf{s}}_{w1} = g^{\mathsf{s}}_1(\mathsf{sh}(w)) && u^{\mathsf{s}}_{w2} = g^{\mathsf{s}}_2(\mathsf{sh}(w))
\end{aligned} \tag{42}
$$

Structural base formula ϕ_6 is therefore equivalent to the quantifier-free formula $\phi_{7,1}$:

$$
\begin{aligned}
\phi_{7,1} \;\equiv\; & \mathsf{Is}_{g^{\mathsf{s}}}(\mathsf{sh}(w)) \wedge \mathsf{Is}_g(w) \wedge \mathsf{Is}_g(z) \;\wedge \\
& \mathsf{distinct}(g^{\mathsf{s}}_1(\mathsf{sh}(w)), g^{\mathsf{s}}_2(\mathsf{sh}(w))) \\
& \mathsf{sh}(z) = \mathsf{sh}(w) \;\wedge\; |g_1(w) \cap g_1(z)^c|_{g^{\mathsf{s}}_1(\mathsf{sh}(w))} \geq 1
\end{aligned} \tag{43}
$$

When transforming formula ϕ_4 we chose the case $u^{\mathsf{s}}_{w1} \neq u^{\mathsf{s}}_{w1}$. If we choose the case $u^{\mathsf{s}}_{w1} = u^{\mathsf{s}}_{w2}$, we obtain quantifier-free formula $\phi_{7,2}$:

$$
\begin{aligned}
\phi_{7,2} \;\equiv\; & \mathsf{Is}_{g^{\mathsf{s}}}(\mathsf{sh}(w)) \wedge \mathsf{Is}_g(w) \wedge \mathsf{Is}_g(z) \;\wedge \\
& \mathsf{sh}(z) = \mathsf{sh}(w) \;\wedge\; g^{\mathsf{s}}_1(\mathsf{sh}(w)) = g^{\mathsf{s}}_2(\mathsf{sh}(w)) \;\wedge \\
& |g_1(w) \cap g_1(z)^c|_{g^{\mathsf{s}}_1(\mathsf{sh}(w))} \geq 1
\end{aligned} \tag{44}
$$

Our quantifier elimination would also consider the case $\mathsf{sh}(g_2(w)) \neq \mathsf{sh}(g_2(z))$. The procedure finds the case contradictory in a larger context, when eliminating $\exists z$, because $\mathsf{sh}(z) = \mathsf{sh}(x) = \mathsf{sh}(w)$ follows from $z \leq x$ and $w \leq x$. Ignoring this case, we observe that $\phi_{7,1} \vee \phi_{7,2}$ is equivalent to the quantifier-free formula ϕ_8, where

$$
\begin{aligned}
\phi_8 \;\equiv\; & \mathsf{Is}_{g^{\mathsf{s}}}(\mathsf{sh}(w)) \wedge \mathsf{Is}_g(w) \wedge \mathsf{Is}_g(z) \;\wedge \\
& \mathsf{sh}(z) = \mathsf{sh}(w) \;\wedge\; |g_1(w) \cap g_1(z)^c|_{g^{\mathsf{s}}_1(\mathsf{sh}(w))} \geq 1
\end{aligned} \tag{45}
$$

Let us therefore assume that the result of quantifier elimination in (28) is $\neg\phi_8$.

We proceed to eliminate the next quantifier, $\forall w$, from

$$
\forall w.\; w \leq x \wedge w \leq y \Rightarrow \neg\phi_8 \tag{46}
$$

(46) is equivalent to

$$
\neg\exists w.\; w \leq x \wedge w \leq y \wedge \phi_8
$$

After eliminating $\leq$ we obtain

$$
\begin{aligned}
\neg\exists w.\;\; & |w \cap x^c|_{\mathsf{sh}(w)} = 0 \;\wedge\; \mathsf{sh}(x) = \mathsf{sh}(w) \;\wedge \\
& |w \cap y^c|_{\mathsf{sh}(w)} = 0 \;\wedge\; \mathsf{sh}(y) = \mathsf{sh}(w) \;\wedge \\
& \mathsf{Is}_{g^{\mathsf{s}}}(\mathsf{sh}(w)) \wedge \mathsf{Is}_g(w) \wedge \mathsf{Is}_g(z) \;\wedge \\
& \mathsf{sh}(z) = \mathsf{sh}(w) \;\wedge\; |g_1(w) \cap g_1(z)^c|_{g^{\mathsf{s}}_1(\mathsf{sh}(w))} \geq 1
\end{aligned} \tag{47}
$$

We now proceed similarly as in eliminating variable v. The result is $\neg\phi_9$ where

$$
\begin{aligned}
\phi_9 \;\equiv\; & \mathsf{sh}(x) = \mathsf{sh}(z) \;\wedge\; \mathsf{sh}(y) = \mathsf{sh}(z) \;\wedge \\
& \mathsf{Is}_g(x) \wedge \mathsf{Is}_g(y) \wedge \mathsf{Is}_g(z) \wedge \mathsf{Is}_{g^{\mathsf{s}}}(\mathsf{sh}(z)) \;\wedge \\
& |g_1(x) \cap g_1(y) \cap g_1(z)^c|_{g^{\mathsf{s}}_1(\mathsf{sh}(z))} \geq 1
\end{aligned} \tag{48}
$$

The remaining quantifiers that bind z, y, and x are eliminated similarly.

To eliminate the quantifier $\exists z$, we need to transform $\neg\phi_9$ into disjunction of base formulas. This transformation requires negation of ϕ_9 and creates several disjuncts. We consider only the two cases, ϕ_{10} and ϕ_{11}, that are not contradictory in the enclosing context of conjuncts $z \le x$ and $z \le y$:

$$\phi_{10} \;\equiv\; \mathsf{sh}(x) = \mathsf{sh}(z) \;\wedge\; \mathsf{sh}(y) = \mathsf{sh}(z) \;\wedge\; \neg\mathsf{ls}_{g^{\mathsf{s}}}(\mathsf{sh}(z)) \tag{49}$$

$$\begin{aligned}\phi_{11} \;\equiv\;& \mathsf{sh}(x) = \mathsf{sh}(z) \;\wedge\; \mathsf{sh}(y) = \mathsf{sh}(z) \;\wedge \\ & \mathsf{ls}_g(x) \wedge \mathsf{ls}_g(y) \wedge \mathsf{ls}_g(z) \wedge \mathsf{ls}_{g^{\mathsf{s}}}(\mathsf{sh}(z)) \;\wedge \\ & |g_1(x) \cap g_1(y) \cap g_1(z)^c|_{g_1^{\mathsf{s}}(\mathsf{sh}(z))} = 0\end{aligned} \tag{50}$$

ϕ_{10} is equivalent to

$$\mathsf{sh}(x) = c^{\mathsf{s}} \wedge \mathsf{sh}(y) = c^{\mathsf{s}} \wedge \mathsf{sh}(z) = c^{\mathsf{s}} \tag{51}$$

The result of eliminating $\exists z$ from

$$\exists z.\; |z \cap x^c|_{\mathsf{sh}(z)} = 0 \;\wedge\; |z \cap y^c|_{\mathsf{sh}(z)} = 0 \;\wedge\; \phi_{10}$$

is therefore

$$\phi_{10,2} \;\equiv\; \mathsf{sh}(x) = \mathsf{sh}(y) \;\wedge\; \neg\mathsf{ls}_{g^{\mathsf{s}}}(\mathsf{sh}(x)) \tag{52}$$

The result of eliminating $\exists z$ from

$$\exists z.\; |z \cap x^c|_{\mathsf{sh}(z)} = 0 \;\wedge\; |z \cap y^c|_{\mathsf{sh}(z)} = 0 \;\wedge\; \phi_{11}$$

is

$$\phi_{11,2} \;\equiv\; \mathsf{sh}(x) = \mathsf{sh}(y) \;\wedge\; \mathsf{ls}_{g^{\mathsf{s}}}(\mathsf{sh}(x))$$

$\phi_{10,2} \vee \phi_{11,2}$ is equivalent to $\mathsf{sh}(x) = \mathsf{sh}(y)$. Converting

$$|x \cap y^c|_{\mathsf{sh}(x)} = 0 \wedge \mathsf{sh}(x) = \mathsf{sh}(y) \Rightarrow \mathsf{sh}(x) = \mathsf{sh}(y)$$

to structural base formula yields true. We conclude that (27) is a true sentence in the structure FT_2, which completes our quantifier elimination procedure example.

♦

Formulas in the Example 42 do not contain disequalities between terms variables, only disequalities between shape variables. If a conjunction contains disequalities between term variables, we eliminate the disequalities using rule (22) in the process of converting formula to disjunction of structural base formulas. The following Example 43 illustrates this process.

Example 43 Consider the formula

$$\phi_6' \;\equiv\; \phi_6 \;\wedge\; u_z \neq u_w$$

Where ϕ_6 is given by (41). By (22), literal $u_z \neq u_w$ is equivalent to $\psi_1 \vee \psi_2$ where

$$\psi_1 \;\equiv\; \mathsf{sh}(u_z) \neq \mathsf{sh}(u_w) \tag{53}$$

and

$$\begin{aligned}\psi_2 \;\equiv\;& \mathsf{sh}(u_z) = \mathsf{sh}(u_w) \;\wedge \\ & |(u_z \cap u_w^c) \cup (u_z^c \cap u_w)|_{\mathsf{sh}(u_z)} \geq 1\end{aligned} \tag{54}$$

unnested form	cardinality constraint
$x = x_1 \cap_s x_2$	$\lvert x + (x_1 \cap x_2)\rvert_s = 0$
$x = x_1 \cup_s x_2$	$\lvert x + (x_1 \cup x_2)\rvert_s = 0$
$x = x_{1s}^c$	$\lvert x + x_1^c\rvert_s = 0$

Figure 7: Elimination of Boolean Algebra Unnested Formulas . Expression $x + y$ is a shorthand for $(x \cap y^c) \cup (y \cap x^c)$.

In this case, formula $\phi_6 \wedge \psi_1$ is contradictory. Formula $\phi_6 \wedge \psi_2$ is equivalent to ϕ_6'' where

$$\begin{aligned}\phi_6'' \;\equiv\;& \exists u_z, u_w, u_{z1}, u_{z2}, u_{w1}, u_{w2}, u_w^{\mathsf{s}}, u_{w1}^{\mathsf{s}}, u_{w2}^{\mathsf{s}}. \\ & u_w^{\mathsf{s}} = g^{\mathsf{s}}(u_{w1}^{\mathsf{s}}, u_{w2}^{\mathsf{s}}) \;\wedge\; \mathsf{distinct}(u_w^{\mathsf{s}}, u_{w1}^{\mathsf{s}}, u_{w2}^{\mathsf{s}}) \;\wedge \\ & u_z = g(u_{z1}, u_{z2}) \;\wedge\; u_w = g(u_{w1}, u_{w2}) \;\wedge \\ & z = u_z \wedge w = u_w \;\wedge \\ & \mathsf{sh}(u_z) = u_w^{\mathsf{s}} \wedge \mathsf{sh}(u_w) = u_w^{\mathsf{s}} \;\wedge \\ & \mathsf{sh}(u_{z1}) = u_{w1}^{\mathsf{s}} \wedge \mathsf{sh}(u_{w1}) = u_{w1}^{\mathsf{s}} \;\wedge \\ & \mathsf{sh}(u_{z2}) = u_{w2}^{\mathsf{s}} \wedge \mathsf{sh}(u_{w2}) = u_{w2}^{\mathsf{s}} \;\wedge \\ & |u_{w1} \cap u_{z1}^c|_{u_{w1}^{\mathsf{s}}} \geq 1 \;\wedge \\ & |(u_z \cap u_w^c) \cup (u_z^c \cap u_w)|_{u_w^{\mathsf{s}}} \geq 1\end{aligned} \tag{55}$$

As in Example 42, we now apply rule (25) to

$$|(u_z \cap u_w^c) \cup (u_z^c \cap u_w)|_{u_w^{\mathsf{s}}} \geq 1$$

and transform ϕ_6'' into a disjunction of base formulas.

♦

We proceed to sketch the general case of quantifier elimination. The following Proposition 44 is analogous to Proposition 27; the proof is again straightforward.

Proposition 44 (Quantification of Structural Base) *If β is a structural base formula and x a free term variable in β, then there exists a base structural formula β_1 equivalent to $\exists x.\beta$.*

The following Proposition 45 corresponds to Proposition 28.

Proposition 45 (Quantifier-Free to Structural Base) *Every well-defined quantifier-free formula ϕ in the language of Figure 5 can be written as* true, false, *or a disjunction of structural base formulas.*

Proof Sketch. Let ϕ be a well-defined quantifier-free formula in the language of Figure 5.

We first use rule (21) to eliminate occurrences of $\leq$ in the formula replacing them with cardinality constraints.

We then convert formula into disjunction $\phi_1 \vee \cdots \vee \phi_n$ of well-formed conjunctions of literals. We next describe how to transform each conjunction ϕ_i into a disjunction of base formulas.

Let ϕ_i be a conjunction of literals. Using the technique of Proposition 9, we convert the formula to unnested form, adding existential quantifiers. We then eliminate unnested conjuncts that contain boolean algebra operations, according to Figure 7. The only atomic formulas in the resulting existentially quantified conjunction are of form $x = a$, $x = b$, $x = g(x_1, x_2)$, $\mathsf{Is}_g(x)$, $x_1 = g_1(x)$, $x_2 = g_2(x)$, $x_1 = x_2$, $x^{\mathsf{s}} = c^{\mathsf{s}}$, $x^{\mathsf{s}} = g^{\mathsf{s}}(x_1^{\mathsf{s}}, x_2^{\mathsf{s}})$, $\mathsf{Is}_{g^{\mathsf{s}}}(x^{\mathsf{s}})$, $x_1^{\mathsf{s}} = g_1^{\mathsf{s}}(x^{\mathsf{s}})$, $x_2^{\mathsf{s}} = g_2^{\mathsf{s}}(x^{\mathsf{s}})$, $x_1^{\mathsf{s}} = x_2^{\mathsf{s}}$, $x^{\mathsf{s}} = \mathsf{sh}(x)$, as well as $|t_1|_{x^{\mathsf{s}}} \geq k$ and $|t_2|_{x^{\mathsf{s}}} = k$ for some x^{s}-terms t_1 and t_2. The only negated atomic formulas are of form $x_1 \neq x_2$, $x_1^{\mathsf{s}} \neq x_2^{\mathsf{s}}$, $\neg\mathsf{Is}_g(x)$ and $\neg\mathsf{Is}_{g^{\mathsf{s}}}(x^{\mathsf{s}})$. As in the proof of Proposition 28, we use (17) to eliminate $\neg\mathsf{Is}_g(x)$ and $\neg\mathsf{Is}_{g^{\mathsf{s}}}(x^{\mathsf{s}})$. This process leaves formulas of form $x_1 \neq x_2$ and $x_1^{\mathsf{s}} \neq x_2^{\mathsf{s}}$ as the only negated atomic formulas.

In the sequel, whenever we perform case analysis and generate a disjunction of conjunctions, existential quantifiers propagate to the conjunctions, so we keep working with a existentially quantified conjunction. The existentially quantified variables will become internal variables of a structural base formula.

We next convert conjuncts that contain only term variables to a base formula, and convert shape part to base formula, as in the proof of Proposition 28. We simultaneously make sure every term variable has an associated shape variable, introducing new shape variables if needed. (This process is interleaved with conversion to base formula, to ensure that there is always a conjunct stating that newly introduced shape variables are distinct.) We also ensure homomorphism requirement by replacing internal variables when we entail their equality. Another condition we ensure is that parameter term variables map to parameter shape variables, and non-parameter term variables to non-parameter shape variables; we do this by performing expansion of term and shape variables. We perform expansion of shape variables as in Section 3.2. Expansion of term variables is even simpler because there is no need to do case analysis on equality of term variable with other variables.

The resulting existentially quantified conjunction might contain disequalities $u \neq u'$ between term variables. We eliminate these disequalities as explained in Example 43, by converting each disequality into a cardinality constraint using (22). In general, we need to consider the case when $\mathsf{sh}(u) \neq \mathsf{sh}(u')$ and generate another disjunct.

Elimination of disequalities might violate previously established homomorphism invariants, so we may need to reestablish these invariants by repeating the previously described steps. The overall process terminates because we never introduce new inequalities between term variables.

As a final step, we convert all cardinality constraints into constraints on parameter term variables, using (25). In the case when the shape of cardinality constraint is c^{s}, we cannot apply (25). However, in that case the homomorphism condition ensures that each of the participating variables is equal to a or equal to b. This means that we can simply evaluate the cardinality constraint in the boolean algebra $\{a, b\}$. If the result is true we simply drop the constraint, otherwise the entire base formula becomes false.

This completes our sketch of transforming a quantifier-free formula into disjunction of structural base formulas. ∎

We introduce the notion of covered variables in structural base formula by generalizing Definition 29.

Definition 46 *The set* covering *of variable coverings of a structural base formula β is the least set S of pairs $\langle u, t\rangle$ where u is an internal (shape or term) variable and t is a term over the free variables of β, such such that:*

1. *if $x = u$ occurs in* termBase *then $\langle u, x\rangle \in S$;*
2. *if $x^{\mathsf{s}} = u^{\mathsf{s}}$ occurs in* shapeBase *then $\langle u^{\mathsf{s}}, x^{\mathsf{s}}\rangle \in S$;*
3. *if $\langle u, t\rangle \in S$ and $u = f(u_1, \ldots, u_k)$ occurs in* termBase *for some $f \in \Sigma$ then $\{\langle u_1, f_1(t)\rangle, \ldots, \langle u_k, f_k(t)\rangle\} \subseteq S$;*
4. *if $\langle u^{\mathsf{s}}, t^{\mathsf{s}}\rangle \in S$ and $u^{\mathsf{s}} = f^{\mathsf{s}}(u_1^{\mathsf{s}}, \ldots, u_k^{\mathsf{s}})$ occurs in* shapeBase *then $\{\langle u_1^{\mathsf{s}}, f_1^{\mathsf{s}}(t^{\mathsf{s}})\rangle, \ldots, \langle u_k^{\mathsf{s}}, f_k^{\mathsf{s}}(t^{\mathsf{s}})\rangle\} \subseteq S$;*
5. *if $\langle u, t\rangle \in S$ and $\mathsf{sh}(u) = u^{\mathsf{s}}$ occurs in* hom *then $\langle u^{\mathsf{s}}, \mathsf{sh}(t)\rangle \in S$.*

Definition 47 *An internal term variable u is covered iff there exists a term t such that $\langle u, t\rangle \in S$. An internal shape variable u^{s} is covered iff there exists a term t^{s} such that $\langle u^{\mathsf{s}}, t^{\mathsf{s}}\rangle \in S$.*

Lemma 48 *Let β be a structural base formula with matrix β_0 and let* covering *be the covering of β.*

1. *If $\langle u, t\rangle \in S$ then $\models \beta_0 \Rightarrow u = t$.*
2. *If $\langle u^{\mathsf{s}}, t^{\mathsf{s}}\rangle \in S$ then $\models \beta_0 \Rightarrow u^{\mathsf{s}} = t^{\mathsf{s}}$.*

Proof. By induction, using Definition 46. ∎

Corollary 49 *Let β be a structural base formula such that every internal variable is covered. Then β is equivalent to a well-defined quantifier-free formula.*

Proof. By Lemma 48 using (7). ∎

Lemma 50 *Let u be an uncovered non-parameter term variable in a structural base formula β such that u is a source i.e. no conjunct of form*

$$u' = f(u_1, \ldots, u, \ldots, u_k)$$

occurs in termBase. *Let β' be the result of dropping u from β. Then β is equivalent to β'.*

Proof. Let u occur in $\mathsf{termBase}$ in form

$$u = f(u_1, \ldots, u_k)$$

The only other occurrence of u in β is in hom and has the form $\mathsf{sh}(u) = u^{\mathsf{s}}$. Because non-parameter term variables are mapped to non-parameter shape variables, $\mathsf{shapeBase}$ contains formula

$$u^{\mathsf{s}} = \mathsf{shapified}(f)(u_1^{\mathsf{s}}, \ldots, u_k^{\mathsf{s}}) \tag{56}$$

where $u_1^{\mathsf{s}}, \ldots, u_k^{\mathsf{s}}$ are such that, by homomorphism property, $\mathsf{sh}(u_i) = u_i^{\mathsf{s}}$ occurs in hom. This means that the conjunct $\mathsf{sh}(u) = u^{\mathsf{s}}$ is a consequence of the remaining conjuncts, so it may be omitted. After that, applying (7) yields a structural base formula β' not containing u, where β' is equivalent to β. ∎

Corollary 51 *Every base formula is equivalent to a base formula without uncovered non-parameter term variables.*

Proof. If a structural base formula has an uncovered non-parameter term variable, then it has an uncovered non-parameter term variable that is a source. By repeated application of Lemma (50) we eliminate all uncovered non-parameter term variables. ∎

The next example illustrates how we deal with cardinality constraints $|1_s|_s \geq k$ and $|1_s| = k$, which contain no term variables. These constraints restrict the size of shape s. Luckily, we can be translate them into shape base formula constraints.

Example 52 (Shape Term Size Constraints)
Let $x < y$ denote conjunction $x \le y \wedge x \ne y$. Let us eliminate quantifiers from formula $\exists x.\phi(x)$ where

$$\begin{aligned} \phi(x) \;\equiv\; & \neg(\exists y.\exists z.\ x < y \wedge y < z) \wedge \\ & \neg(\exists u.\ u < x) \end{aligned} \tag{57}$$

Eliminating variables y, z from the first conjunct and variable u from the second conjunct yields

$$\neg|x^c|_{\mathsf{sh}(x)} \ge 2 \wedge \neg|x|_{\mathsf{sh}(x)} \ge 1$$

which is equivalent to

$$(|x^c|_{\mathsf{sh}(x)} = 0 \vee |x^c|_{\mathsf{sh}(x)} = 1) \wedge |x|_{\mathsf{sh}(x)} = 0$$

and further to disjunction

$$(|x^c|_{\mathsf{sh}(x)} = 0 \wedge |x|_{\mathsf{sh}(x)} = 0)\ \vee\ (|x^c|_{\mathsf{sh}(x)} = 1 \wedge |x|_{\mathsf{sh}(x)} = 0)$$

The first disjunct can be shown contradictory. Let us transform the second disjunct into a structural base formula. After introducing $u = x$ and $u^{\mathsf{s}} = \mathsf{sh}(u)$, we obtain

$$\exists u, u^{\mathsf{s}}.\ x = u\ \wedge\ \mathsf{sh}(u) = u^{\mathsf{s}}\ \wedge\ |u|_{u^{\mathsf{s}}} = 0\ \wedge\ |u^c|_{u^{\mathsf{s}}} = 1$$

Then $\exists x.\phi(x)$ is equivalent to

$$\exists u, u^{\mathsf{s}}.\ \mathsf{sh}(u) = u^{\mathsf{s}}\ \wedge\ |u|_{u^{\mathsf{s}}} = 0\ \wedge\ |u^c|_{u^{\mathsf{s}}} = 1$$

Eliminating parameter term variable u yields

$$\exists u^{\mathsf{s}}.\ |1|_{u^{\mathsf{s}}} = 1$$

Constraint $|1|_{u^{\mathsf{s}}} = 1$ means that the largest set in the boolean algebra $B(s)$ where s is the value of u^{s} has size one. There exists exactly one boolean algebra of size one in the structure FT_2, namely $\{a, b\}$. Therefore, $|1|_{u^{\mathsf{s}}} = 1$ is equivalent to $u^{\mathsf{s}} = c^{\mathsf{s}}$. We may now eliminate u^{s} by letting $u^{\mathsf{s}} = c^{\mathsf{s}}$. We conclude that the sentence $\exists x.\phi(x)$ is true.

Notice that we have also established that formula $\phi(x)$ is equivalent to $\mathsf{sh}(x) = c^{\mathsf{s}}$, as a consequence of

$$|1_{\mathsf{sh}(x)}|_{\mathsf{sh}(x)} = 1$$

♦

The following Proposition 53 corresponds to Proposition 38.

Proposition 53 (Struct. Base to Quantifier-Free)
Every structural base formula β is equivalent to a quantifier-free formula ϕ in the language of Figure 5.

Proof Sketch. By Corollary 51 we may assume that β has no uncovered non-parameter term variables. By Corollary 49 we are done if there are no uncovered variables, so it suffices to eliminate uncovered parameter term variables and uncovered shape variables.

Let u be an uncovered parameter term variable. Then u does not occur in $\mathsf{termBase}$. Indeed, suppose for the sake of contradiction that u occurs in $\mathsf{termBase}$ in some formula

$$u' = f(u_1, \ldots, u, \ldots, u_k)$$

Then u' is an uncovered non-parameter variable in β, which is a contradiction because we have assumed β has no uncovered non-parameter variables. Therefore, u does no occur in $\mathsf{termBase}$, it occurs only in hom and cardin. Let $\mathsf{sh}(u) = u^{\mathsf{s}}$ occur in hom. Let $\psi_1, \ldots, \psi_p$ be all conjuncts of cardin that contain u. Each ψ_i is of form $|t_i|_{u^{\mathsf{s}}} \ge k_i$ or $|t_i|_{u^{\mathsf{s}}} = k_i$ for some u^{s}-term t_i. Let $u_{j_1}, \ldots, u_{j_q}$ be all term variables appearing in t_i terms other than u. Conjunct $\mathsf{sh}(u_{j_r}) = u^{\mathsf{s}}$ occurs in hom for each r where $1 \le r \le q$. The base formula can therefore be written in form

$$\beta_1 \;\equiv\; \exists x_1, \ldots, x_e, x_1^{\mathsf{s}}, \ldots, x_f^{\mathsf{s}}.\ \phi \wedge \phi_1$$

where

$$\begin{aligned} \phi_1 \;\equiv\; \exists u.\ \ & \mathsf{sh}(u) = u^{\mathsf{s}} \wedge \\ & \mathsf{sh}(u_{j_1}) = u^{\mathsf{s}}\ \wedge \ldots \wedge\ \mathsf{sh}(u_{j_q}) = u^{\mathsf{s}} \wedge \\ & \psi_1\ \wedge \ldots \wedge\ \psi_p \end{aligned} \tag{58}$$

All term variables in $\psi_1, \ldots, \psi_k$ range over terms of shape u^{s}. Therefore, ϕ_1 defines a relation in the boolean algebra $B([\![u^{\mathsf{s}}]\!])$. This allows us to apply construction in Section 3.2. We eliminate u from $\psi_1 \wedge \ldots \wedge \psi_p$ and obtain a propositional combination ψ_0 of cardinality constraints with u^{s}-terms. ϕ_0 does not contain variable u. We may assume that ψ_0 is in disjunctive normal form

$$\psi_0 \;\equiv\; \alpha_1 \vee\ \ldots\ \vee \alpha_w$$

Let

$$\phi_{1,i} \;\equiv\; \mathsf{sh}(u_{j_1}) = u^{\mathsf{s}}\ \wedge \ldots \wedge\ \mathsf{sh}(u_{j_q}) = u^{\mathsf{s}}\ \wedge\ \alpha_i$$

for $1 \le i \le w$. Base formula β_1 is equivalent to disjunction of base formulas $\beta_{1,i}$ where

$$\beta_{1,i} \;\equiv\; \exists x_1, \ldots, x_e, x_1^{\mathsf{s}}, \ldots, x_f^{\mathsf{s}}.\ \phi \wedge \phi_{1,i}$$

We have thus eliminated an uncovered parameter term variable u from β_1. By repeating this process we eliminate all uncovered parameter term variables from a base formula. The resulting formula contains no uncovered term variables.

It remains to eliminate uncovered shape variables. This process is similar to term algebra quantifier elimination in Section 3.4. An essential part of construction in Section 3.4 is Lemma 25, which relies on the fact that uncovered parameter variables may take on infinitely many values. We therefore ensure that uncovered parameter shape variables are not constrained by term variables through conjuncts outside $\mathsf{shapeBase}$.

Suppose that u^{s} is an uncovered parameter shape variable in a base formula β. u^{s} does not occur in $\mathsf{termBase}$. u^{s} does not occur in hom either, because all term variables are covered, and a conjunct $\mathsf{sh}(u) = u^{\mathsf{s}}$ would imply that u^{s} is covered. The only possible occurrence of u^{s} is in cardinality constraint ψ of subformula cardin, where ψ is of form $|t|_{u^{\mathsf{s}}} = k$ or of form $|t|_{u^{\mathsf{s}}} \ge k$. Suppose there is some term variable u occurring in t. Then $\mathsf{sh}(u) = u^{\mathsf{s}}$ so u^{s} is covered, which is a contradiction. Therefore, t has no variables. t can thus be simplified to either $0_{u^{\mathsf{s}}}$ or $1_{u^{\mathsf{s}}}$. In general, a constraint of form $|1|_{u^{\mathsf{s}}} = k$ or $|1|_{u^{\mathsf{s}}} \ge k$ is a domain cardinality constraint for boolean algebra $B([\![u^{\mathsf{s}}]\!])$ (see Remark 15 as well as (20)). A constraint containing $|0_{u^{\mathsf{s}}}|$ is equivalent to true or false. A constraint $|1|_{u^{\mathsf{s}}} = 0$ is equivalent to false. A constraint $|1|_{u^{\mathsf{s}}} = k$ for $k \ge 1$ is equivalent to

$$u^{\mathsf{s}} = t_1^{\mathsf{s}} \vee\ \cdots\ \vee u^{\mathsf{s}} = t_p^{\mathsf{s}}$$

where $t_1^{\mathsf{s}}, \ldots, t_p^{\mathsf{s}}$ is the list of all ground terms in signature Σ_0 that have exactly k occurrences of constant c^{s}. We therefore generate a disjunction of base formulas $\beta_1, \dots, \beta_p$ where β_i results from β by replacing $|1|_{u^s} = k$ with $u^s = t_i^s$. We convert each β_i to a disjunction of base formulas by labelling subterms of t_i by internal shape variables and doing case analysis on the equality between new internal shape variables to ensure the invariants of a base formula. The result is a disjunction of base formulas where variable u^s occurs only in shapeBase subformula.

Similarly, $|1|_{u^s} \geq k+1$ is equivalent to $\neg(|1|_{u^s} = k)$ and thus to

$$u^s \neq t_1^s \wedge \dots \wedge u^s \neq t_q^s \tag{59}$$

where $t_1^s, \dots, t_q^s$ is the list of all ground terms in signature Σ_0 that have at most k occurrences of constant c^s. We replace $|1|_{u^s} \geq k+1$ by (59) and again convert the result to a disjunction of base formulas where u^s occurs only in shapeBase subformula.

Each of the resulting base formulas β^1 are such that every uncovered variable in β^1 is a shape variable that occurs only in shapeBase. Let

$$\begin{aligned}\beta^1 \equiv\ & \exists u_1, \dots, u_n, u_1^s, \dots, u_{p^s}^s, u_{p^s+1}^s, \dots, u_{p^s+q^s}^s. \\ & \mathsf{shapeBase}(u_1^s, \dots, u_{n^s}^s, x_1^s, \dots, x_{m^s}^s) \wedge \\ & \mathsf{termBase}(u_1, \dots, u_n, x_1, \dots, x_m) \wedge \\ & \mathsf{hom}(u_1, \dots, u_n, u_1^s, \dots, u_n^s) \wedge \\ & \mathsf{cardin}(u_{p+1}, \dots, u_{p+q}, u_{p^s+1}^s, \dots, u_{p^s+q^s}^s)\end{aligned}$$

where $u_1^s, \dots, u_{p^s}^s$ are uncovered shape variables. Then β^1 is equivalent to β^2:

$$\begin{aligned}\beta^2 \equiv\ & \exists u_1, \dots, u_n, u_{p^s+1}^s, \dots, u_{p^s+q^s}^s. \\ & \phi^2(u_{p^s+1}^s, \dots, u_{p^s+q^s}^s, x_1^s, \dots, x_{m^s}^s) \\ & \mathsf{termBase}(u_1, \dots, u_n, x_1, \dots, x_m) \wedge \\ & \mathsf{hom}(u_1, \dots, u_n, u_1^s, \dots, u_n^s) \wedge \\ & \mathsf{cardin}(u_{p+1}, \dots, u_{p+q}, u_{p^s+1}^s, \dots, u_{p^s+q^s}^s)\end{aligned}$$

Here ϕ^2 is a base formula (Definitions 19 and 21) whose free variables are variables free in β^2 as well as all covered shape variables:

$$\begin{aligned}&\phi^2(u_{p^s+1}^s, \dots, u_{p^s+q^s}^s, x_1^s, \dots, x_{m^s}^s) \equiv \exists u_1^s, \dots, u_{p^s}^s. \\ &\quad \mathsf{shapeBase}(u_1^s, \dots, u_{p^s}^s, u_{p^s+1}^s, u_{p^s+q^s}^s, x_1^s, \dots, x_{m^s}^s)\end{aligned}$$

Applying Lemma 37 we conclude that ϕ^2 is equivalent to some disjunction

$$\bigvee_{i=1}^{k} \phi^{3,i}$$

of base formulas without uncovered variables. Let $\beta^{3,i}$ be the result of replacing ϕ^2 with $\phi^{3,i}$ in β^2. Then β^2 is equivalent to

$$\bigvee_{i=1}^{k} \beta^{3,i}$$

and each $\beta^{3,i}$ has no uncovered variables either, because every free variable of $\phi^{3,i}$ is either free or covered in $\beta^{3,i}$. By Corollary 49 each $\beta^{3,i}$ can be written as a quantifier free formula. ∎

The following Theorem 54 corresponds to 39 of Section 3.4.

Theorem 54 (Two Constants Quant. Elimination) *There exist algorithms A, B such that for a given formula ϕ in the language of Figure 5:*

a) A produces a quantifier-free formula ϕ' in selector language

b) B produces a disjunction ϕ' of structural base formulas

Proof. Analogous to proof of Theorem 39, using Proposition 45 in place of Proposition 28 and Proposition 53 in place of Proposition 38. ∎

Corollary 55 *The first-order theory of the structure FT_2 is decidable.*

This completes description of our quantifier elimination for the first-order theory of structure FT_2, which models structural subtyping with two base types and one binary constructor. It is straightforward to extend the construction of this section to any number of covariant constructors if the base formula has only two constants. In Section 5 we extend the result to any number of constants as well. Finally, in Section 6 we extend the result to allow arbitrary decidable structures for primitive types, even if the number of primitive types is infinite.

5 A Finite Number of Constants

In this section we prove the decidability of structural subtyping of any finite number of constant symbols (primitive types) and any number of function symbols (constructors). We first show the result when all constructors are covariant, we then show the result when some of the constructors are contravariant.

We introduce the notion of Σ-term-power of some structure $\mathcal{C}$ as a generalization of the structure of structural subtyping.

We represent primitive types in structural subtyping as a structure $\mathcal{C}$ with a finite carrier C. We call $\mathcal{C}$ the *base structure.* Without loss of generality, we assume that $\mathcal{C}$ has only relations; functions and constants are definable using relations. Let L_C be a set of relation symbols and let $\leq\ \in L_C$ be a distinguished binary relation symbol. $\leq$ represents the subtype ordering between types. C is finite, so $\mathcal{C}$ is decidable (see Section 6 for the case when C is infinite but decidable).

We represent type constructors as free operations in the term algebra with signature Σ. To represent the variance of constructors we define for each constructor $f \in \Sigma$ of arity $\mathsf{ar}(f) = k$ and each argument $1 \leq i \leq k$ the value $\mathsf{variance}(f, i) \in \{-1, 1\}$. The constructor f is covariant in argument i iff $\mathsf{variance}(f, i) = 1$. For convenience we assume $\mathsf{ar}(f) \geq 1$ for each $f \in \Sigma$.

The Σ-term-power of $\mathcal{C}$ is a structure $\mathcal{P}$ defined as follows. Let $\Sigma' = \Sigma \cup C$. The domain of $\mathcal{P}$ is the set P of finite ground Σ'-terms. Elements of C are viewed as constants of arity 1. The structure $\mathcal{P}$ has signature $\Sigma \cup L_C$. The constructors $f \in \Sigma$ are interpreted in $\mathcal{P}$ as in a free term algebra:

$$[\![f]\!]^{\mathcal{P}}(t_1, \dots, t_k) = f(t_1, \dots, t_k)$$

A relation $r \in L_C \setminus \{\leq\}$ is interpreted pointwise on the terms of same "shape" as follows. $[\![r]\!]^{\mathcal{P}}$ is the least relation ρ such that:

1. if $[\![r]\!]^{\mathcal{C}}(c_1, \dots, c_n)$ then $\rho(c_1, \dots, c_n)$

2. if $\rho(t_{i1}, \ldots, t_{in})$ for all i where $1 \le i \le k$, then

$$\rho(f(t_{11}, \ldots, t_{1k}), \ldots, f(t_{n1}, \ldots, t_{nk}))$$

The relation $\le \in L_C$ is interpreted similarly, but taking into account the variance. $[\![\le]\!]^{\mathcal{P}}$ is the least relation ρ such that

1. if $[\![\le]\!]^{\mathcal{C}}(c_1, c_2)$ then $\rho(c_1, c_2)$
2. if
$$\rho^{\mathsf{variance}(f,i)}(t_{i1}, \ldots, t_{in})$$
for all i where $1 \le i \le k$, then
$$\rho(f(t_{11}, \ldots, t_{1k}), \ldots, f(t_{n1}, \ldots, t_{nk}))$$

Here we use the notation ρ^v for $v \in \{-1, 1\}$ with the meaning: $\rho^1 = \rho$ and $\rho^{-1} = \{\langle y, x\rangle \mid \langle x, y\rangle \in \rho\}$.

We next sketch the decidability of structural subtyping for any finite number of primitive types C. For now we assume that all constructors $f \in \Sigma$ are covariant, the relation $\le$ thus does not play a special role.

5.1 Extended Term-Power Structure

For the purpose of quantifier elimination we define the structure $\mathcal{P}_E$ by extending the domain and the set of operations of the term-power structure $\mathcal{P}$.

The domain of $\mathcal{P}_E$ is $P_E = P \cup P_S$ where P_S is the set of *shapes* defined as follows. Let $\Sigma^{\mathsf{s}} = \{c^{\mathsf{s}}\} \cup \{f^{\mathsf{s}} \mid f \in \Sigma\}$ be a set of function symbols such that c^{s} is a fresh constant symbol with $\mathsf{ar}(c^{\mathsf{s}}) = 0$ and f^{s} are fresh distinct constant symbols with $\mathsf{ar}(f^{\mathsf{s}}) = \mathsf{ar}(f)$ for each $f \in \Sigma$. The set of shapes P_S is the set of ground Σ^{s}-terms. When referring to elements of $\mathcal{P}_E$ by *term* we mean an element of P; by *shape* we mean an element of P_S. We write X^{s} to denote an entity pertaining to shapes as opposed to terms, so $x^{\mathsf{s}}, u^{\mathsf{s}}$ denote variables ranging over shapes, and t^{s} to denotes terms that evaluate to shapes.

The extended structure $\mathcal{P}_E$ contains term algebra operations on terms and shapes (including selector operations and tests, [22, Page 61]), the homomorphism sh, and cardinality constraint relations $|\phi|_{t^{\mathsf{s}}} = k$ and $|\phi|_{t^{\mathsf{s}}} \ge k$:

1. constructors in the term algebra of terms, $f \in \Sigma'$
$[\![f]\!]^{\mathcal{P}_E}(t_1, \ldots, t_k) = f(t_1, \ldots, t_k)$;
2. selectors in term the algebra of terms,
$[\![f_i]\!]^{\mathcal{P}_E}(f(t_1, \ldots, t_k)) = t_i$;
3. constructor tests in the term algebra of terms,
$[\![\mathsf{Is}_f]\!]^{\mathcal{P}_E}(t) = \exists t_1, \ldots, t_k.\ t = f(t_1, \ldots, t_k)$;
4. constructors in the term algebra of shapes, $f^{\mathsf{s}} \in \Sigma^{\mathsf{s}}$
$[\![f^{\mathsf{s}}]\!]^{\mathcal{P}_E}(t_1^{\mathsf{s}}, \ldots, t_k^{\mathsf{s}}) = f^{\mathsf{s}}(t_1^{\mathsf{s}}, \ldots, t_k^{\mathsf{s}})$;
5. selectors in the term algebra of shapes,
$[\![f_i^{\mathsf{s}}]\!]^{\mathcal{P}_E}(f^{\mathsf{s}}(t_1^{\mathsf{s}}, \ldots, t_k^{\mathsf{s}})) = t_i^{\mathsf{s}}$;
6. constructor tests in the term algebra of shapes,
$[\![\mathsf{Is}_{f^{\mathsf{s}}}]\!]^{\mathcal{P}_E}(t^{\mathsf{s}}) = \exists t_1^{\mathsf{s}}, \ldots, t_k^{\mathsf{s}}.\ t^{\mathsf{s}} = f^{\mathsf{s}}(t_1^{\mathsf{s}}, \ldots, t_k^{\mathsf{s}})$;
7. the homomorphism mapping terms to shapes such that:
$$\begin{aligned}&[\![\mathsf{sh}]\!]^{\mathcal{P}_E}(f(t_1, \ldots, t_n)) = \\ &\quad \mathsf{shapified}(f)([\![\mathsf{sh}]\!]^{\mathcal{P}_E}(t_1), \ldots, [\![\mathsf{sh}]\!]^{\mathcal{P}_E}(t_n))\end{aligned} \tag{60}$$
where
$$\begin{aligned}\mathsf{shapified}(x) &= c^{\mathsf{s}}, \quad \text{if } x \in C \\ \mathsf{shapified}(f) &= f^{\mathsf{s}}, \quad \text{if } f \in \Sigma\end{aligned} \tag{61}$$
8. cardinality constraint relations
$$\begin{aligned}&[\![|\phi(x_1, \ldots, x_k)|_{t^{\mathsf{s}}} = k]\!]^{\mathcal{P}_E}(t_1, \ldots, t_k) = \\ &\quad |[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(t_1, \ldots, t_k)| = k\end{aligned} \tag{62}$$
and
$$\begin{aligned}&[\![|\phi(x_1, \ldots, x_k)|_{t^{\mathsf{s}}} \ge k]\!]^{\mathcal{P}_E}(t_1, \ldots, t_k) = \\ &\quad |[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(t_1, \ldots, t_k)| \ge k\end{aligned} \tag{63}$$
where $\phi(x_1, \ldots, x_k)$ is is a first-order formula over the base-structure language L_C with free variables $x_1, \ldots, x_k$, term t^{s} denotes a shape, and k is a nonnegative integer constant.

It remains to complete the semantics of cardinality constraint relations, by defining the set $[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(t_1, \ldots, t_k)$. If s is a shape, we call the set of positions of constant c^{s} in s *leaves* of s, and denote it by $\mathsf{leaves}(s)$. We represent a leaf as a sequence of pairs $\langle f, i\rangle$ where f is a constructor of arity k and $1 \le i \le k$. If $l \in \mathsf{leaves}(s)$ and $\mathsf{sh}(t) = s$, then $t[l]$ denotes the element $c \in C$ at position l in term t i.e. if $l = \langle f^1, i^1\rangle \ldots \langle f^n, i^n\rangle$ then
$$t[l] = f_{i^n}^n(\ldots f_{i^2}^2(f_{i^1}^1(t)) \ldots) \tag{64}$$
We define:
$$\begin{aligned}&[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(t_1, \ldots, t_k) = \\ &\quad \{l \mid [\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{C}}(t_1[l], \ldots, t_k[l])\}\end{aligned} \tag{65}$$
The following equations follow from (65) and can be used as an equivalent alternative definition for cardinality relations:
$$\begin{aligned}&|[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(c_1, \ldots, c_k)| = \\ &\quad \begin{cases} 1, & [\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{C}}(c_1, \ldots, c_k) \\ 0, & \neg[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{C}}(c_1, \ldots, c_k)\end{cases}\end{aligned} \tag{66}$$
$$\begin{aligned}&|[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(f(t_{11}, \ldots, t_{1l}), \ldots, f(t_{k1}, \ldots, t_{kl}))| \\ &\quad = |[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(t_{11}, \ldots, t_{k1})| + \ldots \\ &\quad + |[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(t_{1l}, \ldots, t_{kl})|\end{aligned} \tag{67}$$
Definition (65) generalizes [14, Definition 2.1, Page 63]. We write $|\phi(t_1, \ldots, t_k)|_{t^{\mathsf{s}}} = k$ as a shorthand for the atomic formula $(|\phi(x_1, \ldots, x_k)|_{t^{\mathsf{s}}} = k)(t_1, \ldots, t_k)$, similarly for $|\phi(t_1, \ldots, t_k)|_{t^{\mathsf{s}}} \ge k$. This is more than a notational convenience, see Section 6 for an approach which introduces sets of leaves as elements of the domain of $\mathcal{P}_E$ and defines a cylindric algebra interpreted over sets of leaves. The approach in this section follows [35] in merging the quantifier elimination for products and quantifier elimination for boolean algebras.

Some of the operations in $\mathcal{P}_E$ are partial. We use the definitions and results of Section 2.3 to deal with partial functions. $f_i(t)$ is defined iff $\mathsf{Is}_f(t)$ holds, $f_i^{\mathsf{s}}(t^{\mathsf{s}})$ is defined iff $\mathsf{Is}_{f^{\mathsf{s}}}(t^{\mathsf{s}})$ holds. Cardinality constraints $|\phi(t_1, \ldots, t_k)|_{t^{\mathsf{s}}} = k$ and $|\phi(t_1, \ldots, t_k)|_{t^{\mathsf{s}}} \ge k$ are defined iff $\mathsf{sh}(t_1) = \ldots = \mathsf{sh}(t_k) = t^{\mathsf{s}}$ holds.

The structure $\mathcal{P}_E$ is at least as expressive as $\mathcal{P}$ because the only operations or relations present in $\mathcal{P}$ but not in $\mathcal{P}_E$ are $[\![r]\!]^{\mathcal{P}}$ for $r \in L_C$, and we can express $[\![r]\!]^{\mathcal{P}}(t_1, \ldots, t_k)$ as $|\neg r(t_1, \ldots, t_k)|_{\mathsf{sh}(t_1)} = 0$.

Our goal is to give a quantifier elimination for first-order formulas of structure $\mathcal{P}_E$. By a quantifier-free formula we mean a formula without quantifiers outside cardinality constraints, e.g. the formula $|\forall x. x \le t|_{x^{\mathsf{s}}} = k$ is quantifier-free.

5.2 Structural Base Formulas

In this section we define the notion of structural base formulas for any base structure $\mathcal{C}$ with a finite carrier.

Definition 56 of structural base formula for quantifier elimination in $\mathcal{P}_E$ differs from Definition 41 in the conjuncts of cardin subformula. Instead of cardinality constraints on boolean algebra terms, Definition 56 contains cardinality constraints on first-order formulas.

The notion of base formula and Lemma 25 apply to terms P as well as shapes P_S in the structure P_E because shapes are also terms over the alphabet Σ^{s}. For brevity we write u^* for an internal shape or term variable, and similarly x^* for a free shape or term variable, t^* for terms, f^* for term or shape term algebra constructor and f_i^* for a term or shape term algebra selector.

Definition 56 (Structural Base Formula)
A structural base formula *with:*

- *free term variables* $x_1, \ldots, x_m$*;*
- *internal non-parameter term variables* $u_1, \ldots, u_p$*;*
- *internal parameter term variables* $u_{p+1}, \ldots, u_{p+q}$*;*
- *free shape variables* $x_1^{\mathsf{s}}, \ldots, x_{m^{\mathsf{s}}}^{\mathsf{s}}$*;*
- *internal non-parameter shape variables* $u_1^{\mathsf{s}}, \ldots, u_{p^{\mathsf{s}}}^{\mathsf{s}}$*;*
- *internal parameter shape variables* $u_{p^{\mathsf{s}}}^{\mathsf{s}}, \ldots, u_{p^{\mathsf{s}}+q^{\mathsf{s}}}^{\mathsf{s}}$

is a formula of the form:

$$\begin{aligned}&\exists u_1, \ldots, u_n, u_1^{\mathsf{s}}, \ldots, u_{n^{\mathsf{s}}}^{\mathsf{s}}.\\ &\quad \mathsf{shapeBase}(u_1^{\mathsf{s}}, \ldots, u_{n^{\mathsf{s}}}^{\mathsf{s}}, x_1^{\mathsf{s}}, \ldots, x_{m^{\mathsf{s}}}^{\mathsf{s}}) \land\\ &\quad \mathsf{termBase}(u_1, \ldots, u_n, x_1, \ldots, x_m) \land\\ &\quad \mathsf{termHom}(u_1, \ldots, u_n, u_1^{\mathsf{s}}, \ldots, u_{n^{\mathsf{s}}}^{\mathsf{s}}) \land\\ &\quad \mathsf{cardin}(u_{p+1}, \ldots, u_n, u_{p^{\mathsf{s}}+1}^{\mathsf{s}}, \ldots, u_{n^{\mathsf{s}}}^{\mathsf{s}})\end{aligned}$$

where $n = p + q$, $n^{\mathsf{s}} = p^{\mathsf{s}} + q^{\mathsf{s}}$*, and formulas* shapeBase*,* termBase*,* termHom*,* cardin *are defined as follows.*

$$\begin{aligned}&\mathsf{shapeBase}(u_1^{\mathsf{s}}, \ldots, u_{n^{\mathsf{s}}}^{\mathsf{s}}, x_1^{\mathsf{s}}, \ldots, x_{m^{\mathsf{s}}}^{\mathsf{s}}) =\\ &\quad \bigwedge_{i=1}^{p^{\mathsf{s}}} u_i^{\mathsf{s}} = t_i(u_1^{\mathsf{s}}, \ldots, u_{n^{\mathsf{s}}}^{\mathsf{s}}) \;\land\; \bigwedge_{i=1}^{m^{\mathsf{s}}} x_i^{\mathsf{s}} = u_{j_i}^{\mathsf{s}}\\ &\quad \land \mathsf{distinct}(u_1^{\mathsf{s}}, \ldots, u_n^{\mathsf{s}})\end{aligned}$$

where each t_i *is a shape term of the form* $f^{\mathsf{s}}(u_{i_1}^{\mathsf{s}}, \ldots, u_{i_k}^{\mathsf{s}})$ *for some* $f \in \Sigma_0$, $k = \mathsf{ar}(f)$*, and* $j : \{1, \ldots, m^{\mathsf{s}}\} \to \{1, \ldots, n^{\mathsf{s}}\}$ *is a function mapping indices of free shape variables to indices of internal shape variables.*

$$\begin{aligned}&\mathsf{termBase}(u_1, \ldots, u_n, x_1, \ldots, x_m) =\\ &\quad \bigwedge_{i=1}^{p} u_i = t_i(u_1, \ldots, u_n) \;\land\; \bigwedge_{i=1}^{m} x_i = u_{j_i}\end{aligned}$$

where each t_i *is a term of the form* $f(u_{i_1}, \ldots, u_{i_k})$ *for some* $f \in \Sigma$, $k = \mathsf{ar}(f)$*, and* $j : \{1, \ldots, m\} \to \{1, \ldots, n\}$ *is a function mapping indices of free term variables to indices of internal term variables.*

$$\mathsf{termHom}(u_1, \ldots, u_n, u_1^{\mathsf{s}}, \ldots, u_{n^{\mathsf{s}}}^{\mathsf{s}}) = \bigwedge_{i=1}^{n} \mathsf{sh}(u_i) = u_{j_i}^{\mathsf{s}}$$

where $j : \{1, \ldots, n\} \to \{1, \ldots, n^{\mathsf{s}}\}$ *is some function such that* $\{j_1, \ldots, j_p\} \subseteq \{1, \ldots, p^{\mathsf{s}}\}$ *and* $\{j_{p+1}, \ldots, j_{p+q}\} \subseteq \{p^{\mathsf{s}} + 1, \ldots, p^{\mathsf{s}} + q^{\mathsf{s}}\}$ *(a term variable is a parameter variable iff its shape is a parameter shape variable).*

$$\mathsf{cardin}(u_{p+1}, \ldots, u_n, u_{p^{\mathsf{s}}+1}^{\mathsf{s}}, \ldots, u_{n^{\mathsf{s}}}^{\mathsf{s}}) = \psi_1 \land \cdots \land \psi_d$$

where each ψ_i *is a cardinality constraint of the form*

$$|\phi(u_{j_1}, \ldots, u_{j_l})|_{u^{\mathsf{s}}} = k$$

or

$$|\phi(u_{j_1}, \ldots, u_{j_l})|_{u^{\mathsf{s}}} \geq k$$

where $\{j_1, \ldots, j_l\} \subseteq \{p + 1, \ldots, n\}$ *and the conjunct* $\mathsf{sh}(u_{j_d}) = u^{\mathsf{s}}$ *occurs in* termHom *for* $1 \leq d \leq l$*. We require each structural base formula to satisfy the following conditions:*

P0) the graph associated with shape base formula

$$\exists u_1^{\mathsf{s}}, \ldots, u_{n^{\mathsf{s}}}^{\mathsf{s}}.\ \mathsf{shapeBase}(u_1^{\mathsf{s}}, \ldots, u_{n^{\mathsf{s}}}^{\mathsf{s}}, x_1^{\mathsf{s}}, \ldots, x_{m^{\mathsf{s}}}^{\mathsf{s}})$$

is acyclic;

P1) congruence closure property for shapeBase *subformula: there are no two distinct variables* u_i^{s} *and* u_j^{s} *such that both* $u_i^{\mathsf{s}} = f(u_{l_1}^{\mathsf{s}}, \ldots, u_{l_k}^{\mathsf{s}})$ *and* $u_j^{\mathsf{s}} = f(u_{l_1}^{\mathsf{s}}, \ldots, u_{l_k}^{\mathsf{s}})$ *occur as conjuncts in formula* shapeBase*;*

P2) congruence closure property for termBase *subformula: there are no two distinct variables* u_i *and* u_j *such that both* $u_i = f(u_{l_1}, \ldots, u_{l_k})$ *and* $u_j = f(u_{l_1}, \ldots, u_{l_k})$ *occur as conjuncts in formula* termBase*;*

P3) homomorphism property of sh*: for every non-parameter term variable* u *such that* $u = f(u_{i_1}, \ldots, u_{i_k})$ *occurs in* termBase*, if conjunct* $\mathsf{sh}(u) = u^{\mathsf{s}}$ *occurs in* termHom*, then for some shape variables* $u_{j_1}^{\mathsf{s}}, \ldots, u_{j_k}^{\mathsf{s}}$ *term* $u^{\mathsf{s}} = f^{\mathsf{s}}(u_{j_1}^{\mathsf{s}}, \ldots, u_{j_k}^{\mathsf{s}})$ *occurs in* shapeBase *where* $f^{\mathsf{s}} = \mathsf{shapified}(f)$ *and for every* r *where* $1 \leq r \leq k$*, conjunct* $\mathsf{sh}(u_{i_r}) = u_{j_r}^{\mathsf{s}}$ *occurs in* termHom.

Note that the validity of the occur check for term variables follows from P0) and P3). Another immediate consequence of Definition 56 is the following Proposition 57.

Proposition 57 (Quantification of Str. Base Form.)
If β *is a structural base formula and* x *a free shape or term variable in* β*, then there exists a base structural formula* β_1 *equivalent to* $\exists x.\beta$*.*

We proceed to show that a quantifier-free formula can be written as a disjunction of base formulas, and a base formula can be written as a quantifier-free formula.

5.3 Conversion to Base Formulas

Conversion from a quantifier-free formula to the structural base formula is given by Proposition 57. The proof of Proposition 58 is analogous to the proof of Proposition 45 but uses of (67) instead of (25).

Proposition 58 (Quantifier-Free to Structural Base)
Every well-defined quantifier-free formula ϕ *is equivalent on* $\mathcal{P}_E$ *to* true*,* false*, or some disjunction of structural base formulas.*

5.4 Conversion to Quantifier-Free Formulas

The conversion from structural base formulas to quantifier-free formulas is similar to the case of two constant symbols in Section 4.3, but requires the use of Feferman-Vaught technique.

Definition 59 *The set* determinations *of variable determinations of a structural base formula β is the least set S of pairs $\langle u^*, t^* \rangle$ where u^* is an internal term or shape variable and t^* is a term over the free variables of β, such such that:*

1. *if $x^* = u^*$ occurs in* termBase *or* shapeBase*, then $\langle u^*, x^* \rangle \in S$;*
2. *if $\langle u^*, t^* \rangle \in S$ and $u^* = f^*(u_1^*, \ldots, u_k^*)$ occurs in* shapeBase *or* termBase *then $\{\langle u_1^*, f_1^*(t^*) \rangle, \ldots, \langle u_k^*, f_k^*(t^*) \rangle\} \subseteq S$;*
3. *if $\{\langle u_1^*, f_1^*(t^*) \rangle, \ldots, \langle u_k^*, f_k^*(t^*) \rangle\} \subseteq S$ and $u^* = f^*(u_1^*, \ldots, u_k^*)$ occurs in* shapeBase *or* termBase *then $\langle u^*, t^* \rangle \in S$;*
4. *if $\langle u, t \rangle \in S$ and $\mathsf{sh}(u) = u^{\mathsf{s}}$ occurs in* termHom *then $\langle u^{\mathsf{s}}, \mathsf{sh}(t) \rangle \in S$.*

Definition 60 *An internal variable u^* is* determined *if $\langle u^*, t^* \rangle \in$* determinations *for some term t^{s}. An internal variable is* undetermined *if it is not determined.*

Lemma 61 *Let β be a structural base formula with matrix β_0 and let* determinations *be the determinations of β. If $\langle u^*, t^* \rangle \in S$ then $\models \beta_0 \Rightarrow u^* = t^*$.*

Corollary 62 *Let β be a structural base formula such that every internal variable is determined. Then β is equivalent to a well-defined quantifier-free formula.*

Proof. By Lemma 61 using

$$\exists x. x = t \wedge \phi(x) \iff \phi(t) \tag{68}$$

■

Lemma 63 *Let u be an undetermined non-parameter term variable in a structural base formula β such that u is a source i.e. no conjunct of the form*

$$u' = f(u_1, \ldots, u, \ldots, u_k)$$

occurs in termBase*. Let β' be the result of removing u and conjuncts containing u from β. Then β is equivalent to β'.*

Proof. The conjunct containing u in termHom is a consequence of the remaining conjuncts, so we drop it. We then apply (68). ■

Corollary 64 *Every base formula is equivalent to a base formula without undetermined non-parameter term variables.*

Proof. If a structural base formula has an undetermined non-parameter term variable, then it has an undetermined non-parameter term variable that is a source. Repeatedly apply Lemma 63 to eliminate all undetermined non-parameter term variables. ■

The following Lemma 65 is a consequence of the fact that terms of a fixed shape s form a substructure of $\mathcal{P}$ isomorphic to the finite power $\mathcal{C}^m$ where $m = |\mathsf{leaves}(s)|$ and follows from Feferman-Vaught theorem in Section 3.3.

Lemma 65 *Let*

$$\begin{aligned} \alpha \;\equiv\; \exists u.\; & \mathsf{sh}(u) = u^{\mathsf{s}} \wedge \\ & \mathsf{sh}(u_1) = u^{\mathsf{s}} \;\wedge \ldots \wedge\; \mathsf{sh}(u_k) = u^{\mathsf{s}} \;\wedge \\ & \psi_1 \;\wedge \ldots \wedge\; \psi_p \end{aligned} \tag{69}$$

where each ψ_i is a cardinality constraint of the form $|\phi|_{u^{\mathsf{s}}} = k$ or $|\phi|_{u^{\mathsf{s}}} \geq k$ where all free variables of ϕ are among $u, u_1, \ldots, u_k$. Then there exists formula ψ such that ψ is a disjunction of conjunctions of cardinality constraints $|\phi'| = k$ and $|\phi'| \geq k$ where the free variables in each ϕ' are among $u_1, \ldots, u_k$ and formula α is equivalent on $\mathcal{P}_E$ to α' where

$$\alpha' \;\equiv\; \mathsf{sh}(u_1) = u^{\mathsf{s}} \;\wedge \ldots \wedge\; \mathsf{sh}(u_k) = u^{\mathsf{s}} \;\wedge\; \psi \tag{70}$$

Proposition 66 (Struct. Base to Quantifier-Free) *Every structural base formula β is equivalent on $\mathcal{P}_E$ to some well-defined quantifier-free formula ϕ.*

Proof Sketch. By Corollary 64 we may assume that β has no undetermined non-parameter term variables. By Corollary 62 we are done if there are no undetermined variables, so it suffices to eliminate undetermined parameter term variables and undetermined shape variables.

Let u be an undetermined parameter term variable. u does not occur in termBase because it cannot have a successor or a predecessors in the graph associated with term base formula. Therefore, u' occurs only in termHom and cardin. Let u^{s} be the shape variable such that $u^{\mathsf{s}} = \mathsf{sh}(u)$ occurs in termHom. Let $\psi_1, \ldots, \psi_p$ be all conjuncts of cardin that contain u.

Each ψ_i is of the form $|\phi|_{u^{\mathsf{s}}} \geq k_i$ or $|\phi|_{u^{\mathsf{s}}} = k_i$ and for each variable u' free in ϕ the conjunct $\mathsf{sh}(u) = u^{\mathsf{s}}$ occurs in termHom. The base formula can therefore be written in form

$$\beta_1 \;\equiv\; \exists x_1, \ldots, x_e, x_1^{\mathsf{s}}, \ldots, x_f^{\mathsf{s}}.\; \phi \wedge \alpha$$

where α has the form as in Lemma 65. Applying Lemma 65 we eliminate u and obtain $\psi = \bigvee_{i=1}^{w} \alpha_i$ where and each α_i is a conjunction of cardinality constraints. Base formula β_1 is thus equivalent to the disjunction $\bigvee_{i=1}^{w} \beta_{1,i}$ where each $\beta_{1,i}$ is a base formula

$$\beta_{1,i} \;\equiv\; \exists x_1, \ldots, x_e, x_1^{\mathsf{s}}, \ldots, x_f^{\mathsf{s}}.\; \phi \wedge \phi_{1,i}$$

By repeating this process we eliminate all undetermined parameter term variables from a base formula. Each of the resulting base formulas contains no undetermined term variables.

It remains to eliminate undetermined shape variables. This process is similar to term algebra quantifier elimination; the key ingredient is Lemma 25, which relies on the fact that undetermined parameter variables may take on infinitely many values. We therefore ensure that undetermined parameter shape variables are not constrained by term and parameter variables through conjuncts outside shapeBase.

Consider an undetermined parameter shape variable u^{s}. u^{s} does not occur in termHom, because all term variables are determined and a conjunct $u^{\mathsf{s}} = \mathsf{sh}(u)$ would imply that u^{s} is determined as well. u^{s} can thus occur only in cardin within some cardinality constraint $|\phi|_{u^{\mathsf{s}}} = k$ or $|\phi|_{u^{\mathsf{s}}} \geq k$. Moreover, formula ϕ in each such cardinality constraint is closed: otherwise ϕ would contain some free variable u, by definition of base formula u would have to be a parameter variable, all parameter term variables are determined, so u^{s} would be determined as well. Let u^{s} denote some shape s. Because ϕ is a closed formula, $|\phi|$ is equal to 0 if $[\![\phi]\!]^{\mathcal{C}} = \mathsf{false}$ and to the shape size $m = |\mathsf{leaves}(s)|$ if $[\![\phi]\!]^{\mathcal{C}} = \mathsf{true}$. (The fact that closed formulas reduce to the constraints on domain size appears in [35, Theorem 3.36, Page 13].) After eliminating constraints equivalent to $0 = k$ and $0 \geq k$, we obtain a conjunction of simple linear constraints of the form $m = k$ and $m \geq k$. These constraints specify a finite or infinite set $S \subseteq \{0, 1, \ldots\}$ of possible sizes m. Let $A = \{s \mid |\mathsf{leaves}(s)| \in S\}$. If the set S is infinite then it contains an infinite interval of form $\{m_0, m_0 + 1, \ldots\}$ so the set A is infinite. If Σ contains a unary constructor and S is nonempty, then A is infinite. If Σ contains no unary constructors and S is finite then A is finite and the cardinality constraints containing u^{s} are equivalent to $\bigvee_{i=1}^{p} u^{\mathsf{s}} = t_i^{\mathsf{s}}$ where $A = \{t_1^{\mathsf{s}}, \ldots, t_p^{\mathsf{s}}\}$. We therefore generate a disjunction of base formulas $\beta_1, \ldots, \beta_p$ where β_i results from β by replacing cardinality constraints containing u^{s} with with $u^{\mathsf{s}} = t_i^{\mathsf{s}}$. We convert each β_i to a disjunction of base formulas by labelling subterms of t_i with internal shape variables and doing case analysis on the equality between new internal shape variables to ensure the invariants of a base formula, as in the proof of 58. By repeating this process for all shape variables u^{s} where the set S is finite, we obtain base formulas where the set A is infinite for every undetermined parameter shape variable u^{s}. We may then eliminate all undetermined parameter and non-parameter shape variables along with the conjuncts that contain them. The result is an equivalent formula by Lemma 25.

All variables in each of the resulting base formulas are determined. By Corollary 62 each formula can be written as a quantifier-free formula, and the resulting disjunction is a quantifier-free formula. ■

5.5 One-Relation-Symbol Variance

So far we have assumed that all constructors are covariant. In this section we describe the changes needed to extend the result to the case when the constructors have arbitrary variance with respect to some distinguished binary relation denoted $\leq$.

Definition 67 *If ϕ is a first-order formula in the language L_C the* contravariant version *of ϕ, denoted $\phi^{(-1)}$, is defined by induction on the structure of formula by:*

$$\begin{aligned}
(r(t_1, \ldots, t_k))^{(-1)} &= r(t_1, \ldots, t_k), \text{ if } r \in L_C \setminus \{\leq\} \\
(t_1 \leq t_2)^{(-1)} &= t_2 \leq t_1 \\
(\phi_1 \wedge \phi_2)^{(-1)} &= \phi_1{}^{(-1)} \wedge \phi_2{}^{(-1)} \\
(\phi_1 \vee \phi_2)^{(-1)} &= \phi_1{}^{(-1)} \wedge \phi_2{}^{(-1)} \\
(\neg\phi)^{(-1)} &= \neg\phi^{(-1)} \\
(\exists t.\phi)^{(-1)} &= \exists t.\phi^{(-1)} \\
(\forall t.\phi)^{(-1)} &= \forall t.\phi^{(-1)}
\end{aligned} \tag{71}$$

Define $\mathcal{C}^{-1}$ to have the same domain and same interpretation of operations and relations $r \in L_C \setminus \{\leq\}$ but where

$$[\![\leq]\!]^{\mathcal{C}^{-1}} = ([\![\leq]\!]^{\mathcal{C}})^{-1} \tag{72}$$

We clearly have for every formula ϕ and every valuation σ:

$$[\![\phi^{(-1)}]\!]^{\mathcal{C}} = [\![\phi]\!]^{\mathcal{C}^{-1}} \tag{73}$$

If $l \in \mathsf{leaves}(s)$ is a leaf $l = \langle f^1, i^1\rangle \ldots \langle f^n, i^n\rangle$, define $\mathsf{variance}(l)$ as the product of integers

$$\prod_{j=1}^{n} \mathsf{variance}(f^j, i^j) \tag{74}$$

We generalize (65) to

$$\begin{aligned}
&[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(t_1, \ldots, t_k) = \\
&\qquad \{l \mid [\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{C}'}(t_1[l], \ldots, t_k[l])\}
\end{aligned} \tag{75}$$

where $\mathcal{C}'$ denotes $\mathcal{C}$ for $\mathsf{variance}l = 1$ and $\mathcal{C}^{-1}$ for $\mathsf{variance}l - 1$. Hence, isomorhism between terms of some fixed shape s with $|\mathsf{leaves}(s)| = m$ and $\mathcal{C}^m$ breaks, but there is still an isomorphism with $\mathcal{C}^{P(s)} \times (\mathcal{C}^{-1})^{N(s)}$ where

$$\begin{aligned}
P(s) &= |\{l \in \mathsf{leaves}(s) \mid \mathsf{variance}(l) = 1\}| \\
N(s) &= |\{l \in \mathsf{leaves}(s) \mid \mathsf{variance}(l) = -1\}|
\end{aligned} \tag{76}$$

Because of this isomorphism, Lemma 65 still holds and we may still use Feferman-Vaught theorem from Section 3.3.

Equation (67) generalizes to:

$$\begin{aligned}
&|[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(f(t_{11}, \ldots, t_{1l}), \ldots, f(t_{k1}, \ldots, t_{kl}))| \\
&\quad = \textstyle\sum_{i=1}^{l} |[\![\phi^{(\mathsf{variance}(f,l))}(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(t_{1i}, \ldots, t_{ki})|
\end{aligned} \tag{77}$$

The only change in the proof of Proposition 58 is the use of (77) instead of (67). Most of the proof of Proposition 66 remains unchanged as well; the only additional difficulty is eliminating constraints of the form $|\phi|_{u^{\mathsf{s}}} = k$ and $|\phi|_{u^{\mathsf{s}}} \geq k$ where u^{s} is a parameter shape variable and ϕ is a closed formula. Lemma 68 below addresses this problem.

We say that an algorithm g *finitely computes* some function $f : A \to 2^B$ where B is an infinite set iff g is a function from A to the set $\mathsf{Fin}(B) \cup \{\infty\}$ where $\mathsf{Fin}(B)$ is the set of finite subsets of set B, ∞ is a fresh symbol, and

$$g(a) = \begin{cases} f(a), & \text{if } f(a) \in \mathsf{Fin}(B) \\ \infty, & \text{if } f(a) \notin \mathsf{Fin}(B) \end{cases} \tag{78}$$

Lemma 68 *There exists an algorithm that, given a shape variable u^{s} and a conjunction $\psi \equiv \bigwedge_{i=1}^{n} \psi_i$ of cardinality constraints where each ψ_i is of form $|\phi_i|_{u^{\mathsf{s}}} = k_i$ or $|\phi_i|_{u^{\mathsf{s}}} \geq k_i$ for some closed formula ϕ_i, finitely computes the set*

$$A = \{s \mid [\![\psi]\!]^{\mathcal{P}}[u^{\mathsf{s}} \mapsto s]\} \tag{79}$$

of shapes which satisfy ψ in $\mathcal{P}$.

Proof Sketch. Let ϕ be a closed formula in language L_C. Compute $[\![\phi]\!]^{\mathcal{C}}$ and $[\![\phi^{(-1)}]\!]^{\mathcal{C}}$ and then replace $|\phi|_s$ with one of the expressions $P(s) + N(s)$, $P(s)$, $N(s)$, 0 according to the following table.

$[\![\phi]\!]^{\mathcal{C}}$	$[\![\phi^{(-1)}]\!]^{\mathcal{C}}$	$\|\phi\|_s =$
true	true	$P(s) + N(s)$
true	false	$P(s)$
false	true	$N(s)$
false	false	0

(80)

The constraints of the form $N(s) + P(s) = k$ and $N(s) + P(s) = k$ can be expressed as propositional combinations of constraints of the form $N(s) = k$, $P(s) = k$, $P(s) \geq k$ and $N(s) \geq k$. Therefore, ψ can be written as a propositional combination of these four kinds of constraints and each conjunction $C(s)$ can further be assumed to have one of the forms:

F1) $C_{k_P,k_N}(s) \;\equiv\; P(s) = k_P \wedge N(s) = k_N;$

F2) $C_{k_P,k_N^+}(s) \;\equiv\; P(s) = k_P \wedge N(s) \geq k_N;$

F3) $C_{k_P^+,k_N}(s) \;\equiv\; P(s) \geq k_P \wedge N(s) = k_N;$

F4) $C_{k_P^+,k_N^+}(s) \;\equiv\; P(s) \geq k_P \wedge N(s) \geq k_N.$

Let $A = \{s \in P_S \mid C(s)\}$. To compute A when Σ contains unary constructors, we first restrict Σ to the language Σ' with no unary constructors, and compute the set $A' \subseteq A$ using language Σ'. If A' is empty, so is A, otherwise A is infinite. Assume that Σ contains no unary constructors. Assume further Σ contains at least one binary constructor and at lest one constructor is contravariant in some argument. Let

$$S = \{\langle P(s), N(s)\rangle \mid s \in A\}$$

Because $P(s) + N(s) = |\mathsf{leaves}(s)|$ and there are only finitely many shapes of any given size (every constructor is of arity at least two), it suffices to finitely compute S. S can be given an alternative characterization as follows. If $f \in \Sigma$, $\mathsf{ar}(f) = k$, f is covariant in l arguments and contravariant in $k-l$ arguments define

$$\begin{aligned}&[\![f]\!]^S(\langle p_1, n_1\rangle, \ldots, \langle p_k, n_k\rangle) = \\ &\quad\langle \textstyle\sum_{i=1}^{l} p_i + \sum_{i=l+1}^{k} n_i, \sum_{i=1}^{l} n_i + \sum_{i=l+1}^{k} p_i\rangle\end{aligned} \tag{81}$$

Let U be the subset of $\{\langle p, n\rangle \mid p, n \geq 0\}$ generated from element $\langle 1, 0\rangle$ using operations $[\![f]\!]^S$ for $f \in \Sigma$. Then

$$S = \{\langle p, n\rangle \in U \mid c(p, n)\} \tag{82}$$

where $c(p, n)$ is the linear constraint corresponding to the constraint $C(s)$.

Let $C(s) = C_{k_P,k_N}(s)$. Then $S \subseteq \{\langle p, n\rangle \mid p + n = k_P + k_N\}$. S is therefore a subset of a finite set and is easily computable, which solves case F1).

Let $C(s) = C_{k_P^+,k_N^+}(s)$. Because Σ contains a binary constructor, S contains pairs $\langle p, n\rangle$ with arbitrarily large $p+n$, so either the p components or n component of elements of S grows unboundedly. Because Σ contains a constructor f contravariant in some argument, we can define using f an operation o acting as a constructor covariant in at least one argument and contravariant in at least one argument. Using operation on tuples whose one component grows unboundedly yields tuples whose both components grow unboundedly. Therefore, S is infinite, which solves case F4).

Finally, consider the case $C(s) = C_{k_P,k_N^+}(s)$ (this will solve the case $C(s) = C_{k_P^+,k_N}(s)$ as well). Observe that

$$C_{k_P,k_N^+}(s) = C_{k_P,0^+}(s) \wedge \bigwedge_{i=0}^{k_P-1} \neg C_{k_P,i}(s) \tag{83}$$

Because the set S for each $C_{k_P,i}(s)$ is finite, it suffices to finitely compute S for $C_{k_P,0^+}(s)$. In that case

$$S = \{\langle p, n\rangle \in U \mid p = k_P\} \tag{84}$$

Let

$$\begin{aligned} S_i &= \{\langle p, n\rangle \in U \mid p = i\} \\ T_i &= \{\langle p, n\rangle \in U \mid n = i\} \end{aligned} \tag{85}$$

To finitely compute S, finitely compute the sets S_i and T_i for $0 \leq i \leq k_P$. The algorithm starts with all sets S_i and T_i empty and keeps adding elements according to operations $[\![f]\!]^S$.

Assume that $S_0, T_0, \ldots, S_{i-1}, T_{i-1}$ are finitely computed. The computation of S_i and T_i proceeds as follows. Let $f \in \Sigma$ be a constructor of arity k with l covariant arguments. For S_i we consider all solutions of the equation

$$p_1 + \cdots + p_l + n_{l+1} + \ldots + n_k = i \tag{86}$$

for nonnegative integers $p_1, \ldots, p_l, n_{l+1}, \ldots, n_k$. First consider solution solutions where no variable is equal to i. If for one of the solutions, one of the sets $S_{p_1}, \ldots, S_{p_l}$ is infinite, then S_i is infinite, otherwise add to S_i all elements $\langle i, n\rangle$ where

$$n = n_1 + \cdots + n_l + p_{l+1} + \ldots + p_k \tag{87}$$

If $n \leq k_P$ then also add the same elements $\langle i, n\rangle$ to T_n. Next, proceed analogously with T_i, considering solutions of

$$n_1 + \cdots + n_l + p_{l+1} + \ldots + p_k = i \tag{88}$$

If at this point S_i is not infinite and not empty, then also consider the solutions of (87) where $p_j = i$ for some j. If such solution exists, then mark S_i as infinite. Proceed analogously with T_i. Finally, if both S_i and T_i are still finite but there exists a solution for S_i where $n_{l+j} = i$ for some j *and* exists a solution for T_j where $p_{l+d} = i$ for some d, then mark both S_i and T_i as infinite. This completes the sketch of one step of the computation. (This step also applies to S_0 and T_0; we initially assume that $\langle 1, 0\rangle \in T_0$.) ■

Example 69 Let us apply this algorithm to the special case where $\Sigma = \{f, g\}$ and

$$\mathsf{variance}(g) = \langle 1, 1\rangle$$

$$\mathsf{variance}(f) = \langle -1, 1\rangle$$

Let us see what the set S looks like. If $\langle x, y\rangle \in S$ define $k\langle x, y\rangle = \langle kx, ky\rangle$ as in a vector space.

First, $\langle 1, 0\rangle \in S$ because of c^{s}. Next $\langle 1, 1\rangle \in S$ because of f^{s} and $\langle 2, 0\rangle \in S$ because of g^{s}.

More generally, we have the following composition rule: If $\langle p_1, n_1\rangle, \langle p_2, n_2\rangle$ then

$$\langle p_1 + p_2, n_1 + n_2\rangle \in S$$

because of g^{s}, and

$$\langle n_1 + p_2, p_1 + n_2\rangle \in S$$

because of f^{s}.

Using g^{s} we obtain all pairs $\langle p, 0\rangle$ for $p \geq 1$. Using f^{s} once on those we obtain $\langle 1, n\rangle$ for $n \geq 0$. Adding these we additionally obtain $\langle p, n\rangle$ for $p \geq 2$ and $n \geq 0$. Hence we have all pairs $\langle p, n\rangle$ for $p \geq 1$ and $n \geq 0$ and those are the only ones that can be obtained. Thus,

$$S = \{\langle p, n\rangle \mid p \geq 1 \wedge n \geq 0\}$$

As expected, the case F1) yields a finite and the case F4) an infinite set. The case F2) for $k_P = 0$ is an empty set, otherwise it is an infinite set. The case F3) always yields an infinite set. This solves the problem for two constructors f, g.

lifted relations r' for $r \in L_C$

r' :: $\mathsf{term}^k \to \mathsf{bool}$

term algebra on terms

constructors, $f \in \Sigma$:

f :: $\mathsf{term}^k \to \mathsf{term}$

constructor test, $f \in \Sigma$:

Is_f :: $\mathsf{term} \to \mathsf{bool}$

selectors, $f \in \Sigma$:

f_i :: $\mathsf{term} \to \mathsf{term}$

Figure 8: Basic Operations of Σ-term-power Structure

♦

Lemma 68 allows to carry our the proof of Proposition 66 so we obtain our main result for finite C.

Theorem 70 (Term Power Quant. Elimination) *There exists an algorithm that for a given well-defined formula ϕ produces a quantifier-free formula ϕ' that is equivalent to ϕ on $\mathcal{P}_E$.*

Corollary 71 (Decidability of Structural Subtyping) *Let $\mathcal{C}$ be a structure with a finite carrier and $\mathcal{P}$ a Σ-term-power of $\mathcal{C}$. Then the first-order theory of $\mathcal{P}$ is decidable.*

6 Term-Powers of Decidable Theories

In this section we extend the result of Section 5 on decidability of term-powers of a base structure $\mathcal{C}$ to allow $\mathcal{C}$ to be an arbitrary decidable theory, even if the carrier C is infinite.

To keep a finite language in the case when C is infinite, we introduce a predicate $\mathsf{Is}_{\mathsf{PRI}}$ that allows testing whether $t \in C$ for a term $t \in P$.

In structural base formulas, we now distinguish between 1) composed variables, denoting elements $t \in P$ for which $\mathsf{Is}_f(t)$ holds for some constructor $f \in \Sigma$, and 2) primitive variables, denoting elements $t \in P$ for which $\mathsf{Is}_{\mathsf{PRI}}(t)$ holds.

Another generalization compared to Section 5 is the use of a syntactically richer language for term power algebras; to some extent this richer language can be viewed as syntactic sugar and can be simplified away.

The generalization to infinitely many primitive types and the generalization to a richer language are orthogonal.

For most of the section we focus on covariant constructors, Section 6.5 discusses a generalized notion of variance.

As in Section 3.3 let $\mathcal{C} = \langle C, R\rangle$ be a decidable structure where C is a non-empty set and R is a set of relations interpreting some relational language L_C, such that each $r \in R$ is a relation of arity $\mathsf{ar}(r)$ on set C, i.e. $r \subseteq C^{\mathsf{ar}(r)}$. We assume that R contains a binary relation symbol $r^{=} \in R$, interpreted as equality on the set C.

Operations and relations of the Σ-term-power structure are summarized in Figure 8. We will show the decidability of the first-order theory of the structure with these operations.

In the special case when $C = \{a, b\}$ and

$$r = \{\langle a,a\rangle, \langle a,b\rangle, \langle b,b\rangle\}$$

we obtain the theory in Section 4. When $R = \{r\}$ where r is a partial order on types, we obtain the theory of structural subtyping of non-recursive covariant types. For arbitrary relational structure $\mathcal{C}$, if $f \in \Sigma$ for $\mathsf{ar}(f) = k$ we obtain a structure that properly contains the k-th strong power of structure $\mathcal{C}$, in the terminology of [35].

The structure of this section follows Sections 4. We also associate a boolean algebra of sets with each term t. However, in this case, the elements of the associated boolean algebra are sets of occurrences of the constants that satisfy the given first-order formula interpreted over C. The occurrences of constants within the terms of a given shape correspond to the indices of the product structure in Section 3.3. We call these occurrences *leaves*, because they can be represented as leaves of the tree corresponding to a term.

6.1 Product Theory of Terms of a Given Shape

In this section we define the notions shape and leafset, and state some properties that we use in the sequel.

Let

$$\Sigma_0 = \{c^{\mathsf{s}}\} \cup \{f^{\mathsf{s}} \mid f \in \Sigma\}$$

be a set of function symbols such that c^{s} is a fresh constant symbol with $\mathsf{ar}(c^{\mathsf{s}}) = 0$ and f^{s} are fresh distinct constant symbols with $\mathsf{ar}(f^{\mathsf{s}}) = \mathsf{ar}(f)$ for each $f \in \Sigma$. Let $\mathsf{shapified} : \Sigma' \to \Sigma_0$ be defined by

$$\begin{aligned}\mathsf{shapified}(x) &= c^{\mathsf{s}}, \quad \text{if } x \in C\\ \mathsf{shapified}(f) &= f^{\mathsf{s}}, \quad \text{if } f \in \Sigma\end{aligned}$$

Let $\mathsf{FT}(\Sigma_0)$ be the set of ground terms with signature Σ_0 and $\mathsf{FT}(\Sigma')$ the set of ground terms of signature Σ'.

Define function $\mathsf{sh} :: \mathsf{FT}(\Sigma') \to \mathsf{FT}(\Sigma_0)$ mapping each term to its shape by

$$\mathsf{sh}(f(t_1,\ldots,t_n)) = \mathsf{shapified}(f)(\mathsf{sh}(t_1),\ldots,\mathsf{sh}(t_n))$$

for each $f \in \Sigma'$. Define $t_1 \sim t_2$ iff $\mathsf{sh}(t_1) = \mathsf{sh}(t_2)$.

Let t be a term or shape and t' the tree representing t as in Section 2.2. If p is a path such that $t'(p)$ is defined and denotes a constant, we write $t[p]$ to denote $t'(p)$ and call p a *leaf*. Note that $t[p]$ is defined iff $\mathsf{sh}(t)[p]$ is defined. On the set of equivalent terms leaves act as indices of Section 3.3. If s is a shape, let $\mathsf{leaves}(s)$ denote the set of all leaves defined on shape s.

Generalizing tCont of Section 4.1, define function $\mathsf{tCont} : \mathsf{FT}(\Sigma') \to C^*$ by:

$$\begin{aligned}\mathsf{tCont}(c) &= c^{\mathsf{s}}, \text{if } c \in C\\ \mathsf{tCont}(f(t_1,\ldots,t_k)) &= \mathsf{tCont}(t_1)\cdot\ldots\cdot\mathsf{tCont}(t_k)\end{aligned}$$

Define $\delta(t) = \langle \mathsf{sh}(t), \mathsf{tCont}(t)\rangle$ and

$$B = \{\langle s, w\rangle \mid s \in \mathsf{FT}(\Sigma_0), w \in C^*, \mathsf{tLen}(s) = \mathsf{sLen}(w)\}$$

If all constructors $f \in \Sigma$ are covariant then δ is a bijection between $\mathsf{FT}(\Sigma')$ and B. Let

$$B(s_0) = \{\langle s, w\rangle \in B \mid s = s_0\}$$

For a fixed s_0, the set $B(s_0)$ is isomorphic to the power structure C^n where $n = \mathsf{tLen}(s)$.

For each shape s we introduce operations from Section 3.3. To distinguish the sets of positions belonging to different shapes, we tag each set of positions L with a shape

s. We call the pair $\langle s, L\rangle$ a *leafset*. The interpretation of each relation $r \in L_C$ is the leafset:

$$[\![r_s]\!](t_1,\ldots,t_k) = \langle s, \{p \mid [\![r]\!]^C(t_1[p],\ldots,t_k[p])\}\rangle$$

We let $\wedge^{\mathsf{l}}_s$, $\vee^{\mathsf{l}}_s$, $\neg^{\mathsf{l}}_s$, $\mathsf{true}^{\mathsf{l}}_s$, $\mathsf{false}^{\mathsf{l}}_s$ stand for intersection, union, complement, full set and empty set in the algebra of subsets of the set $\mathsf{leaves}(s)$. We also introduce $\exists^{\mathsf{l}}_s$ as the union of a family of subsets indexed by a term of shape s and $\forall^{\mathsf{l}}_s$ as the intersection of a family of subsets indexed by a term.

We use constructor-selector language for the term algebra on terms. We introduce constructor-selector language on shapes by generalizing operations in Section 4.1 in a natural way. In addition, we introduce a constructor-selector language on leafsets. For each $f \in \Sigma$ we introduce a constructor symbol f^{L} on leafsets and define

$$\mathsf{leafified}(f) = f^{\mathsf{L}}$$

Constructors f^{L} act on leafsets as follows. If $L_i \subseteq \mathsf{leaves}(s_i)$ for $1 \leq i \leq k$ define

$$f^{\mathsf{L}}(\langle s_1, L_1\rangle, \ldots, \langle s_k, L_k\rangle) = \langle s, L\rangle$$

where $s = f^{\mathsf{s}}(s_1,\ldots,s_k)$, and $L \subseteq \mathsf{leaves}(s)$ is given by

$$L = (\{1\}\cdot L_1)\ \cup \cdots \cup\ (\{k\}\cdot L_k)$$

(Here we define $A \cdot B = \{a \cdot b \mid a \in A \wedge b \in B\}$.)

We define selector functions on leafsets as follows. If $s = f^{\mathsf{s}}(s_1,\ldots,s_k)$ and $L \subseteq \mathsf{leaves}(s)$, then $f^{\mathsf{L}}_i(\langle s, L\rangle) = \langle s_i, L_i\rangle$ where $L_i \subseteq \mathsf{leaves}(s_i)$ is defined by

$$L_i = \{w \mid w \cdot i \in A\}$$

Equivalently, we require that

$$f^{\mathsf{L}}_i(f^{\mathsf{L}}(\langle s_1, L_1\rangle, \ldots, \langle s_n, L_n\rangle)) = \langle s_i, L_i\rangle$$

We can now express relations r' in Figure 8 using the fact:

$$\begin{aligned} & r'(t_1,\ldots,t_k) \iff \\ & \mathsf{sh}(t_2) = \mathsf{sh}(t_1)\ \wedge \ldots \wedge\ \mathsf{sh}(t_k) = \mathsf{sh}(t_1)\ \wedge \\ & r_{\mathsf{sh}(t_1)}(t_1,\ldots,t_k) = \mathsf{true}^{\mathsf{l}}_{\mathsf{sh}(t_1)} \end{aligned} \tag{89}$$

To handle an infinite number of elements of the base structure $\mathcal{C}$, we do not introduce into the language constants for every element of C as in Section 5. Instead, we introduce the predicate $\mathsf{Is}_{\mathsf{PRI}} :: \mathsf{term} \to \mathsf{bool}$ called *primitive-term test* that checks whether a term is a constant:

$$\mathsf{Is}_{\mathsf{PRI}}(x) = (x \in C)$$

and the predicate $\mathsf{Is}_{\mathsf{PRI}^{\mathsf{L}}} :: \mathsf{leafset} \to \mathsf{bool}$ called *primitive-leafset test*:

$$\mathsf{Is}_{\mathsf{PRI}^{\mathsf{L}}}(\langle s, L\rangle) = (s = c^{\mathsf{s}})$$

Instead of the rule (16), we have for $f, g \in \Sigma \cup \{\mathsf{PRI}\}$:

$$\begin{aligned} & \forall x.\ \bigvee_{f \in \Sigma \cup \{\mathsf{PRI}\}} \mathsf{Is}_f(x) \\ & \forall x.\ \neg(\mathsf{Is}_f(x) \wedge \mathsf{Is}_g(x)), \qquad \text{for } f \not\equiv g \end{aligned} \tag{90}$$

Analogous rules hold for term algebra of leafsets:

$$\begin{aligned} & \forall x.\ \bigvee_{f \in \Sigma \cup \{\mathsf{PRI}\}} \mathsf{Is}_{f^{\mathsf{L}}}(x) \\ & \forall x.\ \neg(\mathsf{Is}_{f^{\mathsf{L}}}(x) \wedge \mathsf{Is}_{g^{\mathsf{L}}}(x)), \qquad \text{for } f \not\equiv g \end{aligned} \tag{91}$$

Term algebra of shapes satisfies the original rules (16) of term algebra.

6.2 A Logic for Term-Power Algebras

To show the decidability of the first-order theory of the structure FT_* with operations in Figure 8, we show decidability for a richer structure. Figure 9 shows the operations and relations of this richer structure.

The structure has four sorts: bool representing truth values, term representing terms, shape representing shapes, and leafset representing sets of leaves within a given shape. The structure can be seen as as a combination of the operations of Figure 5 and Figure 2.

For each relation symbol $r \in R$ we define a relation symbol r^* of sort $\mathsf{shape} \times \mathsf{term}^k \to \mathsf{bool}$ acting on terms of the same shape. While in Section 4.2 we associate a boolean algebra with the terms of same shape, in this section we associate a cylindric algebra [21] with terms of the same shape. This is a particularly simple cylindric algebra resulting from lifting first-order logic on the base structure $\mathcal{C}$ so that elements are replaced by terms of a given shape (which are isomorphic to functions from leaves to elements), and boolean values are replaced by sets of leaves (isomorphic to functions from leaves to booleans). In both cases, operations on the set X are lifted to operations on the set $\mathsf{leaves}(s) \to X$. Syntactically, we introduce a copy of all propositional connectives and quantifiers: $\wedge^{\mathsf{l}}_{\_}$, $\vee^{\mathsf{l}}_{\_}$, $\neg^{\mathsf{l}}_{\_}$, $\mathsf{true}^{\mathsf{l}}_{\_}$, $\mathsf{false}^{\mathsf{l}}_{\_}$. Like boolean algebra operations in Figure 5, these syntactic constructs in Figure 9 take an additional shape argument, because term-power algebra contains one copy of a strong power $\mathcal{C}^n$ of base structure for each shape. We call formulas built using the operations of the cylindric algebra *inner formulas*.

For each operation in Figure 2 there is an operation in Figure 9, potentially taking a shape as an additional argument (for operations used to build inner formulas). The logic further contains term algebra operations on terms, leafsets, and shapes.

We use undecorated identifiers (e.g. u) to denote variables of term sort, variables with superscript S to denote shape variables (e.g. u^{s}) and variables with superscript L to denote leafset variables (e.g. u^{L}).

Figures 10 and 11 show the semantics of logic in Figure 9. The first row specifies semantics of operations in the case when all arguments are defined and are in the domain of the operation. The domain of each operation is in the second column, it is omitted if it is equal to the entire domain resulting from interpreting the sort of the operation. All operations except for plain logical operations and quantifiers over the bool domain are strict. Logical operations and quantifiers over the bool domain are defined as in the three-valued logic of Section 2.3.

We remark that values of leafset act as terms with two constants in Figure 5. In fact, if the base structure $\mathcal{C}$ has only two constants then the formula $x = a$ and its propositional combinations are sufficient to express all facts about $\mathcal{C}$, so in that case there is no need to distinguish between terms and leafsets.

6.3 Some Properties of Term-Power Structure

In this section we establish some further properties of the term-power structure, including the homomorphism properties between the term algebra of terms and the term algebra of leafsets. We also argue that it suffices to consider a restricted class of formulas called *simple formulas*.

per-shape product structure

inner formula relations for $r \in L_C$:

$$r_\_ \;::\; \mathsf{shape} \times \mathsf{term}^k \to \mathsf{leafset}$$

inner logical connectives:

$$\wedge^{\mathsf{l}}_\_, \vee^{\mathsf{l}}_\_ \;::\; \mathsf{shape} \times \mathsf{leafset} \times \mathsf{leafset} \to \mathsf{leafset}$$

$$\neg^{\mathsf{l}}_\_ \;::\; \mathsf{leafset} \to \mathsf{leafset}$$

$$\mathsf{true}^{\mathsf{l}}_\_, \mathsf{false}^{\mathsf{l}}_\_ \;::\; \mathsf{leafset}$$

inner formula quantifiers:

$$\exists^{\mathsf{l}}_\_, \forall^{\mathsf{l}}_\_ \;::\; \mathsf{shape} \times (\mathsf{term} \to \mathsf{leafset}) \to \mathsf{leafset}$$

leafset equality:

$$=^{\mathsf{L}} \;::\; \mathsf{leafset} \times \mathsf{leafset} \to \mathsf{bool}$$

leafset cardinality constraints, $k \geq 0$:

$$|\_|_\_ \geq k,\; |\_|_\_ = k \;::\; \mathsf{shape} \times \mathsf{leafset} \to \mathsf{bool}$$

leafset quantifiers:

$$\exists^{\mathsf{L}}, \forall^{\mathsf{L}} \;::\; (\mathsf{leafset} \to \mathsf{bool}) \to \mathsf{bool}$$

term equality:

$$= \;::\; \mathsf{term} \times \mathsf{term} \to \mathsf{bool}$$

term quantifiers:

$$\exists, \forall \;::\; (\mathsf{term} \to \mathsf{bool}) \to \mathsf{bool}$$

shape equality:

$$=^{\mathsf{s}} \;::\; \mathsf{shape} \times \mathsf{shape} \to \mathsf{bool}$$

shape quantifiers:

$$\exists^{\mathsf{s}}, \forall^{\mathsf{s}} \;::\; (\mathsf{shape} \to \mathsf{bool}) \to \mathsf{bool}$$

logical connectives:

$$\wedge, \vee \;::\; \mathsf{bool} \times \mathsf{bool} \to \mathsf{bool}$$

$$\neg \;::\; \mathsf{bool} \to \mathsf{bool}$$

$$\mathsf{true}, \mathsf{false}, \mathsf{undef} \;::\; \mathsf{bool}$$

term algebra on terms

constructors, $f \in \Sigma$:

$$f \;::\; \mathsf{term}^k \to \mathsf{term}$$

constructor test, $f \in \Sigma$:

$$\mathsf{Is}_f \;::\; \mathsf{term} \to \mathsf{bool}$$

primitive-term test:

$$\mathsf{Is}_{\mathsf{PRI}} \;::\; \mathsf{term} \to \mathsf{bool}$$

selectors, $f \in \Sigma$:

$$f_i \;::\; \mathsf{term} \to \mathsf{term}$$

term shape:

$$\mathsf{sh} \;::\; \mathsf{term} \to \mathsf{shape}$$

term algebra on leafsets

constructors, $f \in \Sigma$:

$$f^{\mathsf{L}} \;::\; \mathsf{leafset}^k \to \mathsf{leafset}$$

constructor test, $f \in \Sigma$:

$$\mathsf{Is}_{f^{\mathsf{L}}} \;::\; \mathsf{leafset} \to \mathsf{bool}$$

primitive-leafset test:

$$\mathsf{Is}_{\mathsf{PRI}^{\mathsf{L}}} \;::\; \mathsf{leafset} \to \mathsf{bool}$$

selectors, $f \in \Sigma$:

$$f^{\mathsf{L}}_i \;::\; \mathsf{leafset} \to \mathsf{leafset}$$

leafset shape:

$$\mathsf{lssh} \;::\; \mathsf{leafset} \to \mathsf{shape}$$

term algebra on shapes

constructors, $f \in \Sigma_0$:

$$f^{\mathsf{s}} \;::\; \mathsf{shape}^k \to \mathsf{shape}$$

constructor test, $f \in \Sigma_0$:

$$\mathsf{Is}_{f^{\mathsf{s}}} \;::\; \mathsf{shape} \to \mathsf{bool}$$

selectors, $f \in \Sigma$:

$$f^{\mathsf{s}}_i \;::\; \mathsf{shape} \to \mathsf{shape}$$

Figure 9: Operations and relations in structure $\mathcal{P}$

interpretation of sorts

$$\begin{aligned}
[\![\mathsf{term}]\!] &= \mathsf{FT}(\Sigma') \\
[\![\mathsf{shape}]\!] &= \mathsf{FT}(\Sigma_0) \\
[\![\mathsf{leafset}]\!] &= \{\langle s, L\rangle \mid L \subseteq \mathsf{leaves}(s)\} \\
[\![\mathsf{bool}]\!] &= \{\mathsf{true}, \mathsf{false}, \mathsf{undef}\}
\end{aligned}$$

semantics	**well-definedness**
inner formula relations for $r \in L_C$*:*	
$[\![r]\!](s, t_1, \ldots, t_k) = \langle s, \{l \mid [\![r]\!]^C(t_1[l], \ldots, t_k[l])\}\rangle$	$\mathsf{sh}(t_1) = s \wedge \ldots \wedge \mathsf{sh}(t_k) = s$
inner logical connectives:	
$[\![\wedge^{\mathsf{I}}]\!](s, \langle s_1, L_1\rangle, \langle s_2, L_2\rangle) = \langle s, L_1 \cap L_2\rangle$	$s_1 = s \wedge s_2 = s$
$[\![\vee^{\mathsf{I}}]\!](s, \langle s_1, L_1\rangle, \langle s_2, L_2\rangle) = \langle s, L_1 \cup L_2\rangle$	$s_1 = s \wedge s_2 = s$
$[\![\neg^{\mathsf{I}}]\!](s, \langle s_1, L_1\rangle) = \langle s, \mathsf{leaves}(s) \setminus L_1\rangle$	$s_1 = s$
$[\![\mathsf{true}^{\mathsf{I}}]\!](s) = \langle s, \mathsf{leaves}(s)\rangle$	
$[\![\mathsf{false}^{\mathsf{I}}]\!](s) = \langle s, \emptyset\rangle$	
inner formula quantifiers, for $h : [\![\mathsf{term}]\!] \to [\![\mathsf{leafset}]\!]$*:*	
$[\![\exists^{\mathsf{I}}]\!](s, h) = \langle s, \bigcup\{L \mid \exists t \in [\![\mathsf{term}]\!].\ \mathsf{sh}(t) = s \wedge h(t) = \langle s, L\rangle\}\rangle$	$\forall t \in [\![\mathsf{term}]\!].\ \mathsf{lssh}(h(t)) = s$
$[\![\forall^{\mathsf{I}}]\!](s, h) = \langle \bigcap\{L \mid \exists t \in [\![\mathsf{term}]\!].\ \mathsf{sh}(t) = s \wedge h(t) = \langle s, L\rangle\}, \rangle$	$\forall t \in [\![\mathsf{term}]\!].\ \mathsf{lssh}(h(t)) = s$
leafset equality:	
$[\![=^{\mathsf{L}}]\!](\langle s_1, L_1\rangle, \langle s_2, L_2\rangle) = s_1 = s_2 \wedge L_1 = L_2$	
leafset cardinality constraints:	
$[\![\vert\langle s_1, L_1\rangle\vert_s \geq k]\!] = (\vert L_1\vert \geq k)$	$s_1 = s$
$[\![\vert\langle s_1, L_1\rangle\vert_s = k]\!] = (\vert L_1\vert = k)$	$s_1 = s$
leafset quantifiers, for $h : [\![\mathsf{leafset}]\!] \to [\![\mathsf{bool}]\!]$*:*	
$[\![\exists^{\mathsf{L}}]\!]h = \exists\langle s, t\rangle \in [\![\mathsf{leafset}]\!].\ h(\langle s, t\rangle)$	
$[\![\forall^{\mathsf{L}}]\!]h = \forall\langle s, t\rangle \in [\![\mathsf{leafset}]\!].\ h(\langle s, t\rangle)$	
term equality:	
$[\![=]\!](t_1, t_2) = (t_1 = t_2)$	
term quantifiers, for $h : [\![\mathsf{term}]\!] \to [\![\mathsf{bool}]\!]$*:*	
$[\![\exists]\!]h = \exists t \in [\![\mathsf{term}]\!].\ h(t)$	
$[\![\forall]\!]h = \forall t \in [\![\mathsf{term}]\!].\ h(t)$	
shape equality:	
$[\![=^{\mathsf{s}}]\!](t_1^{\mathsf{s}}, t_2^{\mathsf{s}}) = (t_1^{\mathsf{s}} = t_2^{\mathsf{s}})$	
shape quantifiers, for $h : [\![\mathsf{shape}]\!] \to [\![\mathsf{bool}]\!]$*:*	
$[\![\exists^{\mathsf{s}}]\!]h = \exists t \in [\![\mathsf{shape}]\!].\ h(t)$	
$[\![\forall^{\mathsf{s}}]\!]h = \forall t \in [\![\mathsf{shape}]\!].\ h(t)$	

Figure 10: Semantics for Logic of Term-Power Algebra (Part I)

semantics	well-definedness
term algebra on terms	
constructors, $f \in \Sigma$:	
$\llbracket f \rrbracket(t_1, \ldots, t_k) = f(t_1, \ldots, t_k)$	
constructor test, $f \in \Sigma$:	
$\llbracket \mathsf{Is}_f \rrbracket(t) = \exists t_1, \ldots, t_k.\ t = f(t_1, \ldots, t_k)$	
primitive-term test:	
$\llbracket \mathsf{Is}_{\mathsf{PRI}} \rrbracket(t) = (t \in C)$	
selectors, $f \in \Sigma$:	
$\llbracket f_i \rrbracket(t) = \epsilon t_i.\ t = f(t_1, \ldots, t_i, \ldots, t_k)$	$\llbracket \mathsf{Is}_f \rrbracket(t)$
term shape:	
$\llbracket \mathsf{sh}(f(t_1, \ldots, t_n)) \rrbracket = \mathsf{shapified}(f)(\mathsf{sh}(t_1), \ldots, \mathsf{sh}(t_n))$	
term algebra on leafsets	
constructors, $f \in \Sigma$:	
$\llbracket f^{\mathsf{L}} \rrbracket(\langle s_1, L_1 \rangle, \ldots, \langle s_k, L_k \rangle) = \langle f(s_1, \ldots, s_k), (\{1\} \cdot L_1) \ \cup \cdots \cup \ (\{k\} \cdot L_k) \rangle$	
constructor test, $f \in \Sigma$:	
$\llbracket \mathsf{Is}_{f^{\mathsf{L}}} \rrbracket(\langle s, L \rangle) = \exists s_1, L_1, \ldots, s_k, L_k.\ \langle s, L \rangle = \llbracket f^{\mathsf{L}} \rrbracket(\langle s_1, L_1 \rangle, \ldots, \langle s_k, L_k \rangle)$	
primive-leafset test:	
$\llbracket \mathsf{Is}_{\mathsf{PRI}^{\mathsf{L}}} \rrbracket(\langle s, L \rangle) = (s = c^{\mathsf{s}})$	
selectors, $f \in \Sigma$:	
$\llbracket f_i^{\mathsf{L}} \rrbracket(\langle s, L \rangle) = \epsilon \langle s_i, L_i \rangle.\ \langle s, L \rangle = \llbracket f^{\mathsf{L}} \rrbracket(\langle s_1, L_1 \rangle, \ldots, \langle s_i, L_i \rangle, \ldots, \langle s_k, L_k \rangle)$	$\llbracket \mathsf{Is}_{f^{\mathsf{L}}} \rrbracket(\langle s, L \rangle)$
leafset shape:	
$\llbracket \mathsf{lssh} \rrbracket(\langle s, L \rangle) = s$	
term algebra on shapes	
constructors, $f \in \Sigma$:	
$\llbracket f^{\mathsf{s}} \rrbracket(s_1, \ldots, s_k) = f^{\mathsf{s}}(s_1, \ldots, s_k)$	
constructor test, $f \in \Sigma_0$:	
$\llbracket \mathsf{Is}_{f^{\mathsf{s}}} \rrbracket(s) = \exists s_1, \ldots, s_k.\ s = f^{\mathsf{s}}(s_1, \ldots, s_k)$	
selectors, $f \in \Sigma$:	
$\llbracket f_i^{\mathsf{s}} \rrbracket(s) = \epsilon s_i.\ s = f^{\mathsf{s}}(s_1, \ldots, s_i, \ldots, s_k)$	$\llbracket \mathsf{Is}_{f^{\mathsf{s}}} \rrbracket(s)$

Figure 11: Semantics for Logic of Term-Power Algebra (Part II)

Recall that $r^= \in R$ is the equality relation on C. Given $r^=$, we can express the equality between terms by:

$$\begin{aligned} t_1 = t_2 &\iff r^{='}(t_1, t_2) \\ &\iff \mathsf{sh}(t_2) = \mathsf{sh}(t_1) \wedge r^=(t_1, t_2) = \mathsf{true}^{\mathsf{l}}_{\mathsf{sh}(t_1)} \end{aligned} \tag{92}$$

We define the notion of a u^{s}-term as in Definition 40 except that we use different symbols for boolean algebra operations.

Definition 72 (u^{s}-terms) *Let $u^{\mathsf{s}} \in \mathsf{Var}^{\mathsf{s}}$ be a shape variable. The set of u^{s}-terms $\mathsf{Term}(u^{\mathsf{s}})$ is the least set such that:*

1. $u^{\mathsf{L}} \in \mathsf{Term}(u^{\mathsf{s}})$ *for every leafset variable* u^{L};
2. $\mathsf{false}^{\mathsf{l}}_{u^{\mathsf{s}}}, \mathsf{true}^{\mathsf{l}}_{u^{\mathsf{s}}} \in \mathsf{Term}(u^{\mathsf{s}})$;
3. *if* $t_1^{\mathsf{L}}, t_2^{\mathsf{L}} \in \mathsf{Term}(u^{\mathsf{s}})$, *then also*

$$t_1^{\mathsf{L}} \wedge^{\mathsf{l}}_{u^{\mathsf{s}}} t_2^{\mathsf{L}} \in \mathsf{Term}(u^{\mathsf{s}}),$$
$$t_1^{\mathsf{L}} \vee^{\mathsf{l}}_{u^{\mathsf{s}}} t_2^{\mathsf{L}} \in \mathsf{Term}(u^{\mathsf{s}}), \text{ and}$$
$$\neg^{\mathsf{l}}_{u^{\mathsf{s}}} t_1^{\mathsf{L}} \in \mathsf{Term}(u^{\mathsf{s}})$$

If t^{s} is a term of shape sort, the notion of t^{s}-inner formula is defined as follows.

Definition 73 (u^{s}-inner formula) *Let $u^{\mathsf{s}} \in \mathsf{Var}^{\mathsf{s}}$ be a shape variable. The set of u^{s}-inner formulas $\mathsf{Inner}(u^{\mathsf{s}})$ is the least set such that:*

1. *if $u_1, \ldots, u_k$ are term variables and $r \in L_C$ such that $\mathsf{ar}(r) = k$, then*

$$r_{u^{\mathsf{s}}}(u_1, \ldots, u_k) \in \mathsf{Inner}(u^{\mathsf{s}})$$

2. $\mathsf{false}^{\mathsf{l}}_{u^{\mathsf{s}}}, \mathsf{true}^{\mathsf{l}}_{u^{\mathsf{s}}} \in \mathsf{Inner}(u^{\mathsf{s}})$
3. *if $\phi_1, \phi_2 \in \mathsf{Inner}(u^{\mathsf{s}})$ then also*

$$\phi_1 \wedge^{\mathsf{l}}_{u^{\mathsf{s}}} \phi_2 \in \mathsf{Inner}(u^{\mathsf{s}})$$
$$\phi_1 \vee^{\mathsf{l}}_{u^{\mathsf{s}}} \phi_2 \in \mathsf{Inner}(u^{\mathsf{s}})$$
$$\neg^{\mathsf{l}}_{u^{\mathsf{s}}} \phi_1 \in \mathsf{Inner}(u^{\mathsf{s}})$$

4. *if $\phi \in \mathsf{Inner}(u^{\mathsf{s}})$ and u is a term variable that does not occur in u^{s}, then also*

$$\exists^{\mathsf{l}}_{u^{\mathsf{s}}} u.\phi \in \mathsf{Inner}(u^{\mathsf{s}})$$
$$\forall^{\mathsf{l}}_{u^{\mathsf{s}}} u.\phi \in \mathsf{Inner}(u^{\mathsf{s}})$$

If $\phi \in \mathsf{Inner}(u^{\mathsf{s}})$ and $u_1, \ldots, u_n$ is the set of free term variables of ϕ, we write $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ for ϕ. Furthermore, if t^{s} is a term of shape *sort and $t_1, \ldots, t_n$ terms of* term *sort, we write $\phi(t^{\mathsf{s}}, t_1, \ldots, t_n)$ for*

$$\phi[u^{\mathsf{s}} := t^{\mathsf{s}}, u_1 := t_1, \ldots, u_n := t_n]$$

where we assume that variables bound by $\exists^{\mathsf{l}}_{\_}$ and $\forall^{\mathsf{l}}_{\_}$ are renamed to avoid the capture of variables that are free in $t^{\mathsf{s}}, t_1, \ldots, t_n$.

We call $\phi(t^{\mathsf{s}}, t_1, \ldots, t_n)$ an instance *of the u^{s}-inner formula $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$.*

If $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ is an inner formula, we abbreviate it by writing $[\phi'(u_1, \ldots, u_n)]_{u^{\mathsf{s}}}$ where ϕ' results from $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ by omitting the shape argument u^{s} from the operations occurring in $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$. Similarly, we write $[\phi'(t_1, \ldots, t_n)]_{t^{\mathsf{s}}}$ for $\phi(t^{\mathsf{s}}, t_1, \ldots, t_n)$.

According to the semantics in Figure 10, sh is a homomorphism from the term algebra of terms to the term algebra of shapes. In addition, lssh is a homomorphism from the term algebra of leafsets to the term algebra of shapes.

We also have the following important property. Let $r \in L_C$ be a relation symbol of arity n, let $f \in \Sigma$ be a function symbol of arity k, and let

$$\mathsf{sh}(t_{1j}) = \ldots = \mathsf{sh}(t_{nj}) = s_j$$

for $1 \leq j \leq k$. If $f^{\mathsf{s}} = \mathsf{shapified}(f)$, $f^{\mathsf{L}} = \mathsf{leafified}(f)$, and $s = f^{\mathsf{s}}(s_1, \ldots, s_k)$ then

$$\begin{aligned} &r_s(f(t_{11}, \ldots, t_{1k}), \ldots, f(t_{n1}, \ldots, t_{nk})) = \\ &\quad f^{\mathsf{L}}(r_{s_1}(t_{11}, \ldots, t_{n1}), \ldots, r_{s_k}(t_{1k}, \ldots, t_{nk})) \end{aligned} \tag{93}$$

Furthermore, if $\mathsf{lssh}(l_j) = \mathsf{lssh}(l'_j) = s_j$ for $1 \leq j \leq k$ and $s = f^{\mathsf{s}}(s_1, \ldots, s_k)$ then

$$\begin{aligned} &f^{\mathsf{L}}(l_1, \ldots, l_k) \wedge^{\mathsf{l}}_s f^{\mathsf{L}}(l'_1, \ldots, l'_k) =^{\mathsf{L}} \\ &\quad f^{\mathsf{L}}(l_1 \wedge^{\mathsf{l}}_{s_1} l'_1, \ldots, l_k \wedge^{\mathsf{l}}_{s_k} l'_k) \\ &f^{\mathsf{L}}(l_1, \ldots, l_k) \vee^{\mathsf{l}}_s f^{\mathsf{L}}(l'_1, \ldots, l'_k) =^{\mathsf{L}} \\ &\quad f^{\mathsf{L}}(l_1 \wedge^{\mathsf{l}}_{s_1} l'_1, \ldots, l_k \wedge^{\mathsf{l}}_{s_k} l'_k) \\ &\neg^{\mathsf{l}}_s f^{\mathsf{L}}(l_1, \ldots, l_k) =^{\mathsf{L}} \\ &\quad f^{\mathsf{L}}(\neg^{\mathsf{l}}_{s_1} l_1, \ldots, \neg^{\mathsf{l}}_{s_k} l_k) \\ &\exists^{\mathsf{l}}_s t. f^{\mathsf{L}}(h_1(t), \ldots, h_k(t)) =^{\mathsf{L}} \\ &\quad f^{\mathsf{L}}(\exists^{\mathsf{l}}_{s_1} t.h_1(t), \ldots, \exists^{\mathsf{l}}_{s_k} t.h_k(t)) \\ &\forall^{\mathsf{l}}_s t. f^{\mathsf{L}}(h_1(t), \ldots, h_k(t)) =^{\mathsf{L}} \\ &\quad f^{\mathsf{L}}(\forall^{\mathsf{l}}_{s_1} t.h_1(t), \ldots, \forall^{\mathsf{l}}_{s_k} t.h_k(t)) \end{aligned} \tag{94}$$

From these properties by induction we conclude that if $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ is an inner formula, then

$$\begin{aligned} &\phi(s, f(t_{11}, \ldots, t_{1k}), \ldots, f(t_{n1}, \ldots, t_{nk})) = \\ &\quad f^{\mathsf{L}}(\phi(s_1, t_{11}, \ldots, t_{n1}), \ldots, \phi(s_k, t_{1k}, \ldots, t_{nk})) \end{aligned} \tag{95}$$

Let $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ be an inner formula and let $\phi'(u_1, \ldots, u_n)$ be a first-order formula that results from replacing operations $\wedge^{\mathsf{l}}_s, \vee^{\mathsf{l}}_s, \neg^{\mathsf{l}}_s, \forall^{\mathsf{l}}_s, \exists^{\mathsf{l}}_s$ by $\wedge, \vee, \neg, \forall, \exists$. Interpreting $\phi'(u_1, \ldots, u_n)$ over the structure $\mathcal{C}$ yields a relation $\rho' \subseteq C^n$. If

$$\mathsf{sh}(t_1) = \ldots = \mathsf{sh}(t_k) = s$$

then

$$[\![\phi]\!](s, t_1, \ldots, t_k) = \langle s, \{l \mid \rho'(t_1[l], \ldots, t_k[l])\}\rangle$$

The following Definition 75 introduces a more restricted set of formulas than the set of formulas permitted by sort declarations in Figure 9. We call this restricted set of formulas *simple formulas.* One of the main properties of simple formulas compared to arbitrary formulas is that simple formulas allow the use of operations $\exists^{\mathsf{l}}_{\_}, \forall^{\mathsf{l}}_{\_}$, and relations $r_{\_}$, $r \in L_C$ only within instances of u^{s}-inner formulas.

Definition 74 *A simple operation is any operation or relation in Figure 9 except for operations $\exists^{\mathsf{l}}_{\_}, \forall^{\mathsf{l}}_{\_}$, and relations $r_{\_}$ for $r \in L_C$.*

Definition 75 (Simple Formulas) *The set of simple formulas is the least set that satisfies the following.*

1. *if $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ is a an inner formula, t^{s} a term of* shape *sort, $t_1, \ldots, t_n$ terms of* term *sort and u^{L} is a leafset variable, then*

$$u^{\mathsf{L}} =^{\mathsf{L}} \phi(t^{\mathsf{s}}, t_1, \ldots, t_n)$$

is a simple formula.

2. *applying simple operations to simple formulas yields simple formulas.*

Example 76 A formula

$$u^{\mathsf{L}} =^{\mathsf{L}} \exists^{\mathsf{l}}_{u_1^{\mathsf{s}}} u.\ r_{u_2^{\mathsf{s}}}(u, u) \tag{96}$$

is not a simple formula for $u_1^{\mathsf{s}} \not\equiv u_2^{\mathsf{s}}$. Formula

$$\begin{aligned}&(u_1^{\mathsf{s}} = u_2^{\mathsf{s}} \ \wedge\ u^{\mathsf{L}} =^{\mathsf{L}} \exists^{\mathsf{l}}_{u_1^{\mathsf{s}}} u.\ r_{u_1^{\mathsf{s}}}(u, u)) \vee \\ &(u_1^{\mathsf{s}} \neq u_2^{\mathsf{s}} \wedge \mathsf{undef})\end{aligned}$$

is a simple formula equivalent to formula (96). We abbreviate $\exists^{\mathsf{l}}_{u_1^{\mathsf{s}}} u.\ r_{u_1^{\mathsf{s}}}(u, u)$ as $[\exists^{\mathsf{l}} u.\ r(u, u)]_{u_1^{\mathsf{s}}}$.

♦

Lemma 77 shows that for every formula in the logic of Figure 9 there exists an equivalent simple formula. Note that even simple formulas are sufficient to express the relations of structural subtyping. A reader not interested in the decidability of the more general logic of Figure 9 may therefore ignore Lemma 77.

Lemma 77 (Formula Simplification) *For every well-defined formula in the logic of Figure 9 there exists an equivalent well-defined simple formula.*

Proof Sketch. According to the definition of simple formula, we need to ensure that every occurrence of quantifiers $\forall^{\mathsf{l}}_{\_}, \exists^{\mathsf{l}}_{\_}$ and relations $r_{\_}$ is an occurrence in some inner-formula instance $\phi(t^{\mathsf{s}}, t_1, \ldots, t_n)$. Each occurrence $r_{t^{\mathsf{s}}}(t_1, \ldots, t_n)$ is an inner formula instance by itself, so the main difficulty is fitting the quantifiers $\forall^{\mathsf{l}}_{\_}$ and $\exists^{\mathsf{l}}_{\_}$ into inner formulas.

Let us examine the syntactic structure of formulas of logic in Figure 9. This syntactic structure is determined by sort declarations. Each expression of leafset is formed starting from

1. relations $r \in L_C$;
2. leafset variables;
3. $\mathsf{true}^{\mathsf{l}}_{\_}, \mathsf{false}^{\mathsf{l}}_{\_}$

using operations $\wedge^{\mathsf{l}}_{\_}$, $\vee^{\mathsf{l}}_{\_}$, $\neg^{\mathsf{l}}_{\_}$, $\forall^{\mathsf{l}}_{\_}$, $\exists^{\mathsf{l}}_{\_}$, as well as f^{L} and f_i^{L}. The leafset expressions can be used in a formula in the following ways (in addition to constructing new leafset expressions):

1. to compare for equality using $=^{\mathsf{L}}$;
2. to test for the top-level constructor using $\mathsf{Is}_{f^{\mathsf{L}}}$;
3. to form leafset cardinality constraints;
4. to form a shape using lssh.

Because the top-level sort of a formula is bool, every term t_0^{L} of sort leafset occurs within some formula $t_1^{\mathsf{L}} =^{\mathsf{L}} t_2^{\mathsf{L}}$ or $\mathsf{Is}_{f^{\mathsf{L}}}(t^{\mathsf{L}})$, $|t^{\mathsf{L}}|_{t^{\mathsf{s}}} = k$, $|t^{\mathsf{L}}|_{t^{\mathsf{s}}} \geq k$ or as part of some term $\mathsf{lssh}(t^{\mathsf{L}})$. We can replace $\mathsf{Is}_{f^{\mathsf{L}}}(t^{\mathsf{L}})$ with

$$\exists u^{\mathsf{L}}.\ u^{\mathsf{L}} =^{\mathsf{L}} t^{\mathsf{L}} \wedge' \mathsf{Is}_{f^{\mathsf{L}}}(u^{\mathsf{L}})$$

according to Lemma 10, so we need not consider that case. We can similarly eliminate non-variable leafset terms from cardinality constraints. If a leafset term t^{L} occurs in an expression $\mathsf{lssh}(t^{\mathsf{L}})$, we consider the smallest atomic formula $\psi(\mathsf{lssh}(t^{\mathsf{L}}))$ enclosing $\mathsf{lssh}(t^{\mathsf{L}})$, and replace $\psi(t^{\mathsf{L}})$ with

$$\exists u^{\mathsf{L}}.\ u^{\mathsf{L}} =^{\mathsf{L}} t^{\mathsf{L}} \wedge' \psi(u^{\mathsf{L}})$$

This transformation is valid by Lemma 10 because ψ and lssh are strict.

We further assume that in every atomic formula $t_1^{\mathsf{L}} =^{\mathsf{L}} t_2^{\mathsf{L}}$, the term t_1^{L} is a leafset variable.

Suppose that a term t^{L} in a formula $u^{\mathsf{L}} =^{\mathsf{L}} t^{\mathsf{L}}$ is not an instance of an inner formula. Then there are two possibilities.

1. There are some occurrences of leafset term algebra operations f^{L}, f_i^{L} or leafset variables u_1^{L} in t^{L}. Here by "occurrence" in t^{L} we mean occurrence that is reachable without going through a shape argument or a relation, but only through operations $\forall^{\mathsf{l}}_{\_}, \exists^{\mathsf{l}}_{\_}, \wedge^{\mathsf{l}}_{\_}, \vee^{\mathsf{l}}_{\_}, \neg^{\mathsf{l}}_{\_}$. For example, we ignore the occurrences of f^{L}, f_i^{L} within terms t^{s} that occur in $\wedge^{\mathsf{l}}_{t^{\mathsf{s}}}$.
2. not all shape arguments in $\forall^{\mathsf{l}}_{\_}, \exists^{\mathsf{l}}_{\_}, \wedge^{\mathsf{l}}_{\_}, \vee^{\mathsf{l}}_{\_}, \neg^{\mathsf{l}}_{\_}$, $\mathsf{true}^{\mathsf{l}}_{\_}, \mathsf{false}^{\mathsf{l}}_{\_}$, $r_{\_}$ occurring in t^{L} are syntactically identical.

We eliminate the first possibility by propagating leafset term algebra operations f^{L}, f_i^{L} inwards until they reach expressions of form $\mathsf{L}t^{\mathsf{s}}(t_1, \ldots, t_n)$, applying the equations (94) from left to right. We then convert f^{L}, f_i^{L} operations of term algebra of leafsets into operations of the term algebra of terms applying (93) from right to left.

To eliminate the second possibility, let $t_1^{\mathsf{s}}, \ldots, t_n^{\mathsf{s}}$ be the occurrences (reachable through $\mathsf{true}^{\mathsf{l}}_{\_}, \mathsf{false}^{\mathsf{l}}_{\_}$, $\wedge^{\mathsf{l}}_{\_}$, $\vee^{\mathsf{l}}_{\_}$, $\neg^{\mathsf{l}}_{\_}$, $\forall^{\mathsf{l}}_{\_}$, $\exists^{\mathsf{l}}_{\_}$) in term t^{L} of the shape arguments of operations $\mathsf{true}^{\mathsf{l}}_{\_}, \mathsf{false}^{\mathsf{l}}_{\_}$, $\wedge^{\mathsf{l}}_{\_}$, $\vee^{\mathsf{l}}_{\_}$, $\neg^{\mathsf{l}}_{\_}$, $\forall^{\mathsf{l}}_{\_}$, $\exists^{\mathsf{l}}_{\_}$. Then replace

$$u^{\mathsf{L}} = t^{\mathsf{L}}(t_1^{\mathsf{s}}, \ldots t_n^{\mathsf{s}})$$

with

$$\begin{aligned}&(\exists^{\mathsf{s}} u^{\mathsf{s}}.\ \forall_1^{\mathsf{CL}}(u^{\mathsf{s}} =^{\mathsf{s}} t_1^{\mathsf{s}}) \wedge' \ldots \wedge' \forall_n^{\mathsf{CL}}(u^{\mathsf{s}} =^{\mathsf{s}} t_n^{\mathsf{s}}) \wedge' \\ &\qquad u^{\mathsf{L}} =^{\mathsf{L}} t^{\mathsf{L}}(u^{\mathsf{s}}, \ldots, u^{\mathsf{s}})) \vee \\ &(\mathsf{undef}\ \wedge\ \textstyle\bigvee_{1 \leq i < j \leq n} t_i^{\mathsf{s}} \neq t_j^{\mathsf{s}})\end{aligned}$$

Here $\forall_i^{\mathsf{CL}}$ denotes universal quantification $\forall u_{i,1}, \ldots, u_{i,n_i}$ where $u_{i,1}, \ldots, u_{i,n_i}$ is a list of those term variables occurring in t_i^{s} that are bound by some quantifier $\exists^{\mathsf{l}}_{\_}, \forall^{\mathsf{l}}_{\_}$ within t^{L}. ∎

6.4 Quantifier Elimination

In this section we give a quantifier elimination procedure for the term-power structure. The procedure of this section is applicable whenever $\mathcal{C}$ is a structure with a decidable first-order theory.

Definition 78 below generalizes the notion of structural base formula of Definition 41, Section 4.3. There are two main differences between Definition 41 and the present Definition 78.

The first difference is the presence of three (instead of two) base formulas: shape base, leafset base, and term base. This difference is a consequence of the distinction between leafsets and terms and is needed whenever base structure $\mathcal{C}$ has more than two elements. There is a homomorphism formula relating leafset base formula to shape base formula and a homomorphism formula relating term base formula to shape base formula. Furthermore, some of the leafset variables are determined by term variables using inner formula maps, which establishes the relationship between term base formula and leafset base formula. Cardinality constraints now apply to leafset variables.

The second difference is the distinction between composed and primitive non-parameter leafset and term variables. A composed non-parameter variable denotes a leafset or a term whose shape s has property $\mathsf{Is}_{f^{\mathsf{s}}}(s)$ for some $f \in \Sigma$. A primitive non-parameter variable denotes a leafset or a term whose shape is c^{s} and has property $\mathsf{Is}_{\mathsf{PRI}}$ or $\mathsf{Is}_{\mathsf{PRI}^{\mathsf{L}}}$. The purpose of this distinction is to allow cardinality constraints and inner formula maps not only on parameter variables, but also on primitive non-parameter variables, which is useful when the base structure $\mathcal{C}$ is decidable but infinite.

Definition 78 (Structural Base Formula)
A structural base formula *with:*

- *free term variables* $x_1, \ldots, x_m$*;*
- *internal composed non-parameter term variables* $u_1, \ldots, u_r$*;*
- *internal primitive non-parameter term variables* $u_{r+1}, \ldots, u_p$*;*
- *internal parameter term variables* $u_{p+1}, \ldots, u_{p+q}$*;*
- *free leafset variables* $x^{\mathsf{L}}_1, \ldots, x^{\mathsf{L}}_{m^{\mathsf{L}}}$*;*
- *internal composed non-parameter leafset variables* $u^{\mathsf{L}}_1, \ldots, u^{\mathsf{L}}_{r^{\mathsf{L}}}$*;*
- *internal primitive non-parameter leafset variables* $u^{\mathsf{L}}_{r^{\mathsf{L}}+1}, \ldots, u^{\mathsf{L}}_{p^{\mathsf{L}}}$*;*
- *internal parameter leafset variables* $u_{p^{\mathsf{L}}+1}, \ldots, u_{p^{\mathsf{L}}+q^{\mathsf{L}}}$*;*
- *free shape variables* $x^{\mathsf{s}}_1, \ldots, x^{\mathsf{s}}_{m^{\mathsf{s}}}$*;*
- *internal non-parameter shape variables* $u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{p^{\mathsf{s}}}$*;*
- *internal parameter shape variables* $u^{\mathsf{s}}_{p^{\mathsf{s}}}, \ldots, u^{\mathsf{s}}_{p^{\mathsf{s}}+q^{\mathsf{s}}}$

is a formula of form:

$$
\begin{aligned}
&\exists u_1, \ldots, u_n, u^{\mathsf{L}}_1, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}}, u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}. \\
&\quad \mathsf{shapeBase}(u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}, x^{\mathsf{s}}_1, \ldots, x^{\mathsf{s}}_{m^{\mathsf{s}}}) \land \\
&\quad \mathsf{leafsetBase}(u^{\mathsf{L}}_1, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}}, x^{\mathsf{L}}_1, \ldots, x^{\mathsf{L}}_{m^{\mathsf{L}}}) \land \\
&\quad \mathsf{leafsetHom}(u^{\mathsf{L}}_1, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}}, u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}) \land \\
&\quad \mathsf{termBase}(u_1, \ldots, u_n, x_1, \ldots, x_m) \land \\
&\quad \mathsf{termHom}(u_1, \ldots, u_n, u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}) \land \\
&\quad \mathsf{cardin}(u^{\mathsf{L}}_{r^{\mathsf{L}}+1}, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}}, u^{\mathsf{s}}_{p^{\mathsf{s}}+1}, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}) \land \\
&\quad \mathsf{innerMap}(u_{r+1}, \ldots, u_n, u^{\mathsf{L}}_{r^{\mathsf{L}}+1}, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}}, u^{\mathsf{s}}_{p^{\mathsf{s}}+1}, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}})
\end{aligned}
$$

where $n = p + q$*,* $n^{\mathsf{L}} = p^{\mathsf{L}} + q^{\mathsf{L}}$*,* $n^{\mathsf{s}} = p^{\mathsf{s}} + q^{\mathsf{s}}$*, and formulas* shapeBase, leafsetBase, termBase, leafsetHom, termHom, cardin, innerMap *are defined as follows.*

$$
\begin{aligned}
&\mathsf{shapeBase}(u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}, x^{\mathsf{s}}_1, \ldots, x^{\mathsf{s}}_{m^{\mathsf{s}}}) = \\
&\quad \bigwedge_{i=1}^{p^{\mathsf{s}}} u^{\mathsf{s}}_i = t_i(u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}) \ \land\ \bigwedge_{i=1}^{m^{\mathsf{s}}} x^{\mathsf{s}}_i = u^{\mathsf{s}}_{j_i} \\
&\quad \land\ \mathsf{distinct}(u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_n)
\end{aligned}
$$

where each t_i *is a shape term of form* $f^{\mathsf{s}}(u^{\mathsf{s}}_{i_1}, \ldots, u^{\mathsf{s}}_{i_k})$ *for some* $f \in \Sigma_0$*,* $k = \mathsf{ar}(f)$*, and* $j : \{1, \ldots, m^{\mathsf{s}}\} \to \{1, \ldots, n^{\mathsf{s}}\}$ *is a function mapping indices of free shape variables to indices of internal shape variables.*

$$
\begin{aligned}
&\mathsf{leafsetBase}(u^{\mathsf{L}}_1, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}}, x^{\mathsf{L}}_1, \ldots, x^{\mathsf{L}}_{m^{\mathsf{L}}}) = \\
&\quad \bigwedge_{i=1}^{r^{\mathsf{L}}} \quad u^{\mathsf{L}}_i = t_i(u^{\mathsf{L}}_1, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}}) \ \land \\
&\quad \bigwedge_{i=r^{\mathsf{L}}+1}^{p^{\mathsf{L}}} \quad \mathsf{Is}_{\mathsf{PRI}^{\mathsf{L}}}(u^{\mathsf{L}}_i) \ \land \\
&\quad \bigwedge_{i=1}^{m^{\mathsf{L}}} \quad x^{\mathsf{L}}_i = u^{\mathsf{L}}_{j_i}
\end{aligned}
$$

where each t_i *is a term of form* $f(u_{i_1}, \ldots, u_{i_k})$ *for some* $f \in \Sigma$*,* $k = \mathsf{ar}(f)$*, and* $j : \{1, \ldots, m^{\mathsf{L}}\} \to \{1, \ldots, n^{\mathsf{L}}\}$ *is a function mapping indices of free leafset variables to indices of internal leafset variables.*

$$
\mathsf{leafsetHom}(u^{\mathsf{L}}_1, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}}, u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}) = \bigwedge_{i=1}^{n^{\mathsf{L}}} \mathsf{lssh}(u^{\mathsf{L}}_i) = u^{\mathsf{s}}_{j_i}
$$

where $j : \{1, \ldots, n^{\mathsf{L}}\} \to \{1, \ldots, n^{\mathsf{s}}\}$ *is some function such that* $\{j_1, \ldots, j_p\} \subseteq \{1, \ldots, p^{\mathsf{s}}\}$ *and* $\{j_{p^{\mathsf{L}}+1}, \ldots, j_{p^{\mathsf{L}}+q^{\mathsf{L}}}\} \subseteq \{p^{\mathsf{s}}+1, \ldots, p^{\mathsf{s}}+q^{\mathsf{s}}\}$ *(a leafset variable is a parameter variable iff its shape is a parameter shape variable).*

$$
\begin{aligned}
&\mathsf{termBase}(u_1, \ldots, u_n, x_1, \ldots, x_m) = \\
&\quad \bigwedge_{i=1}^{r} \quad u_i = t_i(u_1, \ldots, u_n) \ \land \\
&\quad \bigwedge_{i=r+1}^{p} \quad \mathsf{Is}_{\mathsf{PRI}}(u_i) \ \land \\
&\quad \bigwedge_{i=1}^{m} \quad x_i = u_{j_i}
\end{aligned}
$$

where each t_i *is a term of form* $f(u_{i_1}, \ldots, u_{i_k})$ *for some* $f \in \Sigma$*,* $k = \mathsf{ar}(f)$*, and* $j : \{1, \ldots, m\} \to \{1, \ldots, n\}$ *is a function mapping indices of free term variables to indices of internal term variables.*

$$
\mathsf{termHom}(u_1, \ldots, u_n, u^{\mathsf{s}}_1, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}) = \bigwedge_{i=1}^{n} \mathsf{sh}(u_i) = u^{\mathsf{s}}_{j_i}
$$

where $j : \{1, \ldots, n\} \to \{1, \ldots, n^{\mathsf{s}}\}$ *is some function such that* $\{j_1, \ldots, j_p\} \subseteq \{1, \ldots, p^{\mathsf{s}}\}$ *and* $\{j_{p+1}, \ldots, j_{p+q}\} \subseteq \{p^{\mathsf{s}} + 1, \ldots, p^{\mathsf{s}} + q^{\mathsf{s}}\}$ *(a term variable is a parameter variable iff its shape is a parameter shape variable).*

$$
\mathsf{cardin}(u^{\mathsf{L}}_{r^{\mathsf{L}}+1}, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}}, u^{\mathsf{s}}_{p^{\mathsf{s}}+1}, \ldots, u^{\mathsf{s}}_{n^{\mathsf{s}}}) = \psi_1 \land \cdots \land \psi_d
$$

where each ψ_i *is of form*

$$
|t^{\mathsf{L}}(u^{\mathsf{L}}_{r^{\mathsf{L}}+1}, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}})|_{u^{\mathsf{s}}} = k
$$

or

$$|t^{\mathsf{L}}(u^{\mathsf{L}}_{r^{\mathsf{L}}+1},\ldots,u^{\mathsf{L}}_{n^{\mathsf{L}}})|_{u^{\mathsf{s}}} \geq k$$

for some u^{s}*-term* $t^{\mathsf{L}}(u^{\mathsf{L}}_{r^{\mathsf{L}}+1},\ldots,u^{\mathsf{L}}_{n^{\mathsf{L}}})$ *that contains no variables other than some of the variables* $u^{\mathsf{L}}_{r^{\mathsf{L}}+1},\ldots,u^{\mathsf{L}}_{n^{\mathsf{L}}}$, *and the following condition holds:*

If a variable u^{L}_j *for* $r^{\mathsf{L}}+1 \leq j \leq n^{\mathsf{L}}$ *occurs in the term* $t^{\mathsf{L}}(u^{\mathsf{L}}_{r^{\mathsf{L}}+1},\ldots,u^{\mathsf{L}}_{n^{\mathsf{L}}})$, *then* $\mathsf{lssh}(u^{\mathsf{L}}_j) = u^{\mathsf{s}}$ *occurs in formula* leafsetHom. (97)

$$\mathsf{innerMap}(u_{r+1},\ldots,u_n,u^{\mathsf{L}}_{r^{\mathsf{L}}+1},\ldots,u^{\mathsf{L}}_{n^{\mathsf{L}}},u^{\mathsf{s}}_{p^{\mathsf{s}}+1},\ldots,u^{\mathsf{s}}_{n^{\mathsf{s}}}) = \eta_1 \wedge \cdots \wedge \eta_e$$

where each η_i *is of form*

$$u^{\mathsf{L}}_j = \phi^I(u^{\mathsf{s}},u_{i_1},\ldots,u_{i_k})$$

for some inner formula $\phi^I(u^{\mathsf{s}},u_{i_1},\ldots,u_{i_k}) \in \mathsf{Inner}(u^{\mathsf{s}})$ *where* $\mathsf{L}+1 \leq j \leq n^{\mathsf{L}}$ *i.e.* u^{L}_j *is a primitive non-parameter leafset variable or parameter leafset variable,* $\{u_{i_1},\ldots,u_{i_k}\} \subseteq \{u_{r+1},\ldots,u_n\}$ *are primitive non-parameter term variables and parameter variables, the conjunct* $\mathsf{lssh}(u^{\mathsf{L}}) = u^{\mathsf{s}}$ *occurs in* leafsetHom, *and the following condition holds:*

$\mathsf{sh}(u_{i_j}) = u^{\mathsf{s}}$ *occurs in formula* termHom *for every* j *where* $1 \leq j \leq k$. (98)

We require each structural base formula to satisfy the following conditions:

P0) the graph associated with shape base formula

$$\exists u^{\mathsf{s}}_1,\ldots,u^{\mathsf{s}}_{n^{\mathsf{s}}}.\ \mathsf{shapeBase}(u^{\mathsf{s}}_1,\ldots,u^{\mathsf{s}}_{n^{\mathsf{s}}},x^{\mathsf{s}}_1,\ldots,x^{\mathsf{s}}_{m^{\mathsf{s}}})$$

is acyclic (compare to Definition 21);

P1) congruence closure property for shapeBase *subformula: there are no two distinct variables* u^{s}_i *and* u^{s}_j *such that both* $u^{\mathsf{s}}_i = f(u^{\mathsf{s}}_{l_1},\ldots,u^{\mathsf{s}}_{l_k})$ *and* $u^{\mathsf{s}}_j = f(u^{\mathsf{s}}_{l_1},\ldots,u^{\mathsf{s}}_{l_k})$ *occur as conjuncts in formula* shapeBase*;*

P2) congruence closure property for leafsetBase *subformula: there are no two distinct variables* u^{L}_i *and* u^{L}_j *such that both* $u^{\mathsf{L}}_i = f^{\mathsf{L}}(u^{\mathsf{L}}_{l_1},\ldots,u^{\mathsf{L}}_{l_k})$ *and* $u^{\mathsf{L}}_j = f^{\mathsf{L}}(u^{\mathsf{L}}_{l_1},\ldots,u^{\mathsf{L}}_{l_k})$ *occur as conjuncts in formula* leafsetBase*;*

P3) congruence closure property for termBase *subformula: there are no two distinct variables* u_i *and* u_j *such that both* $u_i = f(u_{l_1},\ldots,u_{l_k})$ *and* $u_j = f(u_{l_1},\ldots,u_{l_k})$ *occur as conjuncts in formula* termBase*;*

P4) homomorphism property of lssh*: for every non-parameter leafset variable* u^{L} *such that* $u^{\mathsf{L}} = f^{\mathsf{L}}(u^{\mathsf{L}}_{i_1},\ldots,u^{\mathsf{L}}_{i_k})$ *occurs in* leafsetBase, *if conjunct* $\mathsf{lssh}(u^{\mathsf{L}}) = u^{\mathsf{s}}$ *occurs in* leafsetHom, *then for some shape variables* $u^{\mathsf{s}}_{j_1},\ldots,u^{\mathsf{s}}_{j_k}$ *term* $u^{\mathsf{s}} = f^{\mathsf{s}}(u^{\mathsf{s}}_{j_1},\ldots,u^{\mathsf{s}}_{j_k})$ *occurs in* shapeBase *where* $f^{\mathsf{s}} = \mathsf{shapified}(f)$ *and for every* r *where* $1 \leq r \leq k$, *conjunct* $\mathsf{lssh}(u_{i_r}) = u^{\mathsf{s}}_{j_r}$ *occurs in* leafsetHom.

P5) homomorphism property of sh*: for every non-parameter term variable* u *such that* $u = f(u_{i_1},\ldots,u_{i_k})$ *occurs in* termBase, *if conjunct* $\mathsf{sh}(u) = u^{\mathsf{s}}$ *occurs in* termHom, *then for some shape variables* $u^{\mathsf{s}}_{j_1},\ldots,u^{\mathsf{s}}_{j_k}$ *term* $u^{\mathsf{s}} = f^{\mathsf{s}}(u^{\mathsf{s}}_{j_1},\ldots,u^{\mathsf{s}}_{j_k})$ *occurs in* shapeBase *where* $f^{\mathsf{s}} = \mathsf{shapified}(f)$ *and for every* r *where* $1 \leq r \leq k$, *conjunct* $\mathsf{sh}(u_{i_r}) = u^{\mathsf{s}}_{j_r}$ *occurs in* termHom.

As in Section 3.4 and Section 4.3 we proceed to show that each quantifier-free formula can be written as a disjunction of base formulas and each base formula can be written as a quantifier-free formula. We first give a small example to illustrate how the techniques of Section 4.3 extend to the more general case of Σ-term-power.

Example 79 We solve one subproblem from Example 42 using the language of term-power algebras.

Consider the formula

$$\begin{aligned}&\exists v.\ g(v,z) \leq g(z,v)\ \wedge\ \mathsf{ls}_g(v)\ \wedge\ \mathsf{ls}_g(w)\ \wedge\\ &\qquad \neg(g_1(w) \leq g_1(v))\end{aligned} \tag{99}$$

Formula (99) is in the language of Figure 8, with $\leq$ a binary lifted relation. After converting (99) into the language of Figure 9 we obtain as one of the possible cases formula:

$$\begin{aligned}\exists v.\ &[g(v,z) \preceq g(z,v)]_{\mathsf{sh}(g(z,v))} =^{\mathsf{L}} \mathsf{true}^{\mathsf{l}}_{\mathsf{sh}(g(z,v))}\ \wedge\\ &\mathsf{sh}(g(z,v)) =^{\mathsf{s}} \mathsf{sh}(g(z,v))\ \wedge\\ &\mathsf{ls}_g(v)\ \wedge\ \mathsf{ls}_g(w)\ \wedge\\ &[g_1(w) \preceq g_1(v)]_{\mathsf{sh}(g_1(w))} \neq^{\mathsf{L}} \mathsf{true}^{\mathsf{l}}_{\mathsf{sh}(g_1(w))}\ \wedge\\ &\mathsf{sh}(g_1(v)) =^{\mathsf{s}} \mathsf{sh}(g_1(w))\end{aligned} \tag{100}$$

where $\preceq$ is the subtyping relation on the base structure C so that $\leq = \preceq'$. We next transform the formula into unnested form, obtaining:

$$\begin{aligned}&\exists v, u_{vz}, u_{zv}, u_{w1}, u_{v1}.\ \exists^{\mathsf{L}} u^{\mathsf{L}}_{vz}, u^{\mathsf{L}}_{w1}.\ \exists^{\mathsf{s}} u^{\mathsf{s}}_{vz}, u^{\mathsf{s}}_{w1}.\\ &\quad u_{vz} = g(v,z)\ \wedge\ u_{zv} = g(z,v)\ \wedge\\ &\quad u_{w1} = g_1(w)\ \wedge\ u_{v1} = g_1(v)\ \wedge\\ &\quad u^{\mathsf{s}}_{vz} =^{\mathsf{s}} \mathsf{sh}(u_{vz})\ \wedge\ u^{\mathsf{s}}_{w1} =^{\mathsf{s}} \mathsf{sh}(u_{w1})\ \wedge\\ &\quad \mathsf{sh}(u_{zv}) =^{\mathsf{s}} u^{\mathsf{s}}_{vz}\ \wedge\ \mathsf{sh}(u_{v1}) =^{\mathsf{s}} u^{\mathsf{s}}_{w1}\ \wedge\\ &\quad \mathsf{ls}_g(v)\ \wedge\ \mathsf{ls}_g(w)\ \wedge\\ &\quad u^{\mathsf{L}}_{vz} =^{\mathsf{L}} [u_{vz} \preceq u_{zv}]_{u^{\mathsf{s}}_{vz}}\\ &\quad |\neg u^{\mathsf{L}}_{vz}|_{u^{\mathsf{s}}_{vz}} = 0\ \wedge\\ &\quad u^{\mathsf{L}}_{w1} =^{\mathsf{L}} [u_{w1} \preceq u_{v1}]_{u^{\mathsf{s}}_{w1}}\ \wedge\\ &\quad |\neg u^{\mathsf{L}}_{w1}| \geq 1\end{aligned} \tag{101}$$

We next transform (101) into disjunction of base formulas. A typical base formula is:

$$\begin{aligned}&\exists u_{vz}, u_{zv}, u_v, u_z, u_w, u_{v1}, u_{v2}, u_{z1}, u_{z2}, u_{w1}, u_{w2}.\\ &\exists^{\mathsf{L}} u^{\mathsf{L}}_{vz}, u^{\mathsf{L}}_v, u^{\mathsf{L}}_z, u^{\mathsf{L}}_{v1}, u^{\mathsf{L}}_{v2}, u^{\mathsf{L}}_{z1}, u^{\mathsf{L}}_{z2}, u^{\mathsf{L}}_{w1}.\\ &\exists^{\mathsf{s}} u^{\mathsf{s}}_{vz}, u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}.\\ &\quad \mathsf{shapeBase}_1\ \wedge\\ &\quad \mathsf{leafsetBase}_1\ \wedge\ \mathsf{leafsetHom}_1\ \wedge\\ &\quad \mathsf{termBase}_1\ \wedge\ \mathsf{termHom}_1\ \wedge\\ &\quad \mathsf{cardin}_1\ \wedge\ \mathsf{innerMap}_1\end{aligned} \tag{102}$$

$$\begin{aligned}\mathsf{shapeBase}_1 = \ &u^{\mathsf{s}}_{vz} = g^{\mathsf{s}}(u^{\mathsf{s}}_w, u^{\mathsf{s}}_w) \wedge u^{\mathsf{s}}_w = g^{\mathsf{s}}(u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2})\ \wedge\\ &\mathsf{distinct}(u^{\mathsf{s}}_{vz}, u^{\mathsf{s}}_w, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2})\end{aligned}$$

$$
\begin{aligned}
\mathsf{leafsetBase}_1 = \;& u^{\mathsf{L}}_{vz} = g^{\mathsf{L}}(u^{\mathsf{L}}_{v}, u^{\mathsf{L}}_{z}) \wedge \\
& u^{\mathsf{L}}_{v} = g^{\mathsf{L}}(u^{\mathsf{L}}_{v1}, u^{\mathsf{L}}_{v2}) \;\wedge u^{\mathsf{L}}_{z} = g^{\mathsf{L}}(u^{\mathsf{L}}_{z1}, u^{\mathsf{L}}_{z2})
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{leafsetHom}_1 = \;& \mathsf{lssh}(u^{\mathsf{L}}_{vz}) = u^{\mathsf{s}}_{vz} \wedge \\
& \mathsf{lssh}(u^{\mathsf{L}}_{v}) = u^{\mathsf{s}}_{w} \wedge \mathsf{lssh}(u^{\mathsf{L}}_{z}) = u^{\mathsf{s}}_{w} \wedge \\
& \mathsf{lssh}(u^{\mathsf{L}}_{v1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{lssh}(u^{\mathsf{L}}_{v2}) = u^{\mathsf{s}}_{w2} \wedge \\
& \mathsf{lssh}(u^{\mathsf{L}}_{z1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{lssh}(u^{\mathsf{L}}_{z2}) = u^{\mathsf{s}}_{w2} \wedge \\
& \mathsf{lssh}(u^{\mathsf{L}}_{w1}) = u^{\mathsf{s}}_{w1}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{termBase}_1 = \;& u_{vz} = g(u_v, u_z) \wedge u_{zv} = g(u_z, u_v) \wedge \\
& u_v = g(u_{v1}, u_{v2}) \wedge u_z = g(u_{z1}, u_{z2}) \wedge \\
& u_w = g(u_{w1}, u_{w2}) \wedge \\
& z = u_z \wedge w = u_w
\end{aligned}
$$

$$
\begin{aligned}
& \mathsf{termHom}_1 = \\
& \quad \mathsf{sh}(u_{vz}) = u^{\mathsf{s}}_{vz} \wedge \mathsf{sh}(u_{zv}) = u^{\mathsf{s}}_{vz} \wedge \\
& \quad \mathsf{sh}(u_{v}) = u^{\mathsf{s}}_{w} \wedge \mathsf{sh}(u_{z}) = u^{\mathsf{s}}_{w} \wedge \mathsf{sh}(u_{w}) = u^{\mathsf{s}}_{w} \wedge \\
& \quad \mathsf{sh}(u_{v1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{sh}(u_{z1}) = u^{\mathsf{s}}_{w1} \wedge \mathsf{sh}(u_{w1}) = u^{\mathsf{s}}_{w1} \wedge \\
& \quad \mathsf{sh}(u_{v2}) = u^{\mathsf{s}}_{w2} \wedge \mathsf{sh}(u_{z2}) = u^{\mathsf{s}}_{w2} \wedge \mathsf{sh}(u_{w2}) = u^{\mathsf{s}}_{w2} \\
& \mathsf{innerMap}_1 = \\
& \quad u^{\mathsf{L}}_{v1} =^{\mathsf{L}} [u_{v1} \preceq u_{z1}]_{u^{\mathsf{s}}_{w1}} \;\wedge u^{\mathsf{L}}_{z1} =^{\mathsf{L}} [u_{z1} \preceq u_{v1}]_{u^{\mathsf{s}}_{w1}} \;\wedge \\
& \quad u^{\mathsf{L}}_{v2} =^{\mathsf{L}} [u_{v2} \preceq u_{z2}]_{u^{\mathsf{s}}_{w2}} \;\wedge u^{\mathsf{L}}_{z2} =^{\mathsf{L}} [u_{z2} \preceq u_{v2}]_{u^{\mathsf{s}}_{w2}} \;\wedge \\
& \quad u^{\mathsf{L}}_{w1} =^{\mathsf{L}} [u_{w1} \preceq u_{v1}]_{u^{\mathsf{s}}_{w1}} \\
& \qquad \mathsf{cardin}_1 = \; |\neg u^{\mathsf{L}}_{v1}|_{u^{\mathsf{s}}_{w1}} = 0 \;\wedge\; |\neg u^{\mathsf{L}}_{z1}|_{u^{\mathsf{s}}_{w1}} = 0 \wedge \\
& \qquad\qquad |\neg u^{\mathsf{L}}_{v2}|_{u^{\mathsf{s}}_{w2}} = 0 \;\wedge\; |\neg u^{\mathsf{L}}_{z2}|_{u^{\mathsf{s}}_{w2}} = 0 \wedge \\
& \qquad\qquad |\neg u^{\mathsf{L}}_{w1}|_{u^{\mathsf{s}}_{w1}}| \geq 1
\end{aligned}
$$

We next show how to transform the base formula (102) into quantifier-free form.

We substitute away non-parameter term variables u_{vz}, u_{zv}, u_v and non-parameter leafset variables $u^{\mathsf{L}}_{vz}, u^{\mathsf{L}}_{v}, u^{\mathsf{L}}_{z}$, because the homomorphism constraints they participate in may be derived from the remaining conjuncts. We next eliminate parameter term variables u_{v1}, u_{v2} and parameter leafset variables $u^{\mathsf{L}}_{v1}, u^{\mathsf{L}}_{v2}, u^{\mathsf{L}}_{z1}, u^{\mathsf{L}}_{z2}, u^{\mathsf{L}}_{w1}$. Grouping the conjuncts in cardin_1 and $\mathsf{innerMap}_1$ by their shape, we may extract the subformulas ψ_1 and ψ_2 of (102).

$$
\begin{aligned}
& \psi_1 \;\equiv \\
& \exists u_{v1}.\exists^{\mathsf{L}} u^{\mathsf{L}}_{v1}, u^{\mathsf{L}}_{z1}, u^{\mathsf{L}}_{w1}. \\
& \quad \mathsf{sh}(u_{v1}) =^{\mathsf{s}} u^{\mathsf{s}}_{w1} \;\wedge\; \mathsf{sh}(u_{z1}) =^{\mathsf{s}} u^{\mathsf{s}}_{w1} \;\wedge\; \mathsf{sh}(u_{w1}) =^{\mathsf{s}} u^{\mathsf{s}}_{w1} \wedge \\
& \quad \mathsf{lssh}(u^{\mathsf{L}}_{v1}) =^{\mathsf{s}} u^{\mathsf{s}}_{w1} \;\wedge\; \mathsf{lssh}(u^{\mathsf{L}}_{z1}) =^{\mathsf{s}} u^{\mathsf{s}}_{w1} \wedge \\
& \quad \mathsf{lssh}(u^{\mathsf{L}}_{w1}) =^{\mathsf{s}} u^{\mathsf{s}}_{w1} \wedge \\
& \quad u^{\mathsf{L}}_{v1} =^{\mathsf{L}} [u_{v1} \preceq u_{z1}]_{u^{\mathsf{s}}_{w1}} \;\wedge u^{\mathsf{L}}_{z1} =^{\mathsf{L}} [u_{z1} \preceq u_{v1}]_{u^{\mathsf{s}}_{w1}} \;\wedge \\
& \quad u^{\mathsf{L}}_{w1} =^{\mathsf{L}} [u_{w1} \preceq u_{v1}]_{u^{\mathsf{s}}_{w1}} \;\wedge \\
& \quad |\neg u^{\mathsf{L}}_{v1}|_{u^{\mathsf{s}}_{w1}} = 0 \;\wedge\; |\neg u^{\mathsf{L}}_{z1}|_{u^{\mathsf{s}}_{w1}} = 0 \wedge \\
& \quad |\neg u^{\mathsf{L}}_{w1}|_{u^{\mathsf{s}}_{w1}}| \geq 1
\end{aligned}
$$

and

$$
\begin{aligned}
& \psi_2 \;\equiv \\
& \exists u_{v2}.\exists^{\mathsf{L}} u^{\mathsf{L}}_{v2}, u^{\mathsf{L}}_{z2}. \\
& \quad \mathsf{sh}(u_{v2}) =^{\mathsf{s}} u^{\mathsf{s}}_{w2} \;\wedge\; \mathsf{sh}(u_{z2}) =^{\mathsf{s}} u^{\mathsf{s}}_{w2} \wedge \\
& \quad \mathsf{sh}(u^{\mathsf{L}}_{v2}) =^{\mathsf{s}} u^{\mathsf{s}}_{w2} \;\wedge\; \mathsf{sh}(u^{\mathsf{L}}_{z2}) =^{\mathsf{s}} u^{\mathsf{s}}_{w2} \wedge \\
& \quad u^{\mathsf{L}}_{v2} =^{\mathsf{L}} [u_{v2} \preceq u_{z2}]_{u^{\mathsf{s}}_{w2}} \;\wedge u^{\mathsf{L}}_{z2} =^{\mathsf{L}} [u_{z2} \preceq u_{v2}]_{u^{\mathsf{s}}_{w2}} \;\wedge \\
& \quad |\neg u^{\mathsf{L}}_{v2}|_{u^{\mathsf{s}}_{w2}} = 0 \;\wedge\; |\neg u^{\mathsf{L}}_{z2}|_{u^{\mathsf{s}}_{w2}} = 0
\end{aligned}
$$

Formula ψ_1 expresses a fact in a structure isomorphic to the power $\mathcal{C}^n$ where n is the number of leaves in the shape denoted by u^{s}_{w1}. Similarly, ψ_2 expresses a fact in a product structure $\mathcal{C}^m$ where m is the number of leaves in the shape denoted by u^{s}_{w2}. We can therefore use the technique of Feferman-Vaught technique (Section 3.3) to eliminate the quantifiers from formulas ψ_1 and ψ_2. According to Example 17, ψ_1 is equivalent to:

$$
\begin{aligned}
& \exists^{\mathsf{L}} u^{\mathsf{L}}_{0}, u^{\mathsf{L}}_{4}. \\
& \quad u^{\mathsf{L}}_{0} =^{\mathsf{L}} [\exists^{\mathsf{I}} t.\; t \preceq u_{z1} \;\wedge^{\mathsf{I}}\; u_{z1} \preceq t \;\wedge^{\mathsf{I}}\; u_{w1} \preceq t]_{u^{\mathsf{s}}_{w1}} \;\wedge \\
& \quad u^{\mathsf{L}}_{4} =^{\mathsf{L}} [\exists^{\mathsf{I}} t.\; t \preceq u_{z1} \;\wedge^{\mathsf{I}}\; u_{z1} \preceq t \;\wedge^{\mathsf{I}}\; \neg^{\mathsf{I}} u_{w1} \preceq t]_{u^{\mathsf{s}}_{w1}} \;\wedge \\
& \quad |u^{\mathsf{L}}_{4}|_{u^{\mathsf{s}}_{w1}} \geq 1 \;\wedge\; |\neg^{\mathsf{I}} u^{\mathsf{L}}_{0} \;\wedge^{\mathsf{I}}\; \neg^{\mathsf{I}} u^{\mathsf{L}}_{4}|_{u^{\mathsf{s}}_{w1}} = 0
\end{aligned}
$$

We similarly apply Feferman-Vaught construction to ψ_2 and obtain the result true. We may now substitute the results of quantifier elimination in ψ_1 and ψ_2. The resulting formula is:

$$
\begin{aligned}
& \exists u_{vz}, u_{zv}, u_v, u_z, u_w, u_{v1}, u_{v2}, u_{z1}, u_{z2}, u_{w1}, u_{w2}. \\
& \exists^{\mathsf{L}} u^{\mathsf{L}}_{vz}, u^{\mathsf{L}}_{v}, u^{\mathsf{L}}_{z}, u^{\mathsf{L}}_{v1}, u^{\mathsf{L}}_{v2}, u^{\mathsf{L}}_{z1}, u^{\mathsf{L}}_{z2}, u^{\mathsf{L}}_{w1}. \\
& \exists^{\mathsf{s}} u^{\mathsf{s}}_{vz}, u^{\mathsf{s}}_{w}, u^{\mathsf{s}}_{w1}, u^{\mathsf{s}}_{w2}. \\
& \quad \mathsf{shapeBase}_1 \;\wedge \\
& \quad \mathsf{leafsetHom}_2 \;\wedge \\
& \quad \mathsf{termBase}_2 \;\wedge\; \mathsf{termHom}_2 \;\wedge \\
& \quad \mathsf{cardin}_1 \;\wedge\; \mathsf{innerMap}_1
\end{aligned}
$$

where

$$
\mathsf{leafsetHom}_2 = \; \mathsf{lssh}(u^{\mathsf{L}}_{0}) = u^{\mathsf{s}}_{w1} \;\wedge\; \mathsf{lssh}(u^{\mathsf{L}}_{4}) = u^{\mathsf{s}}_{w1}
$$

$$
\begin{aligned}
\mathsf{termBase}_2 = \;& u_z = g(u_{z1}, u_{z2}) \;\wedge\; u_w = g(u_{w1}, u_{w2}) \wedge \\
& z = u_z \wedge w = u_w
\end{aligned}
$$

$$
\begin{aligned}
& \mathsf{innerMap}_2 = \\
& \quad u^{\mathsf{L}}_{0} =^{\mathsf{L}} [\exists^{\mathsf{I}} t.\; t \preceq u_{z1} \;\wedge^{\mathsf{I}}\; u_{z1} \preceq t \;\wedge^{\mathsf{I}}\; u_{w1} \preceq t]_{u^{\mathsf{s}}_{w1}} \;\wedge \\
& \quad u^{\mathsf{L}}_{4} =^{\mathsf{L}} [\exists^{\mathsf{I}} t.\; t \preceq u_{z1} \;\wedge^{\mathsf{I}}\; u_{z1} \preceq t \;\wedge^{\mathsf{I}}\; \neg^{\mathsf{I}} u_{w1} \preceq t]_{u^{\mathsf{s}}_{w1}} \;\wedge \\
& \quad \mathsf{cardin}_2 = \; |u^{\mathsf{L}}_{4}|_{u^{\mathsf{s}}_{w1}} \geq 1 \;\wedge\; |\neg^{\mathsf{I}} u^{\mathsf{L}}_{0} \;\wedge^{\mathsf{I}}\; \neg^{\mathsf{I}} u^{\mathsf{L}}_{4}|_{u^{\mathsf{s}}_{w1}} = 0
\end{aligned}
$$

In the resulting formula all variables are expressible in terms of free variables, so we can write the formula without quantifiers $\exists, \forall, \exists^{\mathsf{L}}, \forall^{\mathsf{L}}$.

♦

The following Proposition 80 is analogous to Proposition 44; the proof is straightforward.

Proposition 80 (Quantification of Struct. Base) *If β is a structural base formula and x a free shape, leafset, or term variable in β, then there exists a base structural formula β_1 equivalent to $\exists x.\beta$.*

The following Proposition 81 corresponds to Proposition 45.

Proposition 81 (Quantifier-Free to Structural Base) *Let ϕ be a well-defined simple formula without quantifiers $\exists^{\mathsf{L}}, \forall^{\mathsf{L}}$, $\exists, \forall$, $\exists^{\mathsf{s}}, \forall^{\mathsf{s}}$. Then ϕ can be written as* true, false, *or a disjunction of structural base formulas.*

Proof Sketch. The overall idea of the transformation to base formula is similar to the transformation in the proof of Proposition 45. Additional complexity is due to inner formulas. However, note that an inner formula $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ is well-defined iff $\delta(u^{\mathsf{s}}, u_1, \ldots, u_n)$ holds where

$$\delta(u^{\mathsf{s}}, u_1, \ldots, u_n) \;\equiv\; \mathsf{sh}(u_1) = u^{\mathsf{s}} \;\wedge \ldots \wedge\; \mathsf{sh}(u_n) = u^{\mathsf{s}}$$

Hence, each formula $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ can be treated as a partial operation p of sort

$$\mathsf{shape} \times \mathsf{term}^n \to \mathsf{leafset}$$

and the domain given by

$$D_p = \langle\langle u^{\mathsf{s}}, u_1, \ldots, u_n\rangle, \delta(u^{\mathsf{s}}, u_1, \ldots, u_n)\rangle$$

This means that we may apply Proposition 9 and convert formula to disjunction existentially quantified well-defined conjunctions of literals in one of the following forms:

1. equality with inner formulas: $u_0^{\mathsf{L}} =^{\mathsf{L}} \phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ where $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ is a u^{s}-inner formula;

2. formulas of leafset boolean algebra:

$$u_0^{\mathsf{L}} =^{\mathsf{L}} u_1^{\mathsf{L}} \wedge^{\mathsf{l}}_{u^{\mathsf{s}}} u_2^{\mathsf{L}}$$
$$u_0^{\mathsf{L}} =^{\mathsf{L}} u_1^{\mathsf{L}} \vee^{\mathsf{l}}_{u^{\mathsf{s}}} u_2^{\mathsf{L}}$$
$$u_0^{\mathsf{L}} =^{\mathsf{L}} \neg^{\mathsf{l}}_{u^{\mathsf{s}}} u_1^{\mathsf{L}}$$
$$u_0^{\mathsf{L}} =^{\mathsf{L}} \mathsf{true}^{\mathsf{l}}_{u^{\mathsf{s}}}$$
$$u_0^{\mathsf{L}} =^{\mathsf{L}} \mathsf{false}^{\mathsf{l}}_{u^{\mathsf{s}}}$$

3. formulas of term algebra of terms:

$$u_1 = u_2, \; u_1 \neq u_2$$
$$u_0 = f(u_1, \ldots, u_n)$$
$$u = f_i(u_0)$$
$$\mathsf{Is}_f(u_0), \; \neg\mathsf{Is}_f(u_0)$$
$$\mathsf{sh}(u) = u^{\mathsf{s}}$$

4. formulas of term algebra of leafsets:

$$u_1^{\mathsf{L}} =^{\mathsf{L}} u_2^{\mathsf{L}}, \; u_1^{\mathsf{L}} \neq^{\mathsf{L}} u_2^{\mathsf{L}}$$
$$u_0^{\mathsf{L}} =^{\mathsf{L}} f^{\mathsf{L}}(u_1^{\mathsf{L}}, \ldots, u_n^{\mathsf{L}})$$
$$u^{\mathsf{L}} =^{\mathsf{L}} f_i^{\mathsf{L}}(u_0^{\mathsf{L}})$$
$$\mathsf{Is}_{f^{\mathsf{L}}}(u_0^{\mathsf{L}}), \; \neg\mathsf{Is}_{f^{\mathsf{L}}}(u_0^{\mathsf{L}})$$
$$\mathsf{lssh}(u^{\mathsf{L}}) =^{\mathsf{L}} u^{\mathsf{s}}$$

5. formulas of term algebra of shapes:

$$u_1^{\mathsf{s}} =^{\mathsf{s}} u_2^{\mathsf{s}}, \; u_1^{\mathsf{s}} \neq^{\mathsf{s}} u_2^{\mathsf{s}}$$
$$u_0^{\mathsf{s}} =^{\mathsf{s}} f^{\mathsf{s}}(u_1^{\mathsf{s}}, \ldots, u_n^{\mathsf{s}})$$
$$u^{\mathsf{s}} =^{\mathsf{s}} f_i^{\mathsf{s}}(u_0^{\mathsf{s}})$$
$$\mathsf{Is}_{f^{\mathsf{s}}}(u_0^{\mathsf{s}}), \; \neg\mathsf{Is}_{f^{\mathsf{s}}}(u_0^{\mathsf{s}})$$

We next describe transformation of each existentially quantified conjunction. In the sequel, whenever we perform case analysis and generate a disjunction of conjunctions, existential quantifiers propagate to the conjunctions, so we keep working with a existentially quantified conjunction. The existentially quantified variables will become internal variables of a structural base formula.

Analogously to the proof of Proposition 28, we use (90), (91), (16) to eliminate literals $\neg\mathsf{Is}_f(u_0)$, $\neg\mathsf{Is}_{f^{\mathsf{L}}} f^{\mathsf{L}}(u_0^{\mathsf{L}})$, $\neg\mathsf{Is}_{g^{\mathsf{s}}}(u_0^{\mathsf{s}})$.

As in the proof of Proposition 45, we replace formulas of leafset boolean algebra by cardinality constraints, similarly to Figure 7.

We next convert formulas of term algebra of terms into a base formula, formulas of term algebra of leafsets into a base formula, and formulas of term algebra of shapes into a base formula.

We simultaneously make sure that every term or leafset variable has an associated associated shape variable, introducing new shape variables if needed.

We also ensure homomorphism requirements by replacing internal variables when we entail their equality.

Another condition we ensure is that parameter term variables map to parameter shape variables, and non-parameter term variables to non-parameter shape variables; we do this by performing expansion of term and shape variables.

We perform expansion of shape variables as in Section 3.2. Expansion of term and variables is even simpler because there is no need to do case analysis on equality of term variable with other variables.

We eliminate disequality between term variables using (92). We eliminate disequalities between leafset variables as in Example 43, by converting each disequality into a cardinality constraint. Elimination of disequalities might violate previously established homomorphism invariants, so we may need to reestablish these invariants by repeating the previously described steps. The overall process terminates because we never introduce new inequalities between term or leafset variables.

As a final step, we convert all cardinality constraints into constraints on parameter term variables, using (95).

In the case when the shape of cardinality constraint is c^{s}, we cannot apply (95). However, in this case, unlike Proposition 45, we do not do case analysis on all possible constant leafsets (this is not even possible in general). This is because Definition 78, unlike Definition 41 implies no need to further decompose cardinality constraints in that case, because we allow primitive non-parameter leafset variables.

This completes our sketch of transforming a quantifier-free formula into disjunction of structural base formulas. ∎

We introduce the notion of determined variables in structural base formula generalizing Definition 29 and Definition 46.

For brevity, we write u^* for internal shape, term, or leafset variables, similarly x^* for a free variable, t^* for a term and f^* for a shape, term, or leafset term algebra constructor and f_i^* for a shape, term, or leafset term algebra selector.

Definition 82 *The set* determinations *of variable determinations of a structural base formula β is the least set S of pairs $\langle u^*, t^* \rangle$ where u^* is an internal term, leafset, or shape variable and t^* is a term over the free variables of β, such such that:*

1. *if $x^* = u^*$ occurs in* termBase*,* leafsetBase*, or* shapeBase*, then $\langle u^*, x^* \rangle \in S$;*
2. *if $\langle u^*, t^* \rangle \in S$ and $u^* = f^*(u_1^*, \ldots, u_k^*)$ occurs in* shapeBase*,* termBase*, or* leafsetBase *then $\{\langle u_1^*, f_1^*(t^*) \rangle, \ldots, \langle u_k^*, f_k^*(t^*) \rangle\} \subseteq S$;*
3. *if $\{\langle u_1^*, f_1^*(t^*) \rangle, \ldots, \langle u_k^*, f_k^*(t^*) \rangle\} \subseteq S$ and $u^* = f^*(u_1^*, \ldots, u_k^*)$ occurs in* shapeBase*,* termBase*, or* leafsetBase *then $\langle u^*, t^* \rangle \in S$;*
4. *if $\langle u, t \rangle \in S$ and $\mathsf{sh}(u) = u^{\mathsf{s}}$ occurs in* termHom *then $\langle u^{\mathsf{s}}, \mathsf{sh}(t) \rangle \in S$;*
5. *if $\langle u^{\mathsf{L}}, t^{\mathsf{L}} \rangle \in S$ and $\mathsf{lssh}(u^{\mathsf{L}}) = u^{\mathsf{s}}$ occurs in* leafsetHom *then $\langle u^{\mathsf{s}}, \mathsf{lssh}(t^{\mathsf{L}}) \rangle \in S$;*
6. *if $u^{\mathsf{L}} = \phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ occurs in* innerMap *where $\phi(u^{\mathsf{s}}, u_1, \ldots, u_n)$ is an inner formula and $\{\langle u^{\mathsf{s}}, t^{\mathsf{s}} \rangle, \langle u_1, t_1 \rangle, \ldots, \langle u_n, t_n \rangle\} \subseteq S$, then $\langle u^{\mathsf{L}}, \phi(t^{\mathsf{s}}, t_1, \ldots, t_n) \rangle \in S$. (In the special case when ϕ contains no free term variables, if $\langle u^{\mathsf{s}}, t^{\mathsf{s}} \rangle \in S$ then $\langle u^{\mathsf{L}}, \phi(u^{\mathsf{s}}) \rangle \in S$.*

Definition 83 *An internal variable u^* is* determined *if $\langle u^*, t^* \rangle \in$* determinations *for some term t^{s}. An internal variable is* undetermined *if it is not determined.*

Lemma 84 *Let β be a structural base formula with matrix β_0 and let* determinations *be the determinations of β. If $\langle u^*, t^* \rangle \in S$ then $\models \beta_0 \Rightarrow u^* = t^*$.*

Proof. By induction, using Definition 82. ■

Corollary 85 *Let β be a structural base formula such that every internal variable is determined. Then β is equivalent to a well-defined formula without quantifiers $\exists^{\mathsf{L}}, \forall^{\mathsf{L}}, \exists, \forall, \exists^{\mathsf{s}}, \forall^{\mathsf{s}}$.*

Proof. By Lemma 84 using (7). ■

Lemma 86 *Let u be an undetermined composed non-parameter term variable in a structural base formula β such that u is a source i.e. no conjunct of form*

$$u' = f(u_1, \ldots, u, \ldots, u_k)$$

occurs in termBase*. Let β' be the result of dropping u from β. Then β is equivalent to β'.*

Proof. Because u is a composed non-parameter term variable, it does not occur in innerMap, so it only occurs in termBase and termHom. The conjunct containing u in termHom is a consequence of the remaining conjuncts, so it may be dropped. After that, applying (7) yields a structural base formula β' not containing u, where β' is equivalent to β. ■

Lemma 87 *Let u^{L} be an undetermined composed non-parameter leafset variable in a structural base formula β such that u^{L} is a source i.e. no conjunct of form*

$$u^{\mathsf{L}\prime} = f^{\mathsf{L}}(u_1^{\mathsf{L}}, \ldots, u^{\mathsf{L}}, \ldots, u_k^{\mathsf{L}})$$

occurs in leafsetBase*. Let β' be the result of dropping u^{L} from β. Then β is equivalent to β'.*

Proof. Because u^{L} is a composed non-parameter term variable, it does not occur in innerMap or cardin, so it only occurs in leafsetBase and leafsetHom. The conjunct containing u^{L} in leafsetHom is a consequence of the remaining conjuncts, so it may be dropped. After that, applying (7) yields a structural base formula β' not containing u^{L}, where β' is equivalent to β. ■

Corollary 88 *Every base formula is equivalent to a base formula without undetermined composed non-parameter term variables and without undetermined composed non-parameter leafset variables.*

Proof. If a structural base formula has an undetermined composed non-parameter term variable, then it has an undetermined composed non-parameter term variable that is a source, similarly for leafset variables. By repeated application of Lemma 86 and Lemma 87 we eliminate all undetermined non-parameter term and leafset variables. ■

The following Proposition 89 corresponds to Proposition 53 and Proposition 66.

Proposition 89 (Struct. Base to Quantifier-Free) *Every structural base formula β is equivalent to a well-defined simple formula ϕ without quantifiers $\exists^{\mathsf{L}}, \forall^{\mathsf{L}}, \exists, \forall, \exists^{\mathsf{s}}, \forall^{\mathsf{s}}$.*

Proof Sketch. By Corollary 88 we may assume that β has no undetermined composed non-parameter term and leafset variables. By Corollary 85 we are done if there are no undetermined variables, so it suffices to eliminate:

1. undetermined parameter term variables,
2. undetermined primitive non-parameter term variables,
3. undetermined parameter leafset variables,
4. undetermined primitive non-parameter leafset variables, and
5. undetermined shape variables.

If u is an undetermined parameter term variable or a primitive non-parameter term variable, then u does not occur in termBase, so it occurs only in termHom and innerMap. If u^{L} is an undetermined parameter leafset variable or a primitive non-parameter leafset variable then u^{L} does not occur in leafsetBase, so it occurs only in leafsetHom, innerMap, and cardin.

For a undetermined term or leafset variable of shape u^{s} such that there is an uncovered parameter or primitive non-parameter term or leafset variable with shape u^{s}, consider all conjuncts γ_i in innerMap of form

$$u_j^{\mathsf{L}} = \phi(u^{\mathsf{s}}, u_{i_1}, \ldots, u_{i_k})$$

and all conjuncts δ_i from cardin of form:

$$|t^{\mathsf{L}}(u^{\mathsf{L}}_{r^{\mathsf{L}}+1}, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}})|_{u^{\mathsf{s}}} = k$$

or

$$|t^{\mathsf{L}}(u^{\mathsf{L}}_{r^{\mathsf{L}}+1}, \ldots, u^{\mathsf{L}}_{n^{\mathsf{L}}})|_{u^{\mathsf{s}}} \geq k$$

Together with formulas from $\mathsf{termHom}$ and $\mathsf{leafsetHom}$ that contain term and leafset variables free in formulas γ_i and δ_i, these conjuncts form a formula η which expresses a relation in the substructure of term-power algebra which (because constructors are covariant) is isomorphic to a term-power of $\mathcal{C}$. We therefore use Feferman-Vaught theorem from Section 3.3 to eliminate all term and parameter variables from η. By repeating this process we eliminate all undetermined parameter and leafset variables.

It remains to eliminate undetermined shape variables. This process is similar to term algebra quantifier elimination in Section 3.4. An essential part of construction in Section 3.4 is Lemma 25, which relies on the fact that undetermined parameter variables may take on infinitely many values. We therefore ensure that undetermined parameter shape variables are not constrained by term and parameter variables through conjuncts outside $\mathsf{shapeBase}$. An undetermined parameter shape variable u^{s} does not occur in $\mathsf{termHom}$ or $\mathsf{leafsetHom}$ because there are no parameter term and leafset variables, so u^{s} can occur only in $\mathsf{innerMap}$ and cardin.

However, because undetermined parameter and leafset variables are eliminated from the formula, if u^{s} is a parameter shape variable then exactly one of these two cases holds:

1. there are some conjuncts in $\mathsf{innerMap}$ and cardin that contain u^{s} and contain some determined term and leafset variables, in this case u^{s} is determined, or
2. there are no conjuncts in $\mathsf{innerMap}$ containing u^{s} and cardin contains only domain cardinality constraints of form $|1|_{u^{\mathsf{s}}} = k$ and $|1|_{u^{\mathsf{s}}} \geq k$.

Hence, if u^{s} is a shape variable it remains to eliminate the constraints of form $|1|_{u^{\mathsf{s}}} = k$ and $|1|_{u^{\mathsf{s}}} \geq k$. We eliminate these constraints as in the proof of Proposition 66.

In the resulting formula all variables are determined. By Corollary 85 the formula can be written as a formula without quantifiers $\exists^{\mathsf{L}}, \forall^{\mathsf{L}}$, $\exists, \forall$, $\exists^{\mathsf{s}}, \forall^{\mathsf{s}}$. ∎

The following is the main result of this paper.

Theorem 90 (Term Power Quant. Elimination) *There exist algorithms A, B such that for a given formula ϕ in the language of Figure 9:*

a) A produces a quantifier-free formula ϕ' in selector language

b) B produces a disjunction ϕ' of structural base formulas

We also explicitly state the following corollary.

Corollary 91 *Let $\mathcal{C}$ be a structure with decidable first-order theory. Then the set of true sentences in the logic of Figure 9 interpreted in the structure $\mathcal{P}$ according to Figures 10 and 11 is decidable.*

6.5 Handling Contravariant Constructors

In this section we discuss the decidability of the Σ-term-power structure for a decidable theory $\mathcal{C}$ when some of the function symbols $f \in \Sigma$ are contravariant. We then suggest a generalization of the notion of variance to multiple relations and to relations with arity greater than two.

The modifications needed to accommodate contravariance with respect to some distinguished relation symbol $\leq \in R$ for the case of infinite C are analogous to the modifications in Section 5.5. We this obtain a quantifier elimination procedure for any decidable theory $\mathcal{C}$ in the presence of contravariant constructors.

Theorem 92 (Decidability of Structural Subtyping) *Let $\mathcal{C}$ be a decidable structure and $\mathcal{P}$ a Σ-term-power of $\mathcal{C}$. Then the first-order theory of $\mathcal{P}$ is decidable.*

In the rest of this section we consider a generalization that allows defining variance for every relation symbol $r \in R$ of any arity, and not just the relation symbol $\leq \in R$.

For a given relation symbol $r \in R$, function symbol $f \in \Sigma$, with $k = \mathsf{ar}(f)$, and integer i where $1 \leq i \leq k$, let $P_r(f,i)$ denote a permutation of the set $\{1,\ldots,k\}$ that specifies the variance of the i-th argument of f with respect to the relation r. For example, if r is a binary relation then $P_r(f,i)$ is the identity permutation $\{\langle 1,1\rangle\langle 2,2\rangle\}$ if i-th argument of f is covariant, or a the transpose permutation $\{\langle 1,2\rangle,\langle 2,1\rangle\}$ if i-th argument of f is contravariant.

If $l \in \mathsf{leaves}(s)$ is a leaf $l = \langle f^1, i^1\rangle \ldots \langle f^n, i^n\rangle$, define the permutation $\mathsf{variance}(l)$ as the composition of permutations:

$$\mathsf{variance}(l) = P_r(f^n, i^n) \circ \cdots \circ P_r(f^1, i^1)$$

Then define $[\![r]\!]$ by

$$\begin{aligned}&[\![r]\!](s, t_1, \ldots, t_k) = \\ &\langle s, \{l \mid \; [\![r]\!]^C(t_{p_1}[l], \ldots, t_{p_k}[l]) \wedge \\ &\qquad \langle p_1, \ldots, p_k\rangle = \mathsf{variance}(l) \\ &\quad \}\rangle\end{aligned}$$

We generalize (76) by defining

$$N_\pi(s) = |\{l \in \mathsf{leaves}(s) \mid \mathsf{variance}(l) = \pi\}|$$

As in Section 5.5, we can transform the constraints $|1|_{u^{\mathsf{s}}} = k$ and $|1|_{u^{\mathsf{s}}} \geq k$ on each parameter shape variable into a conjunction of constraints of form:

$$N_\pi(u^{\mathsf{s}}) = k$$

or

$$N_\pi(u^{\mathsf{s}}) \geq k$$

A problem on nonnegative integers. To solve the problem of variance with any number of relation symbols of any arity, it suffices to solve the following problem on sets of tuples of non-negative integers.

Let $\mathsf{Nat} = \{0,1,2,\ldots\}$. Consider the structure $\mathsf{St} = \mathsf{Nat}^d$ for some $d \geq 2$ and let $D = \{1,2,\ldots,d\}$. If p is a permutation on D, let M_p denote an operation $\mathsf{St} \to \mathsf{St}$ defined by

$$M_p(x_1, \ldots, x_d) = (x_{p_1}, \ldots, x_{p_d})$$

If $\langle x_1, \ldots, x_d \rangle, \langle y_1, \ldots, y_d \rangle \in \mathsf{St}$ define

$$\langle x_1, \ldots, x_d \rangle + \langle y_1, \ldots, y_d \rangle = \langle x_1 + y_1, \ldots, x_d + y_d \rangle$$

Consider a finite set of operations $f : \mathsf{St}^k \to \mathsf{St}$ where each operation f is determined by k permutations $p_1^f, \ldots, p_k^f$ in the following way:

$$f(t_1, \ldots, t_k) = M_{p_1^f}(t_1) + \ldots + M_{p_k^f}(t_k)$$

Hence, each operation f of arity k is given by a permutation which specifies how to exchange the order of arguments in the tuple. After permuting the arguments the tuples are summed up.

Given a finite set F of operations f, let S be the set generated by operations in F starting from the element $(1, 0, \ldots, 0) \in \mathsf{St}$. Let $C(n_1, \ldots, n_d)$ be a conjunction of simple linear constraints of the forms

$$n_i = a_i$$

and

$$n_i \geq a_i$$

Consider the set

$$A_C = \{(n_1, \ldots, n_d) \in S | C(n_1, \ldots, n_d)\}$$

The problem is: For given set of operations F, is there an algorithm that given $C(n_1, \ldots, n_d)$ finitely computes the set A_C.

End of a problem on nonnegative integers.

We conjecture that the technique of Lemma 68 can be generalized to yield a solution to the problem on nonnegative integers and thus establish the decidability for the notion of variance with respect to any number of relations with any number of arguments.

6.6 A Note on Element Selection

We make a brief note related to the choice of the language for making statements in term-power algebras. In Section 5 we avoided the use of leafset variables by substituting them into cardinality constraints. In this section we use a cylindric algebra of leafsets.

An apparently even more flexible alternative is to allow the element selection operation

$$\mathsf{select} :: \mathsf{term} \times \mathsf{leaf} \to \mathsf{elem}$$

where elem is a new sort, interpreted over the set C, and leaf is a sort interpreted over the set of pairs of a shape and a leaf. Instead of the formula

$$r_{u^s}(t_1, \ldots, t_n) =^{\mathsf{L}} \mathsf{true}^{\mathsf{l}}_{u^s}$$

we would then write

$$\forall l.\ r_{u^s}(\mathsf{select}(t_1, l), \ldots, \mathsf{select}(t_n, l)) =^{\mathsf{L}} \mathsf{true}^{\mathsf{l}}_{u^s}$$

Using select operation we can define update relation:

$$\begin{aligned} &\mathsf{update}(t_1, l_0, e, t_2) \;\equiv \\ &\forall l.\ ((l = l_0 \ \wedge\ \mathsf{select}(t_2, l) = e) \vee \\ &\qquad (l \neq l_0 \ \wedge\ \mathsf{select}(t_2, l) = \mathsf{select}(t_1, l))) \end{aligned}$$

The resulting language is at least as expressive as the language in Figure 5. This language is interesting because it allows reasoning about updates to leaves of a tree of fixed shape, thus generalizing the theory of updatable arrays [33] to the theory of trees with update operations, which would be useful for program verification. We did not choose this more expressive language in this report for the following reason.

If the base structure $\mathcal{C}$ has a finite domain C, then for certain reasonable choice of the relations interpreting L_C it is possible to express statements of this extended language in the logic of Figure 9. The idea is to assume a partial order on the elements of C with a minimal element, and use terms t with exactly one leaf non-minimal to model the leaves.

On the other hand, in the more interesting case when C is infinite, we can easily obtain undecidable theories in the presence of selection operation. Namely, the selection operation allows terms to be used as finite sets of elements of C. The term-power therefore increases the expressiveness from the first-order theory to the weak monadic second-order theory, which allows quantification over finite sets of objects. Weak monadic theory allows in particular inductive definitions. If theory of structure C is decidable, weak monadic theory might therefore still be undecidable, as an example we might take the term algebra itself, whose weak monadic theory would allow defining subterm relation, yielding an undecidable theory [56, Page 508].

7 Some Connections with MSOL

This section explores some relationships between the theory of structural subtyping and monadic second-order logic (MSOL) interpreted over tree-like structures. We present it as a series of remarks that are potentially useful for understanding the first-order theory of structural subtyping of recursive types, see [36, 37] for similar results in the context of the theory of feature trees.

In Section 7.1 we exhibit an embedding of MSOL of *infinite* binary tree into the first-order theory of structural subtyping of *recursive types* with two constant symbols a,b and one covariant binary function symbol f. MSOL of infinite binary tree is decidable. Although the embedding does not give an answer to the decidability of the structural subtyping of recursive types, it does show that the problem is at least as difficult as decidability of MSOL over infinite trees. We therefore expect that, if the theory of structural subtyping of recursive types is decidable, the decidability proof will likely either use decidability of MSOL over infinite trees, or use directly techniques similar to those of [18, 57].

In Section 7.2 we use the embedding in Section 7.1 to argue the decidability of formulas of the first-order theory of structural subtyping of recursive types where variables range over terms of certain fixed infinite shape s_e.

In Section 7.3 we present an encoding of all terms using terms of shape s_e. We argue that the main obstacle in using this encoding to show the decidability of the first-order theory of structural subtyping recursive types is inability to define the set of all prefix-closed terms of the shape s_e.

In Section 7.4 we generalize the decidability result of Section 7.2 by allowing different variables to range over different constant shapes.

In Section 7.5 we illustrate some of the difficulties in reducing first-order theory of structural subtyping to MSOL over tree-like structures. We show that if we use a certain form of infinite feature trees instead of infinite terms, the decidability follows.

In Section 7.6 we point out that monadic second-order logic with prefix-closed sets is undecidable, which follows from [48]. This fact indicates that if we hope to show the decidability of structural subtyping of recursive types, it is essential to maintain the incomparability of types of different shape.

7.1 Structural Subtyping Recursive Types

In this section we define the problem of structural subtyping of *recursive* types. We then give an embedding of MSOL of the infinite binary tree into the first-order theory of structural subtyping of infinite terms over the signature $\Sigma = \{a, b, g\}$ with the partial order $\leq$.

We define MSOL over infinite binary tree [6, Page 317] as the structure $\mathsf{MSOL}^{(2)} = \langle \{0,1\}^*, \mathsf{succ}_0, \mathsf{succ}_1 \rangle$. The domain of the structure is the set $\{0,1\}^*$ of all finite strings over the alphabet $\{0,1\}$. We denote first-order variables by lowercase letters such as x, y, z. First-order variables range over finite words $w \in \{0,1\}^*$. We denote second-order variables by uppercase letters such as X, Y, Z. Second-order variables range over finite and infinite subsets $S \subseteq \{0,1\}^*$. The only relational symbol is equality, with the standard interpretation. There are two function symbols, denoting the appending of the symbol 0 and the appending of the symbol 1 to a word:

$$\begin{aligned} \mathsf{succ}_0\, w &= w \cdot 0 \\ \mathsf{succ}_1\, w &= w \cdot 1 \end{aligned}$$

For the purpose of embedding into the first-order theory of structural subtyping, we consider a structure $\mathsf{MSOL}^{(1)} = \langle \{0,1\}^*, \subseteq, \mathsf{Succ}_0, \mathsf{Succ}_1 \rangle$ equivalent to $\mathsf{MSOL}^{(2)}$. We use the language of MSOL without first-order variables to make statements within $\mathsf{MSOL}^{(1)}$. $\subseteq$ is a binary relation on sets denoting the subset relation:

$$Y_1 \subseteq Y_2 \iff \forall x.\ x \in Y_1 \Rightarrow x \in Y_2$$

Succ_0 and Succ_1 are binary relations on sets, $\mathsf{Succ}_0, \mathsf{Succ}_1 \subseteq 2^{\{0,1\}^*} \times 2^{\{0,1\}^*}$, defined as follows:

$$\begin{aligned} \mathsf{Succ}_0(Y_1, Y_2) &\iff Y_2 = \{w \cdot 0 \mid w \in Y_1\} \\ \mathsf{Succ}_1(Y_1, Y_2) &\iff Y_2 = \{w \cdot 1 \mid w \in Y_1\} \end{aligned}$$

The structure $\mathsf{MSOL}^{(1)}$ is similar to one in [18]; the difference is that relations Succ_0 and Succ_1 are true even for non-singleton sets.

Lemmas 93 and 94 show the expected equivalence of $\mathsf{MSOL}^{(2)}$ and $\mathsf{MSOL}^{(1)}$.

Lemma 93 ($\mathsf{MSOL}^{(2)}$ **expresses** $\mathsf{MSOL}^{(1)}$) *Every relation on sets definable in* $\mathsf{MSOL}^{(1)}$ *is definable in* $\mathsf{MSOL}^{(2)}$.

Proof. We express relations $\subseteq$, Succ_0, Succ_1 as formulas in $\mathsf{MSOL}^{(2)}$, as follows. We express $Y_1 \subseteq Y_2$ as

$$\forall x.\ Y_1(x) \Rightarrow Y_2(x),$$

$\mathsf{Succ}_0(Y_1, Y_2)$ as

$$\forall x. Y_2(x) \iff \exists y. y = \mathsf{succ}_0(x),$$

and $\mathsf{Succ}_1(Y_1, Y_2)$ as

$$\forall x. Y_2(x) \iff \exists y. y = \mathsf{succ}_1(x).$$

The statement follows by induction on the structure of formulas. ∎

Let $R \subseteq (2^{\{0,1\}^*})^k \times (\{0,1\}^*)^n$ be relation of arity $k+n$. Define $R^* \subseteq (2^{\{0,1\}^*})^k \times (2^{\{0,1\}^*})^n$ by

$$\begin{aligned} & R^*(Y_1, \ldots, Y_k, X_1, \ldots, X_n) \ \equiv \\ & \qquad \exists x_1, \ldots, x_n. \quad X_1 = \{x_1\} \wedge \cdots \wedge X_n = \{x_n\} \wedge \\ & \qquad\qquad R(Y_1, \ldots, Y_k, x_1, \ldots, x_n) \end{aligned}$$

Lemma 94 ($\mathsf{MSOL}^{(1)}$ **expresses** $\mathsf{MSOL}^{(2)}$) *If R is definable in* $\mathsf{MSOL}^{(2)}$*, then R^* is definable in* $\mathsf{MSOL}^{(1)}$.

Proof Sketch. Property of being an empty set is definable in $\mathsf{MSOL}^{(1)}$ by the formula

$$\phi_0(Y_1) \ \equiv\ \forall Y_2. Y_1 \subseteq Y_2$$

The relation $\subset$ of being a proper subset is definable in $\mathsf{MSOL}^{(1)}$ by formula

$$\phi_1(Y_1, Y_2) \ \equiv\ Y_1 \subseteq Y_2 \wedge Y_1 \neq Y_2$$

and the relation $\subset_1$ of having one element more is definable by formula

$$\phi_2(Y_1, Y_2) \ \equiv\ Y_1 \subset Y_2 \wedge \neg \exists Z.\ Y_1 \subset Z \wedge Z \subset Y_2$$

The property of being a singleton set can then be expressed by formula

$$\phi_3(Y_1) \ \equiv\ \exists Y_0.\ \phi_0(Y_0) \wedge Y_0 \subset_1 Y_1$$

We define the relation on singletons corresponding to succ_0 by

$$\phi_4(Y_1, Y_2) \ \equiv\ \phi_3(Y_1) \wedge \phi_3(Y_2) \wedge \mathsf{Succ}_0(Y_1, Y_2)$$

Similarly, the relation corresponding to succ_1 is defined by

$$\phi_5(Y_1, Y_2) \ \equiv\ \phi_3(Y_1) \wedge \phi_3(Y_2) \wedge \mathsf{Succ}_1(Y_1, Y_2)$$

If R is expressible by some formula ψ in $\mathsf{MSOL}^{(2)}$, then R is expressible by a formula in prenex normal form, so suppose ψ is of form

$$Q_1 V_1 \ldots Q_n V_n . \psi_0$$

where ψ_0 is quantifier free. We construct a formula ψ' expressing R^* in $\mathsf{MSOL}^{(1)}$. We obtain the matrix ψ_0' of ψ' by translating ψ_0 as follows. If x is a first-order variable in ψ_0, we represent it with a second-order variable X denoting a singleton set. We replace membership relation $Y(x)$ with subset relation $X \subset Y$. We replace succ_0 with ϕ_4 and succ_1 with ϕ_5. We construct ψ' by adding quantifiers to ψ_0' as follows. Second-order quantifiers remain the same. First-order quantifiers are relativized to range over singleton sets: $\forall x. \psi_i$ becomes $\forall X. \phi_3(X) \Rightarrow \psi_i'$ and $\exists x. \psi_i$ becomes $\exists X.\ \phi_3(X) \wedge \psi_i'(X)$. ∎

We can view $\mathsf{MSOL}^{(1)}$ as a first-order structure with the domain $2^{\{0,1\}^*}$. We show how to embed $\mathsf{MSOL}^{(1)}$ into the first-order theory of structural subtyping.

We define the first-order structure of structural subtyping of *recursive* types similarly to the corresponding structure for non-recursive types in Section 4; the only difference is that the domain contains both finite and *infinite* terms. Infinite terms correspond to infinite trees [12, 30].

We define infinite trees as follows. We use alphabet $\{l, r\}$ to denote paths in the tree. A *tree domain* D is a finite or infinite subset of the set $\{l, r\}^*$ such that:

1. D is prefix-closed: if $w \in \{l, r\}^*$, $x \in \{l, r\}$ then $w \cdot x \in D$ implies $w \in D$;

2. if $w \in D$ then exactly one of the following two properties hold:

   (a) w is an interior node: $\{w \cdot l, w \cdot r\} \subseteq D$

   (b) w is a leaf: $\{w \cdot l, w \cdot r\} \cap D = \emptyset$.

A *tree* with a tree domain D is a total function T from the set of leaves of D to the set $\{a, b\}$.

Note that the tree domain D of a tree T can be reconstructed from T as the prefix closure of the domain of the graph of function T; we write $\mathsf{TDom}(T)$ for the tree domain of tree T.

Two trees are equal if they are equal as functions. Hence, equal trees have equal function domains and equal tree domains.

We say that $T_1 \leq T_2$ iff $\mathsf{TDom}(T_1) = \mathsf{TDom}(T_2)$ and $T_1(w) \leq_0 T_2(w)$ for every word $w \in \mathsf{TDom}(T_1)$. Here $\leq_0$ is the relation $\{\langle a, a\rangle, \langle a, b\rangle, \langle b, b\rangle\}$.

If T_1 and T_2 are trees, then $g(T_1, T_2)$ denotes the tree T such that

$$\mathsf{TDom}(T) = \{l \cdot w \mid w \in T_1\} \cup \{r \cdot w \mid w \in T_2\}$$

$$\begin{aligned} T(l \cdot w) &= T_1(w), && \text{if } w \in T_1 \\ T(r \cdot w) &= T_2(w), && \text{if } w \in T_2 \end{aligned}$$

Let IT denote the set of all infinite trees. The structural subtyping structure is the structure $\mathsf{SIT} = \langle \mathsf{IT}, g, a, b, \leq\rangle$. SIT is an infinite-term counterpart to the structure BS from Section 4.

Similarly to the case of finite terms, define the relation $\sim$ of "being of the same shape" in SIT by

$$t_1 \sim t_2 \;\equiv\; \exists t_0.\ t_0 \leq t_1 \wedge t_0 \leq t_2$$

Observe that $t_1 \sim t_2$ iff $\mathsf{TDom}(t_1) = \mathsf{TDom}(t_2)$.

We next present an embedding ι of $\mathsf{MSOL}^{(1)}$ into SIT. The image of the embedding ι are the infinite trees that are in the same $\sim$-equivalence-class with the tree t_e. We define t_e as the unique solution of the equation:

$$t_e = g(g(t_e, t_e), a)$$

Trees in the $\sim$-equivalence class of t_e have the tree domain $D = \mathsf{TDom}(t_e)$ given by the regular context-free grammar

$$\mathsf{D} \to \epsilon \mid r \mid l \mid lr\mathsf{D} \mid ll\mathsf{D}$$

whereas the leaves L of D are given by the context-free grammar

$$\mathsf{L} \to \epsilon \mid r \mid lr\mathsf{L} \mid ll\mathsf{L}$$

or the regular expression $(lr|ll)^*r$. Let h be the homomorphism of words from $\{0,1\}^*$ to $\{l, r\}^*$ such that

$$\begin{aligned} h(0) &= ll \\ h(1) &= lr \end{aligned}$$

If $w = a_1 \ldots a_n$ is a word, then w^R denotes the reverse of the word, $w^R = a_n \ldots a_1$.

We define the embedding ι to map a set $Y \subseteq \{0,1\}^*$ into the unique tree t such that $t \sim t_e$ and for every $w \in \{0,1\}^*$,

$$w \in Y \iff T(h(w^R) \cdot r) = b \tag{103}$$

Observe that $\iota(\emptyset) = t_e$. Define formulas $\mathsf{TSucc}_0(t_1, t_2)$ and $\mathsf{TSucc}_1(t_1, t_2)$ as follows:

$$\begin{aligned} \mathsf{TSucc}_0(t_1, t_2) &\equiv t_2 = g(g(t_1, t_e), t_e) \\ \mathsf{TSucc}_1(t_1, t_2) &\equiv t_2 = g(g(t_e, t_1), t_e) \end{aligned}$$

It is straightforward to show that ι is an injection and that ι maps relation $\subset$ into $\leq$, relation Succ_0 into TSucc_0, and relation Succ_1 into TSucc_1. Moreover, the range of ι is the set of all terms t such that $\mathsf{sh}(t) = s_e$ where $s_e = \mathsf{sh}(t_e)$.

7.2 A Decidable Substructure

Section 7.1 shows that terms of shape s_e form a substructure within SIT that is isomorphic to $\mathsf{MSOL}^{(1)}$. In this section we consider the following converse problem.

Consider the formulas $\mathcal{BF}$ that, instead of quantifiers $\exists, \forall$, contain bounded quantifiers $\exists_e, \forall_e$ that range over the elements of the set

$$T_e = \{t \mid \mathsf{sh}(t) = s_e\}$$

We show that the set of closed formulas from $\mathcal{BF}$ that are true in SIT is decidable.

Although the quantifiers are bounded, terms in this logic can still denote elements of shape other than s_e. For example, the in the atomic formula

$$g(x_1, x_2) \leq g(x_3, g(g(x_4, x_5), b))$$

the term $g(x_1, x_2)$ denotes a term of the shape $g^{\mathsf{s}}(s_e, s_e)$. First we show that all atomic formulas are of one of the following forms:

1. $x_0 = g(g(x_1, x_2), a)$;

2. $x_0 = g(g(x_1, x_2), b)$;

3. $x_1 = x_2$;

4. $x_1 \leq x_2$.

Consider an atomic formula $t_1 = t_2$. The key idea is that if $\mathsf{sh}(t_1) \neq \mathsf{sh}(t_2)$ then the formula $t_1 = t_2$ is false.

If none of the term t_1 and t_2 is a variable then one of them is a constant or a constructor application. If $t_1 \equiv g(t_{11}, t_{12})$ then either $t_1 = t_2$ is false or $t_2 \equiv g(t_{21}, t_{22})$ for some t_{21}, t_{22}. We may therefore decompose $t_1 = t_2$ into $t_{11} = t_{21}$ and $t_{12} = t_{22}$. By repeating this decomposition we arrive at terms of form $t_1 = t_2$ where both t_1 and t_2 are constants or at the equality of form $x_0 = t(x_1, \ldots, x_n)$. The equalities between the constants can be trivially evaluated. This leaves only terms of form $x_0 = t(x_1, \ldots, x_n)$. Let $t^{\mathsf{s}}(x_1^{\mathsf{s}}, \ldots, x_n^{\mathsf{s}})$ be a shape term that results from replacing a and b with c^{s} and replacing g with g^{s} in t. Because all variables range over T_e, we conclude that $x_0 = t(x_1, \ldots, x_n)$ can be true only if

$$s_e = t^{\mathsf{s}}(s_e, \ldots, s_e)$$

If $t(x_1, \ldots, x_n) \in \{a, b\}$ is then (7.2) is false. If $t(x_1, \ldots, x_n) \equiv x_1$, we obtain formula of the desired form. So assume $t(x_1, \ldots, x_n) \equiv g(t_{21}, t_{22})$. Then $\mathsf{sh}(t_{21}) = g^{\mathsf{s}}(s_e, s_e)$ and $\mathsf{sh}(t_{22}) = c^{\mathsf{s}}$. Therefore, $t_{21} \equiv g(t_{211}, t_{212})$ where either $\mathsf{sh}(t_{211}) = \mathsf{sh}(t_{212}) = s_e$ or $t_1 = t_2$ is false. Similarly, either $t_{22} \in \{a, b\}$ or $t_1 = t_2$ is false. Therefore, $t(x_1, \ldots, x_n) \equiv g(g(t_{211}, t_{212}), a)$, $t(x_1, \ldots, x_n) \equiv g(g(t_{211}, t_{212}), b)$, or $t_1 = t_2$ is false. If $t(x_1, \ldots, x_n) \equiv g(g(t_{211}, t_{212}), a)$ then we may replace the $t_1 = t_2$ with the formula

$$\exists_e y_1, y_2.\ x_0 = g(g(y_1, y_2), a) \ \wedge \ y_1 = t_{211} \ \wedge \ y_2 = t_{212}$$

and similarly in the other case. By continuing this process by the induction on the structure of the term $t(x_1, \ldots, x_n)$ we either conclude that $t_1 = t_2$ is false, or we conclude that $t_1 = t_2$ is equivalent to a conjunction of formulas of the desired form.

Conversion of atomic formula of form $t_1 \leq t_2$ is analogous to the conversion of formulas $t_1 = t_2$.

To see the decidability it now suffices to convert the formulas of the form $x_0 = g(g(x_1, x_2), a)$ and $x_0 = g(g(x_1, x_2), b)$ into formulas $\mathsf{TSucc}_0(t_1, t_2)$ and $\mathsf{TSucc}_1(t_1, t_2)$. Expressibility of $x_0 = g(g(x_1, x_2), a)$ follows from the fact that the following relationship between X_0, X_1, X_2 is expressible in MSOL:

$$X_0 = \{w \cdot 0 \mid w \in X_1\} \cup \{w \cdot 1 \mid w \in X_2\}$$

Similarly, the expressibility of $x_0 = g(g(x_1, x_2), b)$ follows from the fact that

$$X_0 = \{w \cdot 0 \mid w \in X_1\} \cup \{w \cdot 1 \mid w \in X_2\} \cup \{\epsilon\}$$

is expressible in MSOL. We conclude that the set of closed $\mathcal{BF}$ formulas that are true in SIT is decidable.

7.3 Embedding Terms into Terms

We next give an embedding of the set of all terms into T_e. As in Section 7.1 t_e be the unique solution of the equation $t_e = g(g(t_e, t_e), a)$ and let

$$t_4(x_1, x_2, x_3, x_4) \equiv g(g(g(g(x_1, x_2), x_3), t_e), x_4)$$

Define

$$\begin{aligned} t_a &\equiv t_4(t_e, t_e, a, a) \\ t_b &\equiv t_4(t_e, t_e, a, b) \\ t_g(x_1, x_2) &\equiv t_4(x_1, x_2, b, b) \end{aligned}$$

Then define the homomorphism h_T from the set of all terms to the set T_e by

$$\begin{aligned} h_T(a) &= t_a \\ h_T(b) &= t_b \\ h_T(g(t_1, t_2)) &= t_g(h_T(t_1), h_T(t_2)) \end{aligned}$$

Then h_T is embedding of the set of all terms into the subset subset T_e of all terms. The term algebra operations a, b, g map to t_a, t_b, t_g and $\leq$ maps to $\leq$.

Note that, if it were possible to define a predicate $P(t)$ such that

$$P(x) \iff \exists y. h_T(y) = x \qquad (104)$$

then we could express all statements of SIT within the $\mathcal{BF}$ subtheory, and therefore SIT would be decidable.

The fundamental problem with specifying $P(x)$ is not the use of two bits to encode the three possible elements $\{a, b, g\}$, but the constraint that if a term contains a subterm of the form $t_4(t_1, t_2, a, a)$ or $t_4(t_1, t_2, a, b)$ at some even depth, then $t_1 \equiv t_2 \equiv t_e$. Compared to the relationships given by constructor g, this constraint requires taking about successor relation at the opposite side of the paths within a tree, see Section 7.6.

7.4 Subtyping Trees of Known Shape

We next argue that if we allow the logic to have a copy of bounded quantifiers $\exists_s, \forall_s$ for every *constant* shape s, we obtain a decidable theory. To denote constant shapes in a finite number of symbols we consider in addition to term algebra symbols $g^{\mathsf{s}}, c^{\mathsf{s}}$ the expressions that yield solutions of mutually recursive equations on shapes; the details of the representation of types are not crucial for our argument, see e.g. [12]

Consider a closed formula in such language. Because every variable has an associated constant shape, we can compute the set of all shapes occurring in the formula. This means that all variables of the formula range over a finite known set of shapes. This allows us to define the predicate P given by (104) as a disjunction of cases, one case for every shape. Define h_{min}, h_{max} functions that take a shape and produce a lower and upper bound for terms of that shape:

$$\begin{aligned} h_{\mathsf{min}}(c^{\mathsf{s}}) &= t_a \\ h_{\mathsf{min}}(g^{\mathsf{s}}(t_1^{\mathsf{s}}, t_2^{\mathsf{s}})) &= t_g(h_{\mathsf{min}}(t_1^{\mathsf{s}}), h_{\mathsf{min}}(t_2^{\mathsf{s}})) \\ h_{\mathsf{max}}(c^{\mathsf{s}}) &= t_b \\ h_{\mathsf{max}}(g^{\mathsf{s}}(t_1^{\mathsf{s}}, t_2^{\mathsf{s}})) &= t_g(h_{\mathsf{max}}(t_1^{\mathsf{s}}), h_{\mathsf{max}}(t_2^{\mathsf{s}})) \end{aligned}$$

If $s_1, \ldots, s_n$ is the list of shapes occurring in a formula, we then define a predicate P specific to that formula by

$$P(t) = \bigvee_{i=1}^{n} (h_{\mathsf{min}}(s_i) \leq t \wedge t \leq h_{\mathsf{max}}(t))$$

We can therefore define $P(t)$ and use it to translate the formula into a $\mathcal{BF}$ formula of the same truth value. Therefore, structural subtyping with quantification bounded to constant shapes is decidable.

For decidability of the structural subtyping recursive types it would be interesting to examine the decision procedure for MSOL and determine whether there is some uniformity in it that would allow us to handle even quantification over shapes that are determined by variables.

7.5 Recursive Feature Trees

We next remark that certain notion of subtyping of recursive feature trees is decidable. By a feature tree we mean an infinite tree built using a constructor which takes other feature trees and an optional node label as an argument. In this section we consider the simple case of one binary constructor f and assume only one label denoted by 1. Hence, an empty feature tree is a feature tree, and if t_1 and t_2 are feature trees then so are $f^{\epsilon}(t_1, t_2)$ and $f^{1}(t_1, t_2)$. We represent an empty feature tree e by an infinite tree that has all features ϵ. We compare feature trees as follows. Let $\leq$ be defined on the features $\{\epsilon, 1\}$ as the relation $\{\langle \epsilon, \epsilon \rangle, \langle \epsilon, 1 \rangle, \langle 1, 1 \rangle\}$. Define $\leq$ on trees as the least relation such that:

1. $e \leq t$ for all terms t;
2. $t_1 \leq t'_1$ and $t_2 \leq t'_2$ implies
$$f^{r_1}(t_1, t_2) \leq f^{r_2}(t'_1, t'_2)$$
for all $r_1, r_2 \in \{\epsilon, 1\}$ such that $r_1 \leq r_2$.

The decidability of feature trees follows from Section 7.1 because of the isomorphism h_F between the set of terms T_e and the set of feature trees. Here h_F is defined by:

$$\begin{aligned} h_F(e) &= t_e \\ h_F(f^{\epsilon}(t_1, t_2)) &= g(h_F(t_1), h_F(t_2), a) \\ h_F(f^{1}(t_1, t_2)) &= g(h_F(t_1), h_F(t_2), b) \end{aligned}$$

The feature trees as we defined them have a limited feature and node label alphabet. This is not a fundamental problem. Muchnik's theorem [57] gives the decidability of MSOL of trees over arbitrary decidable structures. It is reasonable to expect that the decidability of MSOL over decidable structures yields a generalization of the result of Section 7.1 and therefore the decidability of feature trees with a richer vocabulary of features.

The crucial property of our definition of feature trees is that features can appear in any node of the tree. Hence, there are no prefix closure requirements on trees as in Section 7.3, which is responsible for relatively simple reduction to MSOL.

7.6 Reversed Binary Tree with Prefix-Closed Sets

It is instructive to compare the difficulties our approach faces in showing the decidability of structural subtyping of recursive types with the difficulties reported in [48]. In [48, Section 5.3] the authors remark that the difficulty with applying tree automata is that the set $x = f(y, z)$ is not regular. By reversing the set of paths in a tree representing a term we have shown in Section 7.1 that the relationship $x = f(y, z)$ becomes expressible. However, the difficulty now becomes specifying a set of words that represents a valid term, because there is no immediate way of stating that a set of words is prefix-closed. If we add an operation that allows expressing relationship at both "ends" of the words, we obtain a structure whose MSOL is undecidable due to the following result [52, Page 183].

Theorem 95 *MSOL theory of the structure with two successor operations $w \cdot 0$ and $w \cdot 1$ and one inverse successor operation $0 \cdot w$ is undecidable.*

The case that is of interest of us is the dual to Theorem 95 under the word-reversing isomorphism: a structure with operations $0 \cdot w$, $1 \cdot w$, $w \cdot 0$ has undecidable MSOL closed formulas.

Instead of expressing prefix-closure using operations $w \cdot 0$, $w \cdot 1$, let us consider MSOL over the structure that contains only operations $0 \cdot w$ and $1 \cdot w$, but where all second-order variables range over prefix-closed sets. This logic also turns out to be undecidable.

Let PCl be the set of prefix-closed sets. For each word w, there exists the smallest PCl set containing w, namely the set $C(w)$ given by:

$$C(w) = \{w' \mid w' \prec w\}$$

Every subset of $C(w)$ in PCl is a of the form $C(w_1)$ for some word w_1. Define PSucc_0 and PSucc_1 on PCl by:

$$\begin{aligned} \mathsf{PSucc}_0(X_1, X_2) &= \exists w.\ X_1 = C(w) \wedge X_2 = C(0 \cdot w) \\ \mathsf{PSucc}_1(X_1, X_2) &= \exists w.\ X_1 = C(w) \wedge X_2 = C(1 \cdot w) \end{aligned}$$

Consider a monadic theory PrefT with relations PSucc_0 and PSucc_1 where second-order variables range over the subsets of PCl. It is easy to see that PrefT corresponds to the first-order theory of *non-structural* subtyping of recursive types, with subset relation $\subseteq$ corresponding to subtype relation $\leq$, empty set corresponding to the least type $\perp$, $\mathsf{PSucc}_0(X_1, X_2)$ corresponding to $X_2 = f(X_1, \perp)$, and $\mathsf{PSucc}_1(X_1, X_2)$ corresponding to $X_2 = f(\perp, X_2)$. The first-order theory of non-structural subtyping was shown undecidable in [48], so PrefT is undecidable. An interesting open problem is the decidability of fragments of the first-order theory of structural subtyping. This problem translates directly to the decidability of the fragments of PrefT, a monadic theory with prefix-closed sets, or, under the word-reversal isomorphism, the decidability of fragments of the monadic theory of two successor symbols with suffix-closed sets.

8 Conclusion

In this paper we presented a quantifier elimination procedure for the first-order theory of structural subtyping of non-recursive types. Our proof uses quantifier elimination. Our decidability proof for the first-order theory of structural subtyping clarifies the structure of the theory of structural subtyping by introducing explicitly the notion of *shape* of a term.

We presented the proof in several stages with the hope of making the paper more accessible and self-contained. Our result on the decidability of Σ-term-power is more general than the decidability of structural subtyping non-recursive types, because we allow even infinite decidable base structures for primitive types. We view this decidability result as an interesting generalization of the decidability for term algebras and decidability of products of decidable theories. This generalization is potentially useful in theorem proving and program verification.

Of potential interest might be the study of axiomatizability properties; the quantifier elimination approach is appropriate for this purpose [31, 30], we did not pay much attention to this because we view the language and the mechanism for specifying the axioms of secondary importance.

Our goal in describing quantifier elimination procedure was to argue the decidability of the theory of structural subtyping. While it should be relatively easy to extract an algorithm from our proofs, we did not give a formal description of the decision procedure. One possible formulation of the decision procedure would be a term-rewriting system such as [11]; this formulation is also appropriate for implementation within a theorem prover. Our approach eliminates quantifiers as opposed to quantifier alternations. For that purpose we extended the language with partial functions. The use of Kleene logic for partial functions seems to preserve most of the properties of two valued logic and appears to agree with the way partial functions are used in informal mathematical practice. An alternative direction for proving decidability of structural subtyping would be to use Ehrenfeucht-Fraisse games [53, Page 405]; [15] uses techniques based on games to study both the decidability and the computational complexity of theories.

The complexity of our the decidability for structural subtyping non-recursive types is non-elementary and is a consequence of the non-elementary complexity of the term algebra, whose elements and operations are present in the theory of structural subtyping. Tools like MONA [25] show that non-elementary complexity does not necessarily make the implementation of a decision procedure uninteresting. An interesting property of quantifier elimination is that it can be applied partially to elimination an innermost quantifier from some formula. This property makes our decision procedure applicable as part of an interactive theorem prover or a subroutine of a more general decision procedure.

In this paper we have left open the decidability of structural subtyping of *recursive* types, giving only a few remarks in Section 7. In particular we have observed in Section 7.1 that every formula in the monadic second-order theory of the infinite binary tree [6, Page 317] has a corresponding formula in the first-order theory of structural subtyping of recursive types. In that sense, the decision problem for structural subtyping recursive types is at least as hard as the decision problem for the monadic second-order logic interpreted over the infinite binary tree. This observation is relevant for two reasons.

First, it is unlikely that a minor modification of the quantifier elimination technique we used to show the decidability of structural subtyping non-recursive types can be used to show the decidability of recursive types. Because of the embedding in Section 7.1 such a quantifier-elimination proof would have to subsume the determinization of tree automata over infinite trees.

Second, the embedding suggests even greater difficulties in implementing a decision procedure for the first-order theory of structural subtyping (provided that it exists). While we know at least one interesting example of *weak* monadic second-order logic decision procedure, namely [25] we are not aware of any implementation of the full monadic second-order logic decision procedure for the infinite tree.

The relationship between the non-structural as well as structural subtyping and monadic second-order logic of the infinite binary tree and tree like structures [58] requires further study. In that respect the work on feature trees [36, 37] appears particularly relevant.

Acknowledgements The first author would like to thank Albert Meyer for pointing out to the work [15], and Jens Palsberg and Jakob Rehof for useful discussions about the subject of this paper.

References

[1] Alexander Aiken. Introduction to set constraint-based program analysis. *Science of Computer Programming*, 35:79–111, 1999.

[2] Alexander Aiken, Dexter Kozen, and Ed Wimmers. Decidability of systems of set constraints with negative constraints. *Information and Computation*, 122, 1995.

[3] Alexander Aiken, Edward L. Wimmers, and T. K. Lakshman. Soft typing with conditional types. In *Proc. 21st ACM POPL*, pages 163–173, New York, NY, 1994.

[4] Roberto M. Amadio and Luca Cardelli. Subtyping recursive types. *Transactions on Programming Languages and Systems*, 15(4):575–631, 1993.

[5] L. O. Andersen. *Program Analysis and Specialization of the C Programming Language*. PhD thesis, DIKU, University of Copenhagen, 1994.

[6] Egon Boerger, Erich Graedel, and Yuri Gurevich. *The Classical Decision Problem*. Springer-Verlag, 1997.

[7] Witold Charatonik and Leszek Pacholski. Set constraints with projections are in NEXPTIME. In *Proc. 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 642–653, 1994.

[8] Witold Charatonik and Andreas Podelski. Set constraints with intersection. In *Proc. 12th IEEE LICS*, pages 362–372, 1997.

[9] Hubert Comon. Disunification: A survey. In Jean-Louis Lassez and Gordon Plotnik, editors, *Computational Logic: Essays in Honor of Alan Robinson*. The MIT Press, Cambridge, Mass., 1991.

[10] Hubert Comon and Catherine Delor. Equational formulae with membership constraints. *Information and Computation*, 112(2):167–216, 1994.

[11] Hubert Comon and Pierre Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7(3):371, 1989.

[12] Bruno Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, 25(2):95–169, March 1983.

[13] Rowan Davies and Frank Pfenning. Intersection types and computational effects. In *Proc. ICFP*, pages 198–208, 2000.

[14] S. Feferman and R. L. Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.

[15] Jeanne Ferrante and Charles W. Rackoff. *The Computational Complexity of Logical Theories*, volume 718 of *Lecture Notes in Mathematics*. Springer-Verlag, 1979.

[16] Tim Freeman and Frank Pfenning. Refinement types for ML. In *Proc. ACM PLDI*, 1991.

[17] Alexandre Frey. Satisfying subtype inequalities in polynomial space. *Theoretical Computer Science*, 277:105–117, 2002.

[18] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 60–65, 1982.

[19] Nevin Heintze and Olivier Tardieu. Ultra-fast aliasing analysis using CLA: A million lines of C code in a second. In *Proc. ACM PLDI*, 2001.

[20] Fritz Henglein and Jakob Rehof. The complexity of subtype entailment for simple types. In *Proc. 12th IEEE LICS*, pages 352–361, 1997.

[21] L. Henkin, J. D. Monk, and A. Tarski. *Cylindric Algebras, Part I*. North Holland, 1971.

[22] Wilfrid Hodges. *Model Theory*, volume 42 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1993.

[23] Trevor Jim and Jens Palsberg. Type inference in systems of recursive types with subtyping. `http://www.cs.purdue.edu/homes/palsberg/`, 1999.

[24] Manfred Kerber and Michael Kohlhase. A mechanization of strong Kleene logic for partial functions. In Alan Bundy, editor, *Proc. 12th CADE*, pages 371–385, Nancy, France, 1994. Springer Verlag, Berlin, Germany. LNAI 814.

[25] Nils Klarlund, Anders Møller, and Michael I. Schwartzbach. MONA implementation secrets. In *Proc. 5th International Conference on Implementation and Application of Automata*. Lecture Notes in Computer Science, 2000.

[26] Stephen Cole Kleene. *Introduction to Metamathematics*. D. Van Nostrand Company, Inc., Princeton, New Jersey, 1952. fifth reprint, 1967.

[27] Dexter Kozen. Complexity of boolean algebras. *Theoretical Computer Science*, 10:221–247, 1980.

[28] Dexter Kozen, Jens Palsberg, and Michael I. Schwartzbach. Efficient recursive subtyping. *Mathematical Structures in Computer Science*, 5(1):113–125, 1995.

[29] John W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd edition, 1987.

[30] Michael J. Maher. Complete axiomatizations of the algebras of the finite, rational, and infinite trees. *Proc. 3rd IEEE LICS*, 1988.

[31] Anatolii Ivanovic Mal'cev. *The Metamathematics of Algebraic Systems*, volume 66 of *Studies in Logic and The Foundations of Mathematics*. North Holland, 1971.

[32] Ursula Martin and Tobias Nipkow. Boolean unification: The story so far. *Journal of Symbolic Computation*, 7(3):275–293, 1989.

[33] John McCarthy and James Painter. Correctness of a compiler for arithmetic expressions. In *Proceedings of Symposia in Applied Mathematics*. American Mathematical Society, 1967.

[34] John C. Mitchell. Type inference with simple types. *Journal of Functional Programming*, 1(3):245–285, 1991.

[35] Andrzej Mostowski. On direct products of theories. *Journal of Symbolic Logic*, 17(1):1–31, March 1952.

[36] Martin Mueller and Joachim Niehren. Ordering constraints over feature trees expressed in second-order monadic logic. *Information and Computation*, 159(1/2):22–58, 2000.

[37] Martin Mueller, Joachim Niehren, and Ralf Treinen. The first-order theory of ordering constraints over feature trees. *Discrete Mathematics and Theoretical Computer Science*, 4(2):193–234, September 2001.

[38] Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM (JACM)*, 27(2):356–364, 1980.

[39] Derek C. Oppen. Reasoning about recursively defined data structures. *Journal of the ACM*, 27(3), 1980.

[40] Frank Pfenning. Unification and anti-unification in the calculus of constructions. In *Proc. 6th IEEE LICS*, pages 74–85, 1991.

[41] Francois Pottier. Simplifying subtyping constraints: A theory. *Information and Computation*, 170(2):153–183, November 2001.

[42] Jakob Rehof. *The Complexity of Simple Subtyping Systems*. PhD thesis, Computer Science Department, Univ. of Copenhagen (DIKU), April 1998.

[43] Tatiana Rybina and Andrei Voronkov. A decision procedure for term algebras with queues. *ACM Transactions on Computational Logic (TOCL)*, 2(2):155–181, 2001.

[44] Mooly Sagiv, Thomas Reps, and Reinhard Wilhelm. Parametric shape analysis via 3-valued logic. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 24(3):217–298, 2002.

[45] Jörg H. Siekmann. Unification theory. *Journal of Symbolic Computation*, 7(3):207–274, 1989.

[46] Thoralf Skolem. Untersuchungen über die Axiome des Klassenkalküls and über "Produktations- und Summationsprobleme", welche gewisse Klassen von Aussagen betreffen. Skrifter utgit av Vidnskapsselskapet i Kristiania, I. klasse, no. 3, Oslo, 1919.

[47] Bjarne Steensgaard. Points-to analysis in almost linear time. In *Proc. 23rd ACM POPL*, St. Petersburg Beach, FL, January 1996.

[48] Zhendong Su, Alexander Aiken, Joachim Niehren, Tim Priesnitz, and Ralf Treinen. First-order theory of subtyping constraints. In *Proc. 29th ACM POPL*, 2002.

[49] Madhu Sudan. Quantifier elimination for boolean algebras is trivial. Personal Communication, MIT LCS Elevator, 9 October 2002.

[50] Alfred Tarski. Arithmetical classes and types of algebraically closed and real-closed fields. *Bull. Amer. Math. Soc.*, 55, 64, 1192, 1949.

[51] Alfred Tarski. Arithmetical classes and types of boolean algebras. *Bull. Amer. Math. Soc.*, 55, 64, 1192, 1949.

[52] Wolfgang Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B*, pages 133–191. Elsevier and The MIT Press, 1990.

[53] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages Vol.3: Beyond Words*. Springer-Verlag, 1997.

[54] Jerzy Tiuryn. Subtype inequalities. In *Proc. 7th IEEE LICS*, 1992.

[55] Ralf Treinen. The first-order theory of ordering constraints over feature trees. *Discrete Mathematics and Theoretical Computer Science*, 4(2):193–234, 2001.

[56] K. N. Venkataraman. Decidability of the purely existential fragment of the theory of term algebras. *Journal of the ACM (JACM)*, 34(2):492–510, 1987.

[57] Igor Walukiewicz. Monadic second-order logic on tree-like structures. In *STACS'96*, volume 1046 of *Lecture Notes in Computer Science*, 1996.

[58] Igor Walukiewicz. Monadic second-order logic on tree-like structures. *Theoretical Computer Science*, 275(1–2):311–346, March 2002.