Structural Subtyping of Non-Recursive Types is Decidable

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

Abstract

We show that the first-order theory of structural subtyping of non-recursive types is decidable, as a consequence of a more general result on the decidability of term powers of decidable theories.

Let Σ be a language consisting of function symbols and let $\mathcal{C}$ (with a finite or infinite domain C) be an L-structure where L is a language consisting of relation symbols. We introduce the notion of Σ-term-power *of the structure $\mathcal{C}$, denoted $\mathcal{P}_\Sigma(\mathcal{C})$. The domain of $\mathcal{P}_\Sigma(\mathcal{C})$ is the set of Σ-terms over the set C. $\mathcal{P}_\Sigma(\mathcal{C})$ has one term algebra operation for each $f \in \Sigma$, and one relation for each $r \in L$ defined by lifting operations of $\mathcal{C}$ to terms over C.*

We extend quantifier elimination for term algebras and apply the Feferman-Vaught technique for quantifier elimination in products to obtain the following result. Let K be a family of L-structures and K_P the family of their Σ-term-powers. Then the validity of any closed formula F on K_P can be effectively reduced to the validity of some closed formula $q(F)$ on K.

Our result implies the decidability of the first-order theory of structural subtyping of non-recursive types with covariant constructors, and the construction generalizes to contravariant constructors as well.

1. Introduction

In this paper we show that the first-order theory of structural subtyping constraints for non-recursive types is decidable. We show this result as a consequence of a more general result on the decidability of term powers of decidable theories, which we show using quantifier elimination.

Subtyping Constraints. Subtyping constraints are an important technique for checking and inferring program properties, used both in type systems and program analyses. The study of subtyping constraints is therefore important for developing techniques that increase the reliability of programs.

Subtyping was introduced through the subsumption rule in [29]. [4, 24, 21] treat subtyping in the presence of recursive types. [49] shows that terms typable in a system with structural subtyping denote terminating computations. [12] treats intersection types in ML in the presence of computational effects. [15] presents an extension of ML that allows a more precise typing of programs than the standard ML type system. [34] shows the equivalence of non-structural subtyping and flow-analysis. Set constraints are related to the subtyping constraints and form the basis of several program analyses [2, 1, 7, 8, 5, 17].

The applications of type systems with subtyping have motivated the study of the complexity and the decidability of the subtyping constraints. [19] shows that typability is equivalent to the satisfiability of a conjunction of atomic formulas in the language of structural subtyping constraints. [16] shows that the satisfiability for structural subtyping over an arbitrary structure of base types is in PSPACE. [45] shows that if the ordering on primitive types has the form of "crowns", then the satisfiability is PSPACE hard. The need for efficient handling of constraints arising from type inference, and the need for presenting results of type inference in human-readable form led the researchers to ask more general problems about subtyping constraints [35, 39]. [18] studies the entailment problem for structural subtyping and shows that if the ordering on the primitive types is a lattice, then the entailment is coNP complete. Because the more complicated notions of subtyping involve quantifiers [47, 42], it is natural to consider the decidability and the complexity of the full *first-order theory* of subtyping constraints.

[32] studies the complexity and decidability properties of *feature tree constraints with subsumption*, which correspond closely to subtyping constraints and have applications in constraint logic programming [3] and computa-

*Produced April 7, 2003, 11:02pm for submission to LICS 2003

tional linguistics [37]. [32] shows that the first-order theory of subtyping constraints of feature trees is undecidable and that the existential entailment problem is PSPACE-complete. The first-order theory of *non-structural* subtyping constraints has been shown to be undecidable [42]. In this paper we show that the first-order theory of *structural* subtyping of non-recursive types is decidable.

This problem was left open in [42]. [42] shows the decidability of the first-order theory of non-structural subtyping for the special cases of one unary constructor symbol (where the problem is solved using tree automata techniques), as well as for the special case of one constant symbol (where the problem reduces to the decidability of term algebras).

Contribution. The main contribution of this paper is a proof that a term power of a structure with a decidable first-order theory is a structure with a decidable first-order theory. This result directly implies that the first-order theory of structural subtyping of non-recursive types is decidable. In addition, we believe that the decidability of term powers is of general interest and may be useful for constructing decision procedures in automated theorem proving. The complexity of the decidability problem for term powers is non-elementary because term powers extend term algebras. The non-elementary bound applies to term algebras as a consequence of the lower bound on the theory of pairing functions [14], see also [11].

Previous Quantifier Elimination Results. We show our decidability result using *quantifier elimination*. Quantifier elimination [20, Section 2.7] is a fruitful technique that has been used to show decidability and classification of boolean algebras [40, 44], Presburger arithmetic [36], decidability of products [30, 13], [28, Chapter 12], and algebraically closed fields [43]. Directly relevant to our work are quantifier-elimination techniques for term algebras [28, Chapter 23], [27, 41]. Several extensions of term algebras have been shown decidable using quantifier elimination. [9] gives a terminating term rewriting system for quantifier elimination in term algebras with membership constraints, [38] gives quantifier elimination for term algebras with queues, [6] presents quantifier elimination for the first-order theory of feature trees with arity predicates. [46] shows the decidability of any feature tree structure whose edge labels are elements of a decidable structure, and [48] shows the decidability of the monadic second-order theory of an infinite binary tree whose edges come from a structure with a decidable monadic second-order theory. Compared to structures in [46], term powers allow the additional lifted relations between trees, which perform a global comparison of all leaves in a tree. It may be possible to combine our technique with [46] to obtain a family of decidable structures parameterized by both the edge label theory and the leaf theory. The main difficulty in applying the result of [48] to the decidability of the full first-order theory of structural subtyping stems from the need to simultaneously represent 1) selector operations on trees (which require operations that manipulate the initial segments of paths in a tree) and the prefix-closure property of the tree domain (which requires operations that manipulate the terminal segment of paths in a tree), see [31], [25, Section 7].

Preliminaries. If A is a set, write $|A|$ to denote the cardinality of A. An *L-structure* (model) is a set together with functions and relations interpreting the language L. If S is an L-structure and $r \in L$ a function or relation symbol, write $\mathsf{ar}(r)$ to denote the arity of r. (Arity is a non-negative integer.) Write $[\![r]\!]^S$ to denote the interpretation of r in structure S. An L-formula is a first-order formula in the language L. A *sentence* is a closed formula. If K is a family of L-structures, a *theory* of K is the set of all L-sentences that are true in all structures $S \in K$. If F is a sentence, then $[\![F]\!]^K = \mathsf{true}$ if F is in the theory of K and $[\![F]\!]^K = \mathsf{false}$ otherwise. The notation $\langle E_i \rangle_i^k$ denotes the list $E_1, \ldots, E_k$ (if k is omitted, it is understood from the context).

1.1. Structural Subtyping and Σ-Term-Power

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

We represent primitive types in structural subtyping as an L_C-structure $\mathcal{C}$ with the carrier C. We call $\mathcal{C}$ the *base structure*. We assume that L_C contains only relation symbols because functions and constants can be represented as relations.

We represent type constructors as free operations in the term algebra with a finite signature Σ. Because we represent the primitive types as elements of C, we do not need constants in Σ, so we assume $\mathsf{ar}(f) \geq 1$ for each $f \in \Sigma$.

Before defining term powers, we review the notion of a finite power of a $\mathcal{C}$ structure, which is a special case of direct products of structures [20, Section 9.1, Page 413].

Definition 1 (Finite Power) *Let $m > 0$ be a positive integer and $I_m = \{0, \ldots, m-1\}$. The structure $\mathcal{C}^m$ is defined as follows. The domain of $\mathcal{C}^m$ is the set C^{I_m} of all total functions from I_m to C. Each relation $r \in L_C$ is interpreted by*

$$[\![r]\!]^{\mathcal{C}^m}(\langle t_j \rangle_j) = \ (\{i \mid [\![r]\!]^{\mathcal{C}}(\langle t_j(i) \rangle_j)\} = I_m)$$

The notion of term power is the central notion of this paper.

Definition 2 (Term Power) *The Σ-term-power of $\mathcal{C}$ is a structure $\mathcal{P} = \mathcal{P}_\Sigma(\mathcal{C})$, defined as follows. Let $\Sigma' = \Sigma \cup C$.*

The domain of $\mathcal{P}$ is the set P of finite ground Σ'-terms, where we let $\mathsf{ar}(c) = 0$ for $c \in C$.

The structure $\mathcal{P}$ has the language $L_P = \Sigma \cup L_C \cup \{\mathsf{Is}_{\mathsf{PRI}}\}$. A constructor $f \in \Sigma$ is interpreted in $\mathcal{P}$ as in the free term algebra:

$$[\![f]\!]^{\mathcal{P}}(\langle t_j \rangle_j) = f(\langle t_j \rangle_j)$$

If $r \in L_C$ with $\mathsf{ar}(r) = n$ then $[\![r]\!]^{\mathcal{P}}$ is the least relation ρ such that:

1. *if $[\![r]\!]^{\mathcal{C}}(\langle c_j \rangle_j^n)$ then $\rho(\langle c_j \rangle_j^n)$*
2. *if $\rho(\langle t_{ij} \rangle_j^n)$ for all i where $1 \leq i \leq k$, and $\mathsf{ar}(f) = k$ then*

$$\rho(\langle f(\langle t_{ij} \rangle_i^k)_j^n \rangle)$$

$\mathsf{Is}_{\mathsf{PRI}}$ is a unary relation symbol interpreted by

$$[\![\mathsf{Is}_{\mathsf{PRI}}]\!]^{\mathcal{P}}(p) \Longleftrightarrow p \in C$$

for all $p \in P$.

The reason for introducing $\mathsf{Is}_{\mathsf{PRI}}$ is that we allow C to be infinite, but we keep L_P finite. If there is a need to identify explicitly some elements of C as constants, we represent them as some of the unary relations $r \in L_C$. Note further that if F is a L_C-formula and F' results from F by replacing quantifiers with quantifiers bounded by the $\mathsf{Is}_{\mathsf{PRI}}$ predicate, then $[\![F]\!]^{\mathcal{C}}\eta = [\![F']\!]^{\mathcal{P}}\eta$ for every valuation η of $\mathcal{C}$.

2. The Decidability Result

The main result of this paper is the following Theorem 3, which states the existence of a quantifier-elimination algorithm for term powers that is uniform with respect to the structure $\mathcal{C}$.

Theorem 3 *Let L_C be a language consisting of relation symbols, Σ a language consisting of function symbols, and L_P the language of Σ-term-powers of L_C-structures. There exists a quantifier-elimination algorithm q mapping L_P-sentences to L_C-sentences such that for every structure $\mathcal{C}$ in the language L_C and for every L_P-sentence F*

$$[\![F]\!]^{\mathcal{P}_\Sigma(\mathcal{C})} = [\![q(F)]\!]^{\mathcal{C}}$$

Proposition 4 follows directly from Theorem 3.

Proposition 4 *Let L_C be a language consisting of relation symbols, Σ a language consisting of function symbols, and L_P the language of Σ-term-powers of L-structures. There exists a quantifier-elimination algorithm q mapping L_P-sentences to L_C-sentences with the following property. Let K be a family of L_C structures and*

$$K_P = \{\mathcal{P}_\Sigma(\mathcal{C}) \mid \mathcal{C} \in K\}$$

the family of Σ-term-powers of structures in K. Then $[\![F]\!]^{K_P} = [\![q(F)]\!]^K$ for every L_P-sentence F.

The following Corollary 5 captures the consequence of Theorem 3 for the theory of structural subtyping, it follows from the fact that the structure $\mathcal{C} = \langle C, \leq \rangle$ for finite C and any binary relation $\leq \subseteq C^2$ is decidable.

Corollary 5 *Let C be a finite set of primitive types and $\leq$ a binary relation on C representing an order on primitive types. Let Σ be a finite set of covariant constructors. Then the first-order theory of structural subtyping of non-recursive types built from elements of C as constants using constructors in Σ is decidable.*

Our technique can be generalized to handle *contravariant* constructors as well, see [25, Section 5.5]. The remainder of this paper sketches the proof of Theorem 3. When reading the proof the reader may find it useful to compare how our technique works in two special cases: term algebras [25, Section 3.4] and structural subtyping with two primitive types [25, Section 4]. In the case of structural subtyping with two primitive types it suffices to use quantifier-elimination for Boolean algebras [40] instead of the Feferman-Vaught theorem [30, 13].

2.1. Proof Plan

Our proof uses two main ideas.

The first idea is to extend $\mathcal{P}$ into the *extended term power* structure $\mathcal{P}_E$. The domain of the new structure $\mathcal{P}_E$ is inspired by the observation that if r is a partial order with a least element, then the relation $t_1 \sim t_2$ defined by $\exists t_0.[\![r]\!]^{\mathcal{P}}(t_0, t_1) \wedge [\![r]\!]^{\mathcal{P}}(t_0, t_2)$ is a congruence relation on P with respect to the constructor operations $[\![f]\!]^{\mathcal{P}}$ for $f \in \Sigma$. Like [45, Page 313], we call the $\sim$ equivalence classes *shapes*. A shape is an abstraction of a term obtained by throwing away the information about the constants occurring within the term, e.g. $f(a, f(a, a))$ and $f(a, f(b, a))$ both have the shape $f^{\mathsf{s}}(c^{\mathsf{s}}, f^{\mathsf{s}}(c^{\mathsf{s}}, c^{\mathsf{s}}))$. We introduce shapes as explicit elements of $\mathcal{P}_E$, and introduce into the language of $\mathcal{P}_E$ the homomorphism sh mapping terms to their shapes. Our next observation is that elements of the same $\sim$-equivalence class s together with the operations $[\![r]\!]^{\mathcal{P}}$ for $r \in L_C$ form a finite power structure $\mathcal{C}^m$ where m is the number of constants occurring in the shape s. This allows us to use the Feferman-Vaught theorem [13, 30] as a step in our quantifier elimination algorithm. To enable the application of the Feferman-Vaught technique, we introduce for every n and for every L_C-formula $\phi(\langle x_i \rangle_i^n)$ whose variables are among $\langle x_i \rangle_i^n$ relations $(|\phi(\langle x_i \rangle_i^n)|{=}k)(s, \langle t_i \rangle_i^n)$ and $(|\phi(\langle x_i \rangle_i^n)|{\geq}k)(s, \langle t_i \rangle_i^n)$ of arity $n + 1$. We call these relations *cardinality constraints*. Our cardinality constraints generalize the relations in [30] by introducing an additional shape argument s.

The second idea of our proof is the choice of canonical formulas, which we call *structural base formulas*. Structural base formulas are existentially quantified conjunctions

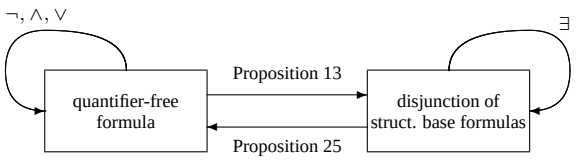

Figure 1. Scheme of Quantifier Elimination

of unnested literals that satisfy certain consistency rules. These consistency rules help justify the elimination of a quantified variable u because they ensure that the remaining conjuncts in the structural base formula entail all the relationships between the remaining variables that are a consequence of the existence of u.

Figure 1 gives a schematic view of our quantifier elimination algorithm for term powers. On the one hand, existentially quantifying a structural base formula yields a structural base formula because structural base formulas are existentially quantified conjunctions. On the other hand, the conjunction, disjunction, and most importantly, negation, of a quantifier-free formula yields a quantifier-free formula. Quantifier elimination therefore reduces to finding an effective transformation from quantifier-free formulas to disjunction of structural base formulas (Proposition 13), and from structural base formulas to quantifier-free formulas (Proposition 25).

Applying Proposition 13, then applying existential quantification and then applying Proposition 25 to obtain a quantifier free formula corresponds to the usual method of eliminating quantifiers from conjunctions of literals [20, Lemma 2.7.4, Page 70]. Dually, applying Proposition 25, negating the resulting quantifier-free formula and then applying Proposition 13 corresponds to the elimination of quantifier alternations [10, 46], [28, Chapter 23].

Several operations in the extended structure $\mathcal{P}_E$ are naturally viewed as *partial* operations. We use Kleene's three-valued logic [23, Page 334], [22] to give a systematic account of partial functions in quantifier elimination, see [25, Section 2.3]. The use of partial functions and the three-valued logic in quantifier elimination can be avoided, but we find that it naturally captures the ideas of our quantifier elimination algorithm.

2.2. 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} denotes terms that evaluate to shapes.

To specify the semantics of cardinality constraints, we define the sets $[\![\phi(x_1, \ldots, x_k)]\!]^{\mathcal{P}_E}(s, t_1, \ldots, t_k)$. We make a parallel with finite direct products [13, Definition 2.1, Page 63], [20, Section 9.6, Page 458].

Definition 6 (Index Sets for Products) *If $\phi(\langle x_j \rangle_j^n)$ is an L_C-formula whose variables are among $\langle x_j \rangle_j^n$ and $\langle t_j \rangle_j^n : I_m \to C$, then*

$$[\![\phi(\langle x_j \rangle_j^n)]\!]^{\mathcal{C}^m}(\langle t_j \rangle_j^n) = \{i \in I_m \mid [\![\phi\langle x_j \rangle_j^n]\!]^{\mathcal{C}}(\langle t_j(i) \rangle_j^n)\}$$

Define $[\![|\phi(\langle x_j \rangle_j^n)|{=}k]\!]^{\mathcal{C}^m}(\langle t_j \rangle_j^n)$ as

$$|[\![\phi(\langle x_j \rangle_j^n)]\!]^{\mathcal{C}^m}(\langle t_j \rangle_j^n)| = k,$$

similarly for $[\![|\phi(\langle x_j \rangle_j^n)|{\geq}k]\!]^{\mathcal{C}^m}(\langle t_j \rangle_j^n)$.

In the case of term powers, we replace the notion of an index $i \in I_m$ by the notion of a leaf of the tree representing a term, as follows.

Definition 7 (Leaf Sets for Term Powers) *If s is a shape, we call the set of positions of constant c^{s} in s* leaves *of s, and denote this set by* $\mathsf{leaves}(s)$. *We represent each leaf as a sequence of pairs $\langle f, i \rangle$ where f is a constructor of arity k and $1 \leq i \leq k$. 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*

$$t[l] = f^n_{i^n}(\ldots f^2_{i^2}(f^1_{i^1}(t)) \ldots)$$

Define:

$$[\![\phi(\langle x_j \rangle_j^n)]\!]^{\mathcal{P}_E}(s, \langle t_j \rangle_j^n) = \{l \in \mathsf{leaves}(s) \mid [\![\phi(\langle x_j \rangle_j^n)]\!]^{\mathcal{C}}(\langle t_j[l] \rangle_j^n)\}$$

Definition 8 (Extended Term Power) *The* extended term power structure $\mathcal{P}_E$ *contains term algebra operations on terms and shapes (including selector operations and tests as in [20, Page 61]), the homomorphism* sh, *and cardinality constraint relations $|\phi|{=}k$ and $|\phi|{\geq}k$, defined as follows:*

1. *constructors in the term algebra of terms, $f \in \Sigma$* $[\![f]\!]^{\mathcal{P}_E}(\langle t_j \rangle_j^k) {=} f(\langle t_j \rangle_j^k)$;
2. *selectors in the term algebra of terms,* $[\![f_i]\!]^{\mathcal{P}_E}(f(\langle t_j \rangle_j^k)) = t_i$;
3. *constructor tests in the term algebra of terms,* $[\![\mathsf{Is}_f]\!]^{\mathcal{P}_E}(t) = \exists \langle t_j \rangle_j^k.\ t = f(\langle t_j \rangle_j^k)$, $[\![\mathsf{Is}_{\mathsf{PRI}}]\!]^{\mathcal{P}_E}(t) = (t \in C)$;

4. *constructors in the term algebra of shapes,* $f^{\mathsf{s}} \in \Sigma^{\mathsf{s}}$ $[\![f^{\mathsf{s}}]\!]^{\mathcal{P}_E}(\langle t_j^{\mathsf{s}}\rangle_j^k) = f^{\mathsf{s}}(\langle t_j^{\mathsf{s}}\rangle_j^k)$*;*
5. *selectors in the term algebra of shapes,* $[\![f_i^{\mathsf{s}}]\!]^{\mathcal{P}_E}(f^{\mathsf{s}}(\langle t_j^{\mathsf{s}}\rangle_j^k)) = t_i^{\mathsf{s}}$*;*
6. *constructor tests in the term algebra of shapes,* $[\![\mathsf{ls}_{f^{\mathsf{s}}}]\!]^{\mathcal{P}_E}(t^{\mathsf{s}}) = \exists \langle t_j^{\mathsf{s}}\rangle_j^k.\ t^{\mathsf{s}} = f^{\mathsf{s}}(\langle t_j^{\mathsf{s}}\rangle_j^k)$*;*
7. *the homomorphism mapping terms to shapes such that:*

$$[\![\mathsf{sh}]\!]^{\mathcal{P}_E}(f(\langle t_j\rangle_j^k)) = \mathsf{shapified}(f)(\langle [\![\mathsf{sh}]\!]^{\mathcal{P}_E}(t_j)\rangle_j^k)$$

where $\mathsf{shapified}(x)=c^{\mathsf{s}}$ *if* $x \in C$ *and* $\mathsf{shapified}(f)=f^{\mathsf{s}}$ *if* $f \in \Sigma$*;*

8. *cardinality constraint relations*

$$[\![|\phi(\langle x_j\rangle_j^n)|{=}k]\!]^{\mathcal{P}_E}(s, \langle t_j\rangle_j^n) = |[\![\phi(\langle x_j\rangle_j^n)]\!]^{\mathcal{P}_E}(s, \langle t_j\rangle_j^n)| = k$$

and

$$[\![|\phi(\langle x_j\rangle_j^n)|{\geq}k]\!]^{\mathcal{P}_E}(s, \langle t_j\rangle_j^n) = |[\![\phi(\langle x_j\rangle_j^n)]\!]^{\mathcal{P}_E}(s, \langle t_j\rangle_j^n)|{\geq}k$$

where $\phi(\langle x_j\rangle_j^n)$ *is is a first-order formula over the base-structure language* L_C *with free variables* $\langle x_j\rangle_j^n$*, argument* s *is a shape, arguments* $\langle t_j\rangle_j^n$ *are terms, and* k *is a nonnegative integer constant.*

The following equations follow from Definition 8 and Definition 7 and can be used as an equivalent alternative definition of cardinality constraints:

$$|[\![\phi(\langle x_j\rangle_j^n)]\!]^{\mathcal{P}_E}(c^{\mathsf{s}}, \langle c_j\rangle_j^n)| = \begin{cases} 1, & [\![\phi(\langle x_j\rangle_j^n)]\!]^{\mathcal{C}}(\langle c_j\rangle_j^n) \\ 0, & \neg[\![\phi(\langle x_j\rangle_j^n)]\!]^{\mathcal{C}}(\langle c_j\rangle_j^n) \end{cases}$$

$$|[\![\phi(\langle x_j\rangle_j^n)]\!]^{\mathcal{P}_E}(f^{\mathsf{s}}(\langle s_i\rangle_i^k), \langle f(\langle t_{ij}\rangle_i^k)\rangle_j^n)| = \textstyle\sum_{i=1}^k |[\![\phi(\langle x_j\rangle_j^n)]\!]^{\mathcal{P}_E}(s_i, \langle t_{ij}\rangle_j^n)| \quad (1)$$

We write $|\phi(\langle t_j\rangle_j^n)|_s{=}k$ as a shorthand for the atomic formula $(|\phi(\langle x_j\rangle_j^n)|{=}k)(s, \langle t_j\rangle_j^n)$, similarly for $|\phi(\langle t_j\rangle_j^n)|_s{\geq}k$. This is more than a notational convenience, see [25] 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 the present paper follows [30] in merging the quantifier elimination for products and quantifier elimination for boolean algebras.

Some of the operations in $\mathcal{P}_E$ are partial. $f_i(t)$ is defined iff $\mathsf{ls}_f(t)$ holds, $f_i^{\mathsf{s}}(t^{\mathsf{s}})$ is defined iff $\mathsf{ls}_{f^{\mathsf{s}}}(t^{\mathsf{s}})$ holds. Cardinality constraints $|\phi(\langle t_j\rangle_j^n)|_{t^{\mathsf{s}}}{=}k$ and $|\phi(\langle t_j\rangle_j^n)|_{t^{\mathsf{s}}}{\geq}k$ are defined iff $\wedge_{i=1}^n \mathsf{sh}(t_i){=}t^{\mathsf{s}}$ holds. We assume that a term evaluates to $\bot$ if some term operation is undefined. Partial and total operations are strict in $\bot$; when a value of atomic formula is undefined it evaluates to undef. Logical operations and quantifiers are interpreted as in Kleene's three-valued logic with truth values $\{\mathsf{false}, \mathsf{undef}, \mathsf{true}\}$. We say that a formula is *well-defined* iff it evaluates to true or false (as opposed to undef) for every valuation assigning values to free variables. The structure $\mathcal{P}_E$ has the property that the domain of every partial function is expressible as a conjunction of atomic formulas. This property enables transformation of each well-defined quantifier-free formula to a disjunction of well-defined conjunctions in Proposition 13, see also [25, Section 2.3].

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

$$|\neg\, r(\langle t_j\rangle_j^n)|_{\mathsf{sh}(t_1)}{=}0 \ \wedge \bigwedge_{i=2}^{k} \mathsf{sh}(t_i) = \mathsf{sh}(t_1) \quad (2)$$

By a quantifier-free formula we mean a formula without quantifiers outside cardinality constraints, e.g. the formula $|\forall x.x \leq t|_{x^{\mathsf{s}}} = k$ is quantifier-free.

We define a subclass of quantifier-free cardinality constraints called *primitive formulas*, denoted $\mathsf{prim}(\phi)$ for every L_C-sentence ϕ: $\mathsf{prim}(\phi) \equiv |\phi|_{c^{\mathsf{s}}} = 1$. Note that

$$[\![\mathsf{prim}(\phi)]\!]^{\mathcal{P}_\Sigma(\mathcal{C})} = [\![\phi]\!]^{\mathcal{C}} \quad (3)$$

so for a given concrete structure $\mathcal{C}$ we may replace primitive formulas with true and false. We nevertheless retain primitive formulas throughout the quantifier elimination algorithm. This ensures that our quantifier elimination algorithm is uniform wrt. the base structure $\mathcal{C}$. In the sequel we therefore assume some fixed structure $\mathcal{C}$ and proceed to give a quantifier elimination algorithm that performs equivalence-preserving transformations wrt. the extended term power $\mathcal{P}_E$ corresponding to $\mathcal{P}_\Sigma(\mathcal{C})$.

2.3. Structural Base Formulas

Our quantifier-elimination algorithm is centered around certain existentially quantified unnested conjunctions of literals. We call these conjunctions *structural base formulas*.

We first introduce several auxiliary definitions. Let $\mathsf{distinct}(u_1, \ldots, u_n)$ be a shorthand for $\bigwedge_{1\leq i<j\leq n} u_i \neq u_j$. If ϕ is a formula and x and y two term variables, then $x \rightarrowtail_\phi y$ means that ϕ contains a conjunct of the form $x = f(y_1, \ldots, y, \ldots, y_k)$ for some $f \in \Sigma$. Similarly if x^{s} and y^{s} are two shape variables then $x^{\mathsf{s}} \rightarrowtail_\phi y^{\mathsf{s}}$ means that ϕ contains a conjunct of the form $x^{\mathsf{s}} = f^{\mathsf{s}}(y_1^{\mathsf{s}}, \ldots, y^{\mathsf{s}}, \ldots, y_k^{\mathsf{s}})$ for some $f \in \Sigma$. The relation $\rightarrowtail_\phi^+$ is the non-reflexive transitive closure of $\rightarrowtail_\phi$. We next define base formulas for term algebras and state some of their properties; [25] presents a quantifier elimination procedure for term algebras based on these definitions.

Definition 9 (Base Formula) *A* 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* $u_{p+1}, \ldots, u_{p+q}$*;*

is a formula of the form:

$$\mathsf{base}(u_1, \ldots, u_n, x_1, \ldots, x_m) = \bigwedge_{i=1}^{p} u_i = t_i(u_1, \ldots, u_n) \wedge \bigwedge_{i=1}^{m} x_i = u_{j_i} \wedge \mathsf{distinct}(u_1, \ldots, u_n)$$

where $n = p + q$*, 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.*

We require each base formula to satisfy the following conditions:

C1) base *does not violate the occur-check [26, 10]:* $\neg(u \rightarrowtail^{+}_{\mathsf{base}} u)$ *for every variable* u *occurring in* base*;*

C2) congruence closure property: 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* base*.*

The following Lemma 10 is important for quantifier elimination in term algebras and term powers.

Lemma 10 *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* $[\![\beta_0]\!]\sigma =$ true *and* $[\![u_i]\!]\sigma \in S_i$ *for* $p + 1 \leq i \leq p + q$*.*

The notion of base formula and Lemma 10 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 a constructor in the term algebra of terms or shapes, and f^*_i for a selector in the term algebra of terms or shapes.

Definition 11 below introduces *structural* base formulas. The disjunction of structural base formulas can be thought of as a normal form for existential formulas interpreted over $\mathcal{P}_E$. A structural base formula contains a copy of a base formula for shapes (shapeBase), a base formula for terms but without term disequalities (termBase), a formula expressing mapping of term variables to shape variables (termHom), and cardinality constraints on term parameter nodes of the term base formula (cardin). A structural base formula contains several kinds of variables, classified according to the positions in which they appear within the structural base formula. Free variables are the free variables of the structural base formula; internal variables are the existentially quantified variables. Parameter variables are variables whose top-level constructor is not specified by the structural base formula, in contrast to non-parameter variables. Primitive non-parameter term variables denote terms in C, composed non-parameter term variables denote terms in $P \setminus C$.

Definition 11 (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 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 the form:

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

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

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

where each t_i *is a shape term of the 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.*

$$\mathsf{termBase}(u_1, \ldots, u_n, x_1, \ldots, x_m) = \bigwedge_{i=1}^{r} u_i = t_i(u_1, \ldots, u_n) \wedge \bigwedge_{i=r+1}^{p} \mathsf{Is}_{\mathsf{PRI}}(u_i) \wedge \bigwedge_{i=1}^{m} x_i = u_{j_i}$$

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.

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

where $j:\{1,\ldots,n\}\to\{1,\ldots,n^{\mathsf{s}}\}$ *is a 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).*

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

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 $\{j_1,\ldots,j_l\}\subseteq\{r+1,\ldots,n\}$ *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) shapeBase *does not violate the occur-check:* $\neg(u^{\mathsf{s}} \rightarrowtail^{+}_{\mathsf{shapeBase}} u^{\mathsf{s}})$ *for every shape variable* u^{s} *occurring in* shapeBase*;*

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 composed 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*; furthermore: for every primitive non-parameter variable* u *(i.e.* u *s.t.* $\mathsf{Is}_{\mathsf{PRI}}u$ *occurs in* termBase*), conjunct* $\mathsf{sh}(u) = u^{\mathsf{s}}$ *occurs in* termHom *where* u^{s} *is the shape variable such that* $u^{\mathsf{s}} = c^{\mathsf{s}}$ *occurs in* shapeBase*.*

As a special case, we allow quantifier-free formulas $\mathsf{prim}(\phi)$ in cardin. Note that $\neg(u \rightarrowtail^{+}_{\mathsf{termBase}} u)$ for each term variable u follows from P0) and P3). An immediate consequence of Definition 11 is the following Proposition 12.

Proposition 12 (Quantification of Structural Base) *If* β *is a structural base formula and* x *a free shape or term variable in* β*, then there exists a structural structural base formula* β_1 *equivalent to* $\exists x.\beta$*.*

For example, if $\beta \equiv \exists u,u^{\mathsf{s}}.\,\mathsf{sh}(u){=}u^{\mathsf{s}}\wedge x{=}u$ then $\exists x.\beta$ is equivalent to $\exists u,u^{\mathsf{s}}.\,\mathsf{sh}(u){=}u^{\mathsf{s}}$ where $x{=}u$ conjunct was removed.

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

2.4. Conversion to Structural Base Formulas

The conversion to structural base formulas builds on the conversion to disjunctions of well-defined conjunctions of unnested literals [25, Section 2.3], congruence closure algorithms [33], and the equality (1).

Proposition 13 (Quantifier-Free to Structural Base) *Every well-defined quantifier-free formula* ϕ *is equivalent on* $\mathcal{P}_E$ *to* true*,* false*, or a disjunction of structural base formulas.*

Proof Sketch. We outline an algorithm for converting ϕ into a disjunction of structural base formulas. Rules for performing the transformation are presented in the Appendix.

First convert ϕ into the disjunctive normal form (DNF) using rules DNF. These rules are valid in three-valued logic because the three-valued domain is a distributive lattice, $\neg$ is idempotent and DeMorgan's laws hold. For example, $\neg(\mathsf{Is}_f(x)\wedge y{=}f_1(x))$ gets transformed into $\neg\mathsf{Is}_f(x)\vee y{\neq}f_1(x)$. The resulting DNF is well-defined, but the individual conjunctions (e.g. $y{\neq}f_1(x)$) need not be well-defined. Applying rules WDNF to all conjuncts yields a disjunction of well-defined conjunctions (e.g. $y\neq f_1(x)$ becomes $\mathsf{Is}_f(x)\wedge y\neq f_1(x)$). This transformation preserves the equivalence because the starting disjunction was well-defined, see [25, Section 2.3].

The next step converts the formula into the unnested (flat) form by introducing existentially quantified variables for subterms and free variables, using rules UNF (e.g. $x{=}f(f(y,z),y)$ becomes $\exists u.\,u{=}f(y,z)\wedge x{=}f(u,y)$ whereas $y{\neq}f_1(x)$ becomes $\exists u.u{=}f_1(x)\wedge y{\neq}u$). The result is a disjunction of well-defined existentially quantified conjunctions of unnested literals. Apply rules ELNG to eliminate negations of all atomic formulas except for disequalities (e.g. if $\Sigma=\{f\}$ then $\neg\mathsf{Is}_f(x)$ becomes $\mathsf{Is}_{\mathsf{PRI}}(x)$). ELNG rules may violate DNF; use DNF rules again to reestablish the normal form (this also applies to all subsequent rules that may violate DNF). Eliminate selector functions and constructor tests using rules SelEl (e.g. if f is a binary constructor, then $\exists u.\,\mathsf{Is}_f(x)\wedge u{=}f_1(x)$

becomes $\exists u, v_1, v_2.\, x{=}f(v_1, v_2) \wedge u{=}v_1$). The result contains only the relation and function symbols that occur in structural base formulas. Make sure each term variable has a corresponding shape variable by applying rules ShpInt. For example, $\exists v_1, v_2.\, x{=}f(v_1, v_2)$ becomes $\exists v_1, v_2, x^{\mathsf{s}}, v_1^{\mathsf{s}}, v_2^{\mathsf{s}}.\, \mathsf{sh}(v_1){=}v_1^{\mathsf{s}} \wedge \mathsf{sh}(v_2){=}v_2^{\mathsf{s}} \wedge \mathsf{sh}(x){=}x^{\mathsf{s}} \wedge x{=}f(v_1, v_2)$. Next, apply congruence closure (CongCl) and occur check (OccChk). For example, $\exists x, u, v_1, v_2.\, x{=}f(v_1, v_2) \wedge y{=}f(u, v_2) \wedge u{=}v_1$ becomes $\exists x, u, v_2.\, x{=}f(u, v_2) \wedge y{=}x$, whereas $x{=}f(u, v) \wedge u{=}f(x, v)$ becomes false. Use HomExp rules to ensure that parameter term variables are mapped to parameter shape variables, non-parameter term variables are mapped to non-parameter shape variables, and that the homomorphism property P3) of Definition 11 holds. Repeat CongCl and OccChk rules if needed. For example, $\exists v_1, v_2, x^{\mathsf{s}}, v_1^{\mathsf{s}}, v_2^{\mathsf{s}}.\, \mathsf{sh}(v_1){=}v_1^{\mathsf{s}} \wedge \mathsf{sh}(v_2){=}v_2^{\mathsf{s}} \wedge \mathsf{sh}(x){=}x^{\mathsf{s}} \wedge x{=}f(v_1, v_2)$ becomes $\exists v_1, v_2, x^{\mathsf{s}}, v_1^{\mathsf{s}}, v_2^{\mathsf{s}}.\, \mathsf{sh}(v_1){=}v_1^{\mathsf{s}} \wedge \mathsf{sh}(v_2){=}v_2^{\mathsf{s}} \wedge \mathsf{sh}(x){=}x^{\mathsf{s}} \wedge x{=}f(v_1, v_2) \wedge x^{\mathsf{s}}{=}f^{\mathsf{s}}(v_1^{\mathsf{s}}, v_2^{\mathsf{s}})$. Eliminate all disequalities between term variables using the NEQEl rule, which is justified by the negation of the equivalence:

$$t_1 = t_2 \iff \mathsf{sh}(t_1) = \mathsf{sh}(t_2) \wedge |t_1 = t_2|_{\mathsf{sh}(t_1)} = 0 \quad (4)$$

For example, $u{\neq}x \wedge \mathsf{sh}(u){=}u^{\mathsf{s}} \wedge \mathsf{sh}(x){=}x^{\mathsf{s}}$ becomes $((u^{\mathsf{s}}{\neq}x^{\mathsf{s}}) \vee (u^{\mathsf{s}}{=}x^{\mathsf{s}} \wedge |u{\neq}x|_{u^{\mathsf{s}}} {\geq} 1)) \wedge \mathsf{sh}(u){=}u^{\mathsf{s}} \wedge \mathsf{sh}(x){=}x^{\mathsf{s}}$. Repeat previous stages (e.g. DNF, CongCl, OccChk) if needed. Convert all cardinality constraints into constraints on parameter term variables, using CCD rules justified by (1), e.g. $|u{\neq}v|_{u^{\mathsf{s}}}{=}1$ becomes $(|u_1{\neq}v_1|_{u_1^{\mathsf{s}}}{=}0 \wedge |u_2{\neq}v_2|_{u_2^{\mathsf{s}}}{=}1) \vee (|u_1{\neq}v_1|_{u_1^{\mathsf{s}}}{=}1 \wedge |u_2{\neq}v_2|_{u_2^{\mathsf{s}}}{=}0)$ in the context of $u{=}f(u_1, v_1) \wedge v{=}f(v_1, v_2) \wedge u^{\mathsf{s}}{=}f^{\mathsf{s}}(u_1^{\mathsf{s}}, u_2^{\mathsf{s}}) \wedge \mathsf{sh}(u){=}\mathsf{sh}(v){=}u^{\mathsf{s}} \wedge \mathsf{sh}(u_1){=}\mathsf{sh}(v_1){=}u_1^{\mathsf{s}} \wedge \mathsf{sh}(u_2){=}\mathsf{sh}(v_2){=}u_2^{\mathsf{s}}$. Finally, to produce the formula $\mathsf{distinct}(u_1^{\mathsf{s}}, \ldots, u_n^{\mathsf{s}})$ use ShDis to ensure that for every two shape variables x_1^{s} and x_2^{s} occurring in the conjunction exactly one of the conjuncts $x_1^{\mathsf{s}}{=}x_2^{\mathsf{s}}$ or $x_1^{\mathsf{s}}{\neq}x_2^{\mathsf{s}}$ is present. ■

2.5. Conversion to Quantifier-Free Formulas

The conversion from structural base formulas to quantifier-free formulas is the main phase of our quantifier-elimination algorithm. We split this conversion into several stages; Proposition 25 below summarizes the overall conversion process.

Consider a structural base formula $\beta \equiv\ \exists \bar{u}^*.\, C(\bar{x}^*, \bar{u}^*)$ with free variables $\bar{x}^*$ and internal variables $\bar{u}^*$, where $C(\bar{x}^*, \bar{u}^*)$ is quantifier-free. $C(\bar{x}^*, \bar{u}^*)$ defines a relation between variables $\bar{x}^*, \bar{u}^*$. If this relation has a functional dependence from the free variables $\bar{x}^*$ to some internal variable u, with a term $t(\bar{x}^*)$ such that $C(\bar{x}^*, \bar{u}^*) \models u = t(\bar{x}^*)$, then the internal variable u can be replaced by $t(\bar{x}^*)$ and the quantification over u can be eliminated. This leads to the notion of *determinations*.

Definition 14 *The set* dets *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 *for a free variable x^*, 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 $\{\langle u_1^*, t_1^*\rangle, \ldots, \langle u_k^*, t_k^*\rangle\} \subseteq S$ and $u^* = f^*(u_1^*, \ldots, u_k^*)$ occurs in* shapeBase *or* termBase *then $\langle u^*, f^*(t_1^*, \ldots, t_k^*)\rangle \in S$;*
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 15 *An internal variable u^* is* determined *if $\langle u^*, t^* \rangle \in$ dets for some term t^{s}. An internal variable is* undetermined *if it is not determined.*

Lemma 16 follows by induction using Definition 14.

Lemma 16 *Let $\beta \equiv\ \exists \bar{u}.\, C(\bar{x}^*, \bar{u}^*)$ be a structural base formula. If $\langle u^*, t^* \rangle \in \mathsf{dets}(\beta)$ then $C(\bar{x}^*, \bar{u}^*) \models u^* = t^*$.*

Corollary 17 *Let $\beta \equiv\ \exists \langle u_i^* \rangle_i.\, C(\bar{x}^*, \langle u_i^* \rangle_i)$ be a structural base formula such that each internal variable u_i^* is determined by some term t_i^*, that is, $\langle u_i^*, t_i^* \rangle \in \mathsf{dets}(\beta)$. Then β is equivalent to the well-defined quantifier-free formula $\beta' \equiv\ C(\bar{x}^*, \langle t_i^* \rangle_i)$.*

Proof. By Lemma 16 using the rule

$$\exists u.u = t \wedge \phi(u) \iff \phi(t) \quad (5)$$

which holds when the term t is well-defined. If t is not well-defined, then both β and β' evaluate to false. ■

Our goal thus reduces to eliminating all undetermined variables from a structural base formula. We first show how to eliminate undetermined composed non-parameter term variables.

Lemma 18 *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 the form $u'{=}f(u_1, \ldots, u, \ldots, u_k)$ occurs in* termBase*. Let β' be the result of removing from β the variable u and all conjuncts containing u. Then β is equivalent to β'.*

Proof. The conjunct containing $\mathsf{sh}(u) = u^{\mathsf{s}}$ in $\mathsf{termHom}$ is a consequence of the remaining conjuncts in β, so we may drop it. The only remaining occurrence of u is in the atomic formula $u{=}f(\bar{v})$ of $\mathsf{termBase}$ subformula. Applying (5) therefore makes u disappear from β. ∎

Corollary 19 (Composed Term Variable Elimination) *Dropping all undetermined composed non-parameter term variables from a structural base formula together with the conjuncts that contain them yields an equivalent structural base formula.*

Proof. If a structural base formula has an undetermined non-parameter composed term variable, then it has an undetermined non-parameter composed term variable that is a source. Repeatedly apply Lemma 18 to eliminate all undetermined non-parameter term variables. ∎

Our next goal is to eliminate undetermined primitive non-parameter term variables and undetermined parameter term variables. The key insight is that these variables are related to the determined variables of a structural base formula only through the relations that are expressible in the product structure of the terms of the same shape. To clarify the connection with the product-structure, let $s \in P_S$ be a shape and $P^{(s)} = \{t \in P \mid \mathsf{sh}(t) = s\}$. Define $\eta : P^{(s)} \to C^*$ where C^* is the set of finite sequences of elements from C, as follows: $\eta(c) = c$ if $c \in C$; $\eta(f(t_1, \ldots, t_k)) = \eta(t_1) \cdot \ldots \cdot \eta(t_k)$ where $l_1 \cdot l_2$ denotes the concatenation of sequences l_1 and l_2. Let $\eta_s = \eta|_{P^{(s)}}$ be the restriction of η to the set $P^{(s)}$. Let $m = |\mathsf{leaves}(s)|$.

Observation 20 *The map η_s is an isomorphism of the substructure of $\mathcal{P}$ with the domain $P^{(s)}$ and the finite power $\mathcal{C}^m$. Moreover,*

$$|[\![\phi(\langle x_j\rangle_j^n)]\!]^{\mathcal{P}_E}(s, \langle t_j\rangle_j^n)| = |[\![\phi(\langle x_j\rangle_j^n)]\!]^{\mathcal{C}^m}(\langle \eta_s(t_j)\rangle_j^n)|$$

The following is the quantifier-elimination property that implies Feferman-Vaught theorem [13, 30], [20, Section 9.6, Page 460] for the case of finite products.

Lemma 21 *Let $k \geq 0$. Consider a formula α of the form $\alpha \equiv \exists u.\ \bigwedge_{i=1}^{p} \psi_i$ where each ψ_i is a cardinality constraint of the form $(|\phi_i|{=}k)(u, \langle u_j\rangle_j^n)$ or $(|\phi_i|{\geq}k)(u, \langle u_j\rangle_j^n)$. Then α can be effectively transformed into α' where α' is a disjunction of conjunctions of cardinality constraints of the form $(|\phi'_i|{=}k)(\langle u_j\rangle_j^n)$ and $(|\phi'_i|{\geq}k)(\langle u_j\rangle_j^n)$. The result α' is equivalent to α on each finite power $\mathcal{C}^m$.*

Lemma 22 is a direct consequence of Lemma 21 and Observation 20.

Lemma 22 *Let $k \geq 0$. Consider a formula α of the form*

$$\alpha \ \equiv\ \exists u.\ \ \mathsf{sh}(u) = u^{\mathsf{s}} \ \wedge \textstyle\bigwedge_{i=1}^{n} \mathsf{sh}(u_i) = u^{\mathsf{s}} \ \wedge \bigwedge_{i=1}^{p} \psi_i$$

for some shape variable u^{s} where each ψ_i is a cardinality constraint of the form $(|\phi_i|{=}k)(u^{\mathsf{s}}, u, \langle u_j\rangle_j^n)$ or of the form $(|\phi_i|{\geq}k)(u^{\mathsf{s}}, u, \langle u_j\rangle_j^n)$. Then α can be effectively transformed into the formula α' which is a disjunction of formulas of the form

$$\alpha'_j \ \equiv\ \ \textstyle\bigwedge_{i=1}^{n} \mathsf{sh}(u_i) = u^{\mathsf{s}} \ \wedge \bigwedge_{i=1}^{q} \psi'_{i,j}$$

where each $\psi'_{i,j}$ is of the form $(|\phi'_{i,j}|{=}k)(u^{\mathsf{s}}, \langle u_j\rangle_j^n)$ or $(|\phi'_{i,j}|{\geq}k)(u^{\mathsf{s}}, \langle u_j\rangle_j^n)$. The resulting formula α' is equivalent to α on all term powers $\mathcal{P}_E$.

Lemma 23 (Term Parameter Elimination) *Every structural base formula β without undetermined composed non-parameter term variables can be effectively transformed into an equivalent disjunction of structural base formulas without undetermined term variables.*

Proof. We show how to eliminate undetermined parameter term variables and undetermined primitive non-parameter term variables from β.

Let u be an undetermined parameter term variable or an undetermined primitive non-parameter term variable. If u is a parameter variable then u does not occur in $\mathsf{termBase}$ because $\neg(u \rightarrowtail^{+} u')$ for all u', and $\neg(u'' \rightarrowtail^{+} u)$ for all u'' since there are no undetermined composed non-parameter term variables. Therefore, u occurs only in $\mathsf{termHom}$ and cardin. If u is a primitive non-parameter term variable, then $\mathsf{termBase}$ contains only one occurrence of u, namely the conjunct $\mathsf{ls}_{\mathsf{PRI}}(u)$, which is a consequence of the conjuncts $\mathsf{sh}(u) = u^{\mathsf{s}}$ in $\mathsf{termHom}$ and $u^{\mathsf{s}} = c^{\mathsf{s}}$ in $\mathsf{shapeBase}$, so we drop $\mathsf{ls}_{\mathsf{PRI}}(u)$. In both cases, the resulting formula contains u only in $\mathsf{termHom}$ and cardin.

Let u^{s} be the shape variable such that $u^{\mathsf{s}} = \mathsf{sh}(u)$ occurs in $\mathsf{termHom}$. 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 $\mathsf{termHom}$. The structural base formula can therefore be written in the form $\exists \bar{u}^*.\ \phi \wedge \alpha$ where α has the form as in Lemma 22. Applying Lemma 22 we eliminate u. Applying rules DNF results in a disjunction of structural base formulas. By repeating this process we eliminate all undetermined parameter term variables and undetermined primitive non-parameter term variables from a structural base formula. Each of the resulting structural base formulas contains no undetermined term variables. ∎

Finally, we show how to eliminate the undetermined shape variables.

Lemma 24 (Shape Variable Elimination) *Every structural base formula β without undetermined term variables can be effectively transformed into an equivalent disjunction of structural base formulas without undetermined variables.*

Proof Sketch. It remains to eliminate undetermined shape variables from β. This process is similar to term algebra quantifier elimination [25, Section 3.4]; the key ingredient is Lemma 10, which relies on the fact that undetermined parameter variables may take on infinitely many values. We therefore ensure that the conjuncts outside shapeBase do not constrain the undetermined parameter shape variables to denote the values from a finite set.

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 term variable u and since all term variables are determined, 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 the domain size appears in [30, Theorem 3.36, Page 13]. In term powers, these constraints become constraints on the size of the shape.) We transform β into the disjunction $\beta_1 \vee \beta_2$ of base formulas where $\beta_1 \equiv \beta \wedge \mathsf{prim}(\phi)$ and $\beta_2 \equiv \beta \wedge \mathsf{prim}(\neg\phi)$. Constraints of the form $\mathsf{prim}(\neg\phi) \wedge |\phi|_{u^{\mathsf{s}}}{=}k$ reduce to $0{=}k$, we replace them by true if $k{\equiv}0$ and false if $k{\not\equiv}0$. On the other hand, $\mathsf{prim}(\phi) \wedge |\phi|_{u^{\mathsf{s}}}{=}k$ denotes the constraint $m = k$ and $\mathsf{prim}(\phi) \wedge |\phi|_{u^{\mathsf{s}}}{\geq}k$ denotes $m{\geq}k$. Hence, by repeating this process for every formula ϕ which appears in some cardinality constraint $|\phi|_{u^{\mathsf{s}}}{=}k$ or $|\phi|_{u^{\mathsf{s}}}{\geq}k$, we obtain a conjunction of 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 $A = \{s \mid |\mathsf{leaves}(s)| \in S\}$. By nature of our constraints, 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 also infinite. If Σ contains no unary constructors and S is finite then A is finite and we can effectively compute A. The cardinality constraints containing u^{s} are thus equivalent to $\bigvee_{i=1}^{p} u^{\mathsf{s}} = t_i^{\mathsf{s}}$ where $A = \{t_1^{\mathsf{s}}, \ldots, t_p^{\mathsf{s}}\}$. Transform the structural base formula β into a disjunction of formulas $\bigvee_{i=1}^{p} \beta_i$ where β_i results from β by replacing the cardinality constraints containing u^{s} with $u^{\mathsf{s}} = t_i^{\mathsf{s}}$. Convert each β_i to a structural base formula by labelling the subterms of t_i^{s} with internal shape variables using UNF rules, and by doing case analysis on the equality between the new internal shape variables, using ShDis rule. 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 because Lemma 10 implies that it is always possible to find the values of eliminated parameter variables, so their existence is a redundant condition. We therefore eliminate all undetermined shape variables and the resulting structural base formulas contain only determined variables. ■

Proposition 25 (Struct. Base to Quantifier-Free) *Every structural base formula β can be effectively transformed to an equivalent well-defined quantifier-free formula ϕ.*

Proof. Apply Corollary 19, then Lemma 23, and then Lemma 24. All variables in the resulting disjunction of structural base formulas are determined, so each of them is equivalent to some quantifier free formula ϕ_i by Corollary 17. The disjunction $\bigvee_i \phi_i$ is the desired quantifier-free formula ϕ. ■

Summary of Our Quantifier Elimination Algorithm. Consider a closed L_P-formula ϕ. Convert ϕ to an extended-term-power formula ϕ_1 using (2). Convert ϕ_1 to prenex form ϕ_2. Eliminate all quantifiers from ϕ_2 starting from the innermost one, as follows. If $\phi_2 \equiv \langle Q_i u_i^* \rangle_i \exists v^*.\, \psi$ where ψ is quantifier-free then apply Proposition 13, Proposition 12 and then Proposition 25. If $\phi_2 \equiv \langle Q_i u_i^* \rangle \forall v^*.\, \psi$ then consider $\langle Q_i u_i^* \rangle . \neg\exists v^*.\, \neg\psi$ and proceed as in the previous case. By applying Proposition 13 and Proposition 25 to the resulting variable-free formula we obtain a propositional combination of $\mathsf{prim}(\phi)$ formulas. Theorem 3 then follows by (3).

Acknowledgements We thank Albert Meyer, Jens Palsberg, Tim Priesnitz, Stefan Ratschan, Jakob Rehof, Zhendong Su, and anonymous reviewers for useful comments.

References

[1] A. Aiken, D. Kozen, and E. Wimmers. Decidability of systems of set constraints with negative constraints. *Information and Computation*, 122, 1995.

[2] A. Aiken, E. L. Wimmers, and T. K. Lakshman. Soft typing with conditional types. In *Proc. 21st ACM POPL*, pages 163–173, New York, NY, 1994.

[3] H. Ait-Kaci, A. Podelski, and G. Smolka. A feature constraint system for logic programming with entailment. In *Theoretical Computer Science*, volume 122, pages 263–283, January 1994.

[4] R. M. Amadio and L. 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] R. Backofen. A complete axiomatization of a theory with feature and arity constraints. *Journal of Logic Programming*, 24:37–72, 1995.

[7] W. Charatonik and L. Pacholski. Set constraints with projections are in NEXPTIME. In *Proc. 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 642–653, 1994.

[8] W. Charatonik and A. Podelski. Set constraints with intersection. In *Proc. 12th IEEE LICS*, pages 362–372, 1997.

[9] H. Comon and C. Delor. Equational formulae with membership constraints. *Information and Computation*, 112(2):167–216, 1994.

[10] H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7(3):371, 1989.

[11] K. J. Compton and C. W. Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *Annals of Pure and Applied Logic*, 48(1):1–79, July 1990.

[12] R. Davies and F. Pfenning. Intersection types and computational effects. In *Proc. ICFP*, pages 198–208, 2000.

[13] S. Feferman and R. L. Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.

[14] J. Ferrante and C. W. Rackoff. *The Computational Complexity of Logical Theories*, volume 718 of *Lecture Notes in Mathematics*. Springer-Verlag, 1979.

[15] T. Freeman and F. Pfenning. Refinement types for ML. In *Proc. ACM PLDI*, 1991.

[16] A. Frey. Satisfying subtype inequalities in polynomial space. *Theoretical Computer Science*, 277:105–117, 2002.

[17] N. Heintze and O. Tardieu. Ultra-fast aliasing analysis using CLA: A million lines of C code in a second. In *Proc. ACM PLDI*, 2001.

[18] F. Henglein and J. Rehof. The complexity of subtype entailment for simple types. In *Proc. 12th IEEE LICS*, pages 352–361, 1997.

[19] M. Hoand and J. C. Mitchell. Lower bounds on type inference with subtypes. In *Proc. 22rd ACM POPL*, pages 176–185, 1995.

[20] W. Hodges. *Model Theory*, volume 42 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1993.

[21] T. Jim and J. Palsberg. Type inference in systems of recursive types with subtyping. `http://www.cs.purdue.edu/homes/palsberg/`, 1999.

[22] M. Kerber and M. Kohlhase. A mechanization of strong Kleene logic for partial functions. In A. Bundy, editor, *Proc. 12th CADE*, pages 371–385, Nancy, France, 1994. Springer Verlag, Berlin, Germany. LNAI 814.

[23] S. C. Kleene. *Introduction to Metamathematics*. D. Van Nostrand Company, Inc., Princeton, New Jersey, 1952. fifth reprint, 1967.

[24] D. Kozen, J. Palsberg, and M. I. Schwartzbach. Efficient recursive subtyping. *Mathematical Structures in Computer Science*, 5(1):113–125, 1995.

[25] V. Kuncak and M. Rinard. On the theory of structural subtyping. Technical Report 879, Laboratory for Computer Science, Massachusetts Institute of Technology, `http://www.mit.edu/~vkuncak/papers/`, 2003.

[26] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd edition, 1987.

[27] M. J. Maher. Complete axiomatizations of the algebras of the finite, rational, and infinite trees. *Proc. 3rd IEEE LICS*, 1988.

[28] A. I. Mal'cev. *The Metamathematics of Algebraic Systems*, volume 66 of *Studies in Logic and The Foundations of Mathematics*. North Holland, 1971.

[29] J. C. Mitchell. Type inference with simple types. *Journal of Functional Programming*, 1(3):245–285, 1991.

[30] A. Mostowski. On direct products of theories. *Journal of Symbolic Logic*, 17(1):1–31, March 1952.

[31] M. Mueller and J. Niehren. Ordering constraints over feature trees expressed in second-order monadic logic. *Information and Computation*, 159(1/2):22–58, 2000.

[32] M. Mueller, J. Niehren, and R. Treinen. The first-order theory of ordering constraints over feature trees. *Discrete Mathematics and Theoretical Computer Science*, 4(2):193–234, September 2001.

[33] G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM (JACM)*, 27(2):356–364, 1980.

[34] J. Palsberg and P. M. O'Keefe. A type system equivalent to flow analysis. *Transactions on Programming Languages and Systems*, 17(4):576–599, July 1995.

[35] F. Pottier. Simplifying subtyping constraints: A theory. *Information and Computation*, 170(2):153–183, Nov. 2001.

[36] M. Presburger. über die vollständigkeit eines gewissen systems der aritmethik ganzer zahlen, in welchem die addition als einzige operation hervortritt. In *Comptes Rendus du premier Congrès des Mathématiciens des Pays slaves, Warsawa*, pages 92–101, 1929.

[37] W. C. Rounds. Feature logics. In J. v. Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier, 1997.

[38] T. Rybina and A. Voronkov. A decision procedure for term algebras with queues. *ACM Transactions on Computational Logic (TOCL)*, 2(2):155–181, 2001.

[39] V. Simonet. Type inference with structural subtyping: A faithful formalization of an efficient constraint solver. Submitted for publication, Mar. 2003.

[40] T. 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.

[41] T. Sturm and V. Weispfenning. Quantifier elimination in term algebras: The case of finite languages. In V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing (CASC)*, TUM Muenchen, 2002.

[42] Z. Su, A. Aiken, J. Niehren, T. Priesnitz, and R. Treinen. First-order theory of subtyping constraints. In *Proc. 29th ACM POPL*, 2002.

[43] A. Tarski. Arithmetical classes and types of algebraically closed and real-closed fields. *Bull. Amer. Math. Soc.*, 55, 64, 1192, 1949.

[44] A. Tarski. Arithmetical classes and types of boolean algebras. *Bull. Amer. Math. Soc.*, 55, 64, 1192, 1949.

[45] J. Tiuryn. Subtype inequalities. In *Proc. 7th IEEE LICS*, 1992.

[46] R. Treinen. Feature trees over arbitrary structures. In P. Blackburn and M. de Rijke, editors, *Specifying Syntactic Structures*, chapter 7, pages 185–211. CSLI Publications and FoLLI, 1997.

[47] V. Trifonov and S. Smith. Subtyping constrained types. In *Proc. 3rd International Static Analysis Symposium*, volume 1145 of *Lecture Notes in Computer Science*, 1996.

[48] I. Walukiewicz. Monadic second-order logic on tree-like structures. *Theoretical Computer Science*, 275(1–2):311–346, Mar. 2002.

[49] M. Wand, P. M. O'Keefe, and J. Palsberg. Strong normalization with non-structural subtyping. *Mathematical Structures in Computer Science*, 5(3):419–430, 1995.

Appendix: Transforming Quantifier Free Formulas to Structural Base Formulas

Rules are applied modulo associativity and commutativity of $\wedge, \vee$ and symmetry of equality $=$. $\bar{E}$ denotes a sequence of expressions $\langle E_i \rangle_i$. The result of substituting term t for variable x in formula C is denoted $C(x \mapsto t)$.

DNF: **Disjunctive Normal Form**

$$C[\neg(P \wedge Q)] \to C[\neg P \vee \neg Q]$$
$$C[\neg(P \vee Q)] \to C[\neg P \wedge \neg Q]$$
$$C[\neg\neg P] \to C[P]$$
$$C[P \wedge (Q \vee R)] \to C[(P \wedge Q) \vee (P \wedge R)]$$

WDNF: **Disjunction of Well-Defined Conjunctions**

$F \to \mathsf{DomCl}(F)$ where F in DNF
$\mathsf{DomCl}(\vee_i C_i) = \vee_i \mathsf{DomCl}(C_i)$
$\mathsf{DomCl}(\wedge_i L_i) = \wedge_i \mathsf{DomCl}(L_i)$
$\mathsf{DomCl}(R(\bar{t})) = R(\bar{t}) \wedge \mathsf{DefCl}(\bar{t})$
$\mathsf{DomCl}(\neg R(\bar{t})) = \neg R(\bar{t}) \wedge \mathsf{DefCl}(\bar{t})$
$\mathsf{DefCl}(\bar{t}) = \bigwedge\{ D_f(\bar{s}) \mid$
f a partial function symbol of arity n
D_f the relation specifying the domain of f
$f(\bar{s})$ a subterm occuring in $\bar{t}$ $\}$

UNF: **Unnested Form**

$C_1 \vee (\exists \bar{y}. C_2[f(\bar{x})]) \to C_1 \vee (\exists \bar{y} \exists z. z{=}f(\bar{x}) \wedge C_2[z])$ where
$C_2[f(\bar{x})]$ a conjunction of literals
occurence $C_2[\,]$ not in a literal of form $w = f(\bar{x})$
$C_1 \vee (\exists \bar{y}. C_2) \to C_1 \vee (\exists \bar{y} \exists u. u{=}x \wedge C_2(x \mapsto u))$ where
u a fresh variable
x a free variable s.t. C_2 contains no $u'{=}x$ for u' bound

ELNG: **Negative Literal Elimination**

$C[\neg \mathsf{ls}_f(y)] \to C[\mathsf{ls}_{\mathsf{PRI}}(y) \vee \bigvee\{ \mathsf{ls}_g(y) \mid g \in \Sigma \setminus \{f\} \}]$
$C[\neg \mathsf{ls}_{\mathsf{PRI}}(y)] \to C[\bigvee\{ \mathsf{ls}_g(y) \mid g \in \Sigma \}]$
$C[\neg \mathsf{ls}_{f^{\mathsf{s}}}(y^{\mathsf{s}})] \to C[\mathsf{ls}_{c^{\mathsf{s}}}(y^{\mathsf{s}}) \vee \bigvee\{ \mathsf{ls}_{g^{\mathsf{s}}}(y^{\mathsf{s}}) \mid g \in \Sigma \setminus \{f\} \}]$
$C[\neg \mathsf{ls}_{c^{\mathsf{s}}}(y^{\mathsf{s}})] \to C[\bigvee\{ \mathsf{ls}_{g^{\mathsf{s}}}(y^{\mathsf{s}}) \mid g \in \Sigma \}]$
$C[\neg |\phi|_{u^{\mathsf{s}}}{=}k] \to C[|\phi|_{u^{\mathsf{s}}}{\geq}k{+}1 \vee \bigvee_{i=0}^{k-1} |\phi|_{u^{\mathsf{s}}}{=}i]$
$C[\neg |\phi|_{u^{\mathsf{s}}}{\geq}k] \to C[\bigvee_{i=0}^{k-1} |\phi|_{u^{\mathsf{s}}}{=}i]$

SelEl: **Selector and Test Elimination**

$C_1 \vee (\exists \bar{y}^*. C_2 \wedge \mathsf{ls}_{f^*}(y^*)) \to$
$\quad C_1 \vee (\exists \bar{y}^* \exists \bar{z}^*. C_2 \wedge y^* = f^*(\bar{z}^*))$
$C_1 \vee (\exists \bar{y}^*. C_2 \wedge u^*{=}f^*(\langle u_i^* \rangle_i) \wedge v^*{=}f_j^*(u^*)) \to$
$\quad C_1 \vee (\exists \bar{y}^*. C_2 \wedge u^*{=}f^*(\langle u_i^* \rangle_i) \wedge v^*{=}u_j^*)$

ShpInt: **Shape Introduction**

$C_1 \vee (\exists \bar{u}^*. C_2) \to C_1 \vee (\exists \bar{u}^*, u^{\mathsf{s}}. \mathsf{sh}(u){=}u^{\mathsf{s}} \wedge C_2)$
u occurs in C_2
u^{s} fresh shape variable
C_2 contains no $\mathsf{sh}(u) = u^{\mathsf{s}'}$

CongCl: **Congruence Closure**

$C_1 \vee (\exists \bar{y}^* \exists u_1^* \exists u_2^*. u_1^*{=}u_2^* \wedge C_2) \to$
$\quad C_1 \vee (\exists \bar{y}^* \exists u_1^*. C_2(u_2^* \mapsto u_1^*))$
$C[u_1^*{=}f^*(\bar{z}^*) \wedge u_2^*{=}f^*(\bar{z}^*)] \to C[u_1^*{=}f^*(\bar{z}^*) \wedge u_2^*{=}u_1^*]$
$C[u^*{=}f^*(\bar{z}^*) \wedge u^*{=}g^*(\bar{x}^*)] \to C[\mathsf{false}], \quad f^* \not\equiv g^*$
$C[u^*{=}f^*(\bar{u}^*) \wedge u^*{=}f^*(\bar{v}^*)] \to$
$\quad C[u^*{=}f^*(\bar{u}^*) \wedge u^*{=}f^*(\bar{v}^*) \wedge \wedge_i u_i^*{=}v_i^*]$
$C[u^*{\neq}u^*] \to C[\mathsf{false}]$
$C[u^*{=}u^*] \to C[\mathsf{true}]$
$C[P \wedge \mathsf{false}] \to C[\mathsf{false}]$
$C[P \vee \mathsf{false}] \to C[P]$
$C[P \wedge \mathsf{true}] \to C[P]$
$C[P \vee \mathsf{true}] \to C[\mathsf{true}]$

OccChk: **Occur Check**

$C_1 \vee \beta \to C_1$ where
$\beta \equiv \exists \bar{u}. C_2$ for C_2 conjunction of literals
$u \rightarrowtail_\beta^+ u$ for some variable u

HomExp: **Homomorphism Property and Expansion**

$C[\mathsf{sh}(u){=}u_1^{\mathsf{s}} \wedge \mathsf{sh}(u){=}u_2^{\mathsf{s}}] \to$
$\quad C[\mathsf{sh}(u){=}u_1^{\mathsf{s}} \wedge \mathsf{sh}(u){=}u_2^{\mathsf{s}} \wedge u_1^{\mathsf{s}} = u_2^{\mathsf{s}}]$
$C_1 \vee (\exists \bar{y}^*. C_2 \wedge v{=}f(\bar{u}) \wedge \mathsf{sh}(v){=}v^{\mathsf{s}}) \to$
$\quad C_1 \vee (\exists \bar{y}^* \exists \bar{u}^{\mathsf{s}}. C_2 \wedge v{=}f(\bar{u}) \wedge \mathsf{sh}(v){=}v^{\mathsf{s}}$
$\qquad \wedge v^{\mathsf{s}}{=}f^{\mathsf{s}}(\bar{u}^{\mathsf{s}}) \wedge \wedge_i \mathsf{sh}(u_i) = u_i^{\mathsf{s}})$
$C_1 \vee (\exists \bar{y}^*. C_2 \wedge v^{\mathsf{s}}{=}f^{\mathsf{s}}(\bar{u}^{\mathsf{s}}) \wedge \mathsf{sh}(v){=}v^{\mathsf{s}}) \to$
$\quad C_1 \vee (\exists \bar{y}^* \exists \bar{u}. C_2 \wedge v^{\mathsf{s}}{=}f^{\mathsf{s}}(\bar{u}^{\mathsf{s}}) \wedge \mathsf{sh}(v){=}v^{\mathsf{s}}$
$\qquad \wedge v{=}f(\bar{u}) \wedge \wedge_i \mathsf{sh}(u_i) = u_i^{\mathsf{s}})$

NEQEl: **Term Disequality Elimination**

$C[u_1{\neq}u_2 \wedge \mathsf{sh}(u_1){=}u_1^{\mathsf{s}} \wedge \mathsf{sh}(u_2){=}u_2^{\mathsf{s}}] \to$
$\quad C[(u_1^{\mathsf{s}} \neq u_2^{\mathsf{s}} \vee (u_1^{\mathsf{s}} \neq u_2^{\mathsf{s}} \wedge |u_1 \neq u_2|_{u_1^{\mathsf{s}}} \geq 1)) \wedge$
$\qquad \mathsf{sh}(u_1){=}u_1^{\mathsf{s}} \wedge \mathsf{sh}(u_2){=}u_2^{\mathsf{s}}]$

CCD: **Cardinality Constraint Decomposition**

$C_1[|\phi(\langle f(\langle u_{ij} \rangle_j) \rangle_i)|_{u^{\mathsf{s}}}{=}k] \to$
$\quad C_1[\bigvee\{ \wedge_j |\phi(\langle u_{ij} \rangle_i)|_{u_j^{\mathsf{s}}}{=}k_j \mid \Sigma_j k_j = k \} \wedge C_2]$
$C_1[|\phi(\langle f(\langle u_{ij} \rangle_j) \rangle_i)|_{u^{\mathsf{s}}}{\geq}k] \to$
$\quad C_1[\bigvee\{ \wedge_j |\phi(\langle u_{ij} \rangle_i)|_{u_j^{\mathsf{s}}}{\geq}k_j \mid \Sigma_j k_j = k \} \wedge C_2]$
where C_2 contains
$u^{\mathsf{s}} = f(\langle u_j^{\mathsf{s}} \rangle_j) \wedge \bigwedge_{i,j} \mathsf{sh}(u_{ij}){=}u_j^{\mathsf{s}}$

ShDis: **Shape Distinction**

$C_1 \vee (\exists \bar{u}^*. C_2) \to C_1 \vee (\exists \bar{u}^*. (u_i^{\mathsf{s}} = u_j^{\mathsf{s}} \vee u_i^{\mathsf{s}} \neq u_j^{\mathsf{s}}) \wedge C_2)$