Expressing Polymorphic Types in a Many-Sorted Language

François Bobot, Andrei Paskevich

Expressing Polymorphic Types in a Many-Sorted Language

François Bobot[1,2] and Andrei Paskevich[1,2]

[1] LRI, Université Paris-Sud 11, CNRS, Orsay F-91405
[2] INRIA Saclay-Île de France, ProVal, Orsay F-91893

Abstract. In this paper, we study translation from a first-order logic with polymorphic types à la ML (of which we give a formal description) to a many-sorted or one-sorted logic as accepted by mainstream automated theorem provers. We consider a three-stage scheme where the last stage eliminates polymorphic types while adding the necessary "annotations" to preserve soundness, and the first two stages serve to protect certain terms so that they can keep their original unannotated form. This protection allows us to make use of provers' built-in theories and operations. We present two existing translation procedures as sound and complete instances of this generic scheme. Our formulation generalizes over the previous ones by allowing to protect terms of arbitrary monomorphic types. In particular, we can benefit from the built-in theory of arrays in SMT solvers such as Z3, CVC3, and Yices. The proposed methods are implemented in the Why3 tool and we compare their performance in combination with several automated provers.

1 Introduction

Polymorphic types are a means of abstraction over families of different types; a polymorphic definition or proposition stands for a potentially infinite number of its type-specific instances. Type systems employing polymorphism arise naturally in programming languages and they are a prominent feature of interactive proof assistants such as Coq [17] or Isabelle/HOL [15].

However, a proof task written in a language with polymorphic types is today a difficult subject for automation. This is not because polymorphism handling in a prover is complicated or inefficient *per se*. As was demonstrated by the Alt-Ergo project [3], this only requires a straightforward extension of the unification procedure and does not impose any significant overhead. The fact is, advanced type systems have not yet become mainstream in automated deduction: SMT solvers use many-sorted languages such as SMT-LIB [1], and TPTP provers are content with one-sorted first-order language. Thus, to apply a mainstream prover to a problem expressed in a polymorphic language, we have to translate it into an equivalent monomorphic or even one-sorted problem.

The challenge is not new and a number of solutions is known, ranging from adding per-variable "type guards" (also known as "relativisation of quantifiers", see [11] and [10, Sect. 3.0]), to throughout decoration of terms with their types [8, 5], to various flavours of type erasure [9, 12, 10]. The latter method is logically

unsound, though adding type annotations can prevent certain unsound inference steps (see [12, Sect. 2.5,2.6] and [10, Sect. 3.1]).

An important feature of a polymorphism encoding method is special treatment of types and operations that are directly handled by provers' built-in decision procedures, e.g., for linear arithmetic or bit-vectors. The idea is to prevent the terms that can be interpreted by a prover from being modified by translation, to preserve their original form [5, 10]. In what follows, we call this "type protection" to emphasize that we are interested in terms of particular types.

In this work, we aim to lift (or at least work around) several limitations we perceive in the previous approaches. Firstly, the existing type protection techniques only handle "simple" types, like integers or booleans, but not instances of polymorphic types, like lists of integers or arrays of reals. Yet decision procedures for such "complex" types are implemented in some SMT solvers; for example, Z3 [13], CVC3 [2], and Yices [6] have a built-in support for arrays. Secondly, type protection, as defined in [5], cannot be used to protect finite types such as booleans: given an axiom "every boolean is equal either to 'true' or to 'false'", one can derive that there are only two values in any encoded type, which can easily lead to a contradiction. Thirdly, while translation by type erasure with addition of type arguments to polymorphic symbols [10, Sect. 3.1] is less intrusive and more efficient than full term decoration [5], the former method is unsound and, according to [10], is only applicable in combination with provers that use trigger-based rather than unification-based instantiation. Such a requirement excludes the superposition-based provers and may be difficult to test when a third-party prover is used.

We begin with a formal presentation of first-order logic with polymorphic types (Section 2). In particular, we show that complete monomorphisation is undecidable, that is, we cannot effectively compute a finite set of monomorphic instances of a polymorphic formula F that is equisatisfiable to F. Then we introduce a generic three-stage scheme of polymorphism encoding (Section 3). In this scheme, we start by replacing interpreted polymorphic symbols (such as operations of access and update in arrays) with selected monomorphic instances. The translation proceeds then to type protection, which we consider as a separate transformation, and concludes with polymorphism elimination proper.

We present a sound and complete method of type instantiation with symbol discrimination that we use (in slightly different variants) both for the first and the second stage (Sections 3.1 and 3.2). Furthermore, we give a generalized formulation of the type protection method from [5], free from the aforesaid restrictions (Section 3.3). As third-stage transformations, we consider full term decoration from [8, 5] (Section 3.4) and type erasure with added type annotations from [10, Sect. 3.1] (Section 3.5). We show the latter method to be sound on problems that admit models with infinite domains for every non-protected type and we discuss how this condition can be handled in practice.

We conclude by comparing the described techniques in combination with the SMT solvers Z3, CVC3, and Yices [6] on a set of about 4100 proof obligations in the Why3 tool [4] (Section 4).

2 First-Order Logic With Polymorphic Types

The logic $\mathbf{FOL}_\mathrm{T}$, presented below, is an extension of classical first-order many-sorted logic. In $\mathbf{FOL}_\mathrm{T}$, types are built from type constants (such as "integer"), type functions (such as "list of"), and type variables that stand for arbitrary monomorphic types. We do not admit quantifiers over type variables, neither in types, nor in formulas: a polymorphic formula is rather seen as a scheme, a potentially infinite conjunction of its monomorphic type instances. In other words, every type variable that occurs in a formula is bound by an implicit prenex universal quantifier. Basically, we use type polymorphism as a convenient way to write once a set of polymorphic axioms for, say, lists or arrays, instead of copying them for every particular instance of these types.

The principal purpose of $\mathbf{FOL}_\mathrm{T}$ is to help specify and prove programs and its type system can be seen as the first-order fragment of the ML type system. The Why3 verification tool [4] is based upon $\mathbf{FOL}_\mathrm{T}$ with some extensions such as algebraic types. The papers [5] and [10] work in a similar setting, though the latter employs explicit quantifiers over type variables in logic formulas.

Syntax. We define types as syntactical expressions built from *type constructors* of fixed arity (denoted with capital sans-serif letters) and *type variables* (denoted α, β, γ). For example, β, I, $\mathsf{F}(\mathsf{I}, \gamma)$ are well-formed types. Type constructors of arity 0 are called *type constants*. A type that contains no type variables is called *monomorphic type*, *closed type*, or *sort*. A vector of types $\langle T_1, \ldots, T_n \rangle$ is called *type signature*. We define *type substitution* and *type matching* in the same way as they are usually defined for terms. A *monomorphic* type substitution instantiates every type variable either to itself or to a closed type.

We use letters S and T for types, boldface letters $\mathbf{S}$, $\mathbf{T}$ for type signatures, and Greek letters τ, θ, and π for type substitutions. We denote the set of available type variables with $\mathbb{V}_\mathrm{T}$, the set of type constructors with $\mathbb{F}_\mathrm{T}$, the set of all types built from $\mathbb{V}_\mathrm{T}$ and $\mathbb{F}_\mathrm{T}$ with $\mathcal{T}(\mathbb{F}_\mathrm{T}, \mathbb{V}_\mathrm{T})$, and the set of all sorts with $\mathcal{T}(\mathbb{F}_\mathrm{T})$. We presume that $\mathbb{V}_\mathrm{T}$ is infinite and $\mathbb{F}_\mathrm{T}$ contains at least one type constant.

We use traditional first-order terms and formulas, built from variable symbols (denoted u, v, w), function symbols (denoted f, g, h), and predicate symbols (denoted p, q), with the following additions:

- Every term, be it a variable or a function application, carries a type, e.g.: $w : \mathsf{C}(\mathsf{I})$, $w : \mathsf{C}(\alpha)$, $f(u : \alpha, v : \mathsf{L}(\alpha)) : \mathsf{L}(\alpha)$. The first two terms are considered as different variables even though they share the variable symbol. We denote terms with letters s and t, and, by abuse of notation, we sometimes write the type of a term to the right of the letter: $s : T_1$, $t : T_2$, and so on.
- To each function symbol of arity n we assign a type signature of length $n+1$. A term of the form $f(t_1 : T_1, \ldots, t_n : T_n) : T$ is well-formed if and only if the type signature of f matches $\langle T_1, \ldots, T_n, T \rangle$.
- To each predicate symbol of arity n we give a type signature of length n. An atomic formula of the form $p(t_1 : T_1, \ldots, t_n : T_n)$ is well-formed if and only if the type signature of p matches $\langle T_1, \ldots, T_n \rangle$.

– An atomic equality formula of the form $t_1 \approx t_2$ is well-formed if and only if the terms t_1, t_2 have the same type.
– Quantifiers bind variables, i.e., typed variable symbols: $\forall(u:\alpha)\, p(u:\alpha, u:\mathsf{C})$. Here, the first argument of p is bound, but the second one is free.

We treat equality ($\approx$), negation ($\neg$), conjunction ($\wedge$), and the universal quantifier ($\forall$) as logical symbols and we treat disjunction ($\vee$), implication ($\supset$), equivalence ($\equiv$), disequality ($\not\approx$), and the existential quantifier ($\exists$) as abbreviations.

We use letters x, y, z for variables, letters F, G, H for formulas, and Greek letters Γ, Δ for sets of formulas. We denote the (infinite) set of variable symbols with $\mathbb{V}$, the set of function symbols with $\mathbb{F}$, and the set of predicate symbols with $\mathbb{P}$. Given a term or a formula e, the set of type variables occurring in e is denoted $\mathrm{FV}_\mathrm{T}(e)$ and the set of free variables of e is denoted $\mathrm{FV}(e)$. If $\mathrm{FV}_\mathrm{T}(e)$ is empty, we call e *monomorphic*. If $\mathrm{FV}(e)$ is empty, we call e *closed* or *ground*.

Substitutions apply to terms and formulas, replacing free term variables with terms of the same type. We denote substitutions with letters σ and δ. The symbol $\circ$ denotes the composition of two (type) substitutions: $x(\sigma \circ \delta) \triangleq x\sigma\delta$.

Type substitutions apply only to closed formulas and ground terms; also, we require type instantiation to rename every bound variable symbol to some fresh variable symbol. In this way, we avoid variable collisions: for example, the type substitution $[\mathsf{I}/\alpha]$ would not instantiate the formula $\forall(u:\alpha)\forall(u:\mathsf{I})\, p(u:\alpha, u:\mathsf{I})$ to $\forall(u:\mathsf{I})\forall(u:\mathsf{I})\, p(u:\mathsf{I}, u:\mathsf{I})$, but to $\forall(u':\mathsf{I})\forall(u'':\mathsf{I})\, p(u':\mathsf{I}, u'':\mathsf{I})$. In our future examples, we will not use a variable symbol in two different variables in the same formula to avoid confusion.

Throughout the rest of the paper, we illustrate our transformations on the following simple polymorphic formula (for the sake of readability, we omit the most obvious type annotations):

$$\forall(m:\mathsf{M}(\alpha,\mathsf{I}))\,\forall(c:\alpha)\,\mathtt{get}(\mathtt{set}(m,c,\mathsf{6}):\mathsf{M}(\alpha,\mathsf{I}),c):\mathsf{I}*\mathsf{7}\approx\mathsf{42}$$

Here, the type I represents integers and the type $\mathsf{M}(\alpha,\beta)$ is that of polymorphic α-to-β maps. The function symbol `get` is of type signature $\langle\mathsf{M}(\alpha,\beta),\alpha,\beta\rangle$ and `set` is of type signature $\langle\mathsf{M}(\alpha,\beta),\alpha,\beta,\mathsf{M}(\alpha,\beta)\rangle$.

Satisfiability. Given sets $\mathbb{F}_\mathrm{T}$, $\mathbb{F}$, $\mathbb{P}$, an interpretation $\mathfrak{I}$ is defined by three maps:

– to each sort $S \in \mathcal{T}(\mathbb{F}_\mathrm{T})$, we assign a non-empty *domain* $\mathcal{D}^\mathfrak{I}_S$;
– to each symbol $f \in \mathbb{F}$ and each vector of sorts $\mathbf{S} = \langle S_1, \ldots, S_n, S\rangle$ matched by the type signature of f, we assign a function $f^\mathfrak{I}_\mathbf{S} : \mathcal{D}^\mathfrak{I}_{S_1} \times \cdots \times \mathcal{D}^\mathfrak{I}_{S_n} \to \mathcal{D}^\mathfrak{I}_S$;
– to each symbol $p \in \mathbb{P}$ and each vector of sorts $\mathbf{S} = \langle S_1, \ldots, S_n\rangle$ matched by the type signature of p, we assign a function $p^\mathfrak{I}_\mathbf{S} : \mathcal{D}^\mathfrak{I}_{S_1} \times \cdots \times \mathcal{D}^\mathfrak{I}_{S_n} \to \{\top, \bot\}$, where $\top$ and $\bot$ stand for Boolean constants "true" and "false", respectively.

We call *type valuation* a type substitution that instantiates every type variable in $\mathbb{V}_\mathrm{T}$ to a sort. Given a type valuation π, we call *variable valuation under* π a function that maps every variable $u:T$ to some element of $\mathcal{D}^\mathfrak{I}_{T\pi}$. We simply say *variable valuation* when the implied type valuation is known from the context or in a purely monomorphic setting, where every type is already closed.

Let π be a type valuation and φ be a variable valuation under π. We evaluate terms and formulas according to the following equalities:

$$\begin{aligned}
\mathfrak{I}_{\pi,\varphi}(u:T) &\triangleq \varphi(u:T) & \mathfrak{I}_{\pi,\varphi}(t_1 \approx t_2) &\triangleq (\mathfrak{I}_{\pi,\varphi}(t_1) = \mathfrak{I}_{\pi,\varphi}(t_2)) \\
\mathfrak{I}_{\pi,\varphi}(f(\mathbf{t}:\mathbf{T})):T) &\triangleq f^{\mathfrak{I}}_{\langle \mathbf{T},T\rangle\pi}(\mathfrak{I}_{\pi,\varphi}(\mathbf{t})) & \mathfrak{I}_{\pi,\varphi}(\neg F) &\triangleq \neg\, \mathfrak{I}_{\pi,\varphi}(F) \\
\mathfrak{I}_{\pi,\varphi}(p(\mathbf{t}:\mathbf{T})) &\triangleq p^{\mathfrak{I}}_{\mathbf{T}\pi}(\mathfrak{I}_{\pi,\varphi}(\mathbf{t})) & \mathfrak{I}_{\pi,\varphi}(F \wedge G) &\triangleq \mathfrak{I}_{\pi,\varphi}(F) \wedge \mathfrak{I}_{\pi,\varphi}(G) \\
& & \mathfrak{I}_{\pi,\varphi}(\forall(u:T)F) &\triangleq \bigwedge_{a \in \mathcal{D}^{\mathfrak{I}}_{T\pi}} \mathfrak{I}_{\pi,\varphi[u:T \mapsto a]}(F)
\end{aligned}$$

where $\mathbf{t}:\mathbf{T}$ stands for a vector of terms $t_1:T_1, \ldots, t_n:T_n$ and $\varphi[u:T \mapsto a]$ is a valuation that coincides with φ everywhere except $u:T$, which is mapped to a.

It is easy to see that evaluation of a term or a formula e under $\mathfrak{I}_{\pi,\varphi}$ does not depend on the values of π and φ on (type) variables that do not occur in e.

Lemma 1. *Let F be a formula and $\mathfrak{I}$ be an interpretation. Let π and τ be type valuations that coincide on every type variable in $\mathrm{FV_T}(F)$. Let φ be a variable valuation under π and ψ be a variable valuation under τ. Let us suppose that $\varphi(u:T) = \psi(u:T)$ for every variable $u:T$ in $\mathrm{FV}(F)$. Then $\mathfrak{I}_{\pi,\varphi}(F) = \mathfrak{I}_{\tau,\psi}(F)$.*

In what follows, when we evaluate closed or monomorphic expressions, we often omit the variable valuation or the type valuation, respectively.

Lemma 2. *Let F be a closed formula and $\mathfrak{I}$ be an interpretation. For any type valuation π, $\mathfrak{I}_\pi(F) = \mathfrak{I}(F\pi)$.*

As we said above, we treat type variables as implicitly universally quantified at the top of a polymorphic formula. Thus, a closed formula F is *satisfied* by $\mathfrak{I}$ if and only if $\mathfrak{I}_\pi(F)$ is true for every type valuation π. A closed formula is *satisfiable* if and only if it is satisfied by some interpretation, called a *model* of this formula. These definitions are trivially extended to sets of closed formulas.

It is easy to see that on monomorphic terms and formulas our definitions correspond to the traditional many-sorted logic with disjoint sorts. Moreover, by Lemma 2, we can see closed polymorphic formulas as mere "schemes" of their monomorphic type instances (of which there may be an infinite number). To put this precisely, we denote the set of all monomorphic type instances of a closed formula F with $\mathrm{MI}(F)$. Similarly, given a set of closed formulas Γ, the set $\mathrm{MI}(\Gamma)$ contains all monomorphic type instances of formulas in Γ. The trivial corollary of Lemma 2 is that Γ is satisfiable in $\mathbf{FOL}_\mathrm{T}$ if and only if $\mathrm{MI}(\Gamma)$ is so.

Computing monomorphic instances? Of course, as soon as a closed formula F is non-monomorphic and there is at least one non-constant type constructor, F has infinitely many monomorphic type instances. But can't we find out, in finite time, all sorts that are potentially relevant to the problem in question, and deal with a finite subset of instances, produced just with these sorts?

On one hand, this resembles an attempt to pre-compute the relevant ground instances in a set of first-order formulas — a problem well known to be undecidable. On the other hand, type handling does not need to be as hard as proof

search in general, and complete monomorphisation is often possible in programming languages (e.g., C++ templates).

Theorem 1. *There is no computable function that maps an arbitrary closed formula F to an equisatisfiable finite set of monomorphic type instances of F (notice that such a set always exists by compactness).*

Proof. It turns out that our type system is expressive enough to encode an undecidable theory, namely, combinatory logic. Consider the following signature:

$$\mathbb{F}_{\mathrm{T}} = \{\, \mathsf{A}(\cdot,\cdot),\ \mathsf{S},\ \mathsf{K} \,\} \quad \mathbb{F} = \{\, \mathtt{A} : \langle \alpha, \beta, \mathsf{A}(\alpha,\beta) \rangle,\ \mathtt{S} : \langle \mathsf{S} \rangle,\ \mathtt{K} : \langle \mathsf{K} \rangle \,\} \quad \mathbb{P} = \{\, \mathtt{R} : \langle \alpha, \beta \rangle \,\}$$

along with five axioms (for brevity, we omit some type annotations):

$$\forall(u:\alpha)\,\forall(v:\beta)\,\forall(w:\gamma)\,((\mathtt{R}(u,v) \wedge \mathtt{R}(v,w)) \supset \mathtt{R}(u,w))$$

$$\forall(u:\alpha)\,\forall(v:\beta)\,\forall(w:\gamma)\,(\mathtt{R}(u,v) \supset \mathtt{R}(\mathtt{A}(u,w) : \mathsf{A}(\alpha,\gamma), \mathtt{A}(v,w) : \mathsf{A}(\beta,\gamma)))$$

$$\forall(u:\alpha)\,\forall(v:\beta)\,\forall(w:\gamma)\,(\mathtt{R}(u,v) \supset \mathtt{R}(\mathtt{A}(w,u) : \mathsf{A}(\gamma,\alpha), \mathtt{A}(w,v) : \mathsf{A}(\gamma,\beta)))$$

$$\forall(u:\alpha)\,\forall(v:\beta)\,\mathtt{R}(\mathtt{A}(\mathtt{A}(\mathtt{K},u),v) : \mathsf{A}(\mathsf{A}(\mathsf{K},\alpha),\beta),\ u:\alpha)$$

$$\forall(u:\alpha)\,\forall(v:\beta)\,\forall(w:\gamma)\,\mathtt{R}(\mathtt{A}(\mathtt{A}(\mathtt{A}(\mathtt{S},u),v),w) : \mathsf{A}(\mathsf{A}(\mathsf{A}(\mathsf{S},\alpha),\beta),\gamma),\ \mathtt{A}(\mathtt{A}(u,w),\mathtt{A}(v,w)) : \mathsf{A}(\mathsf{A}(\alpha,\gamma),\mathsf{A}(\beta,\gamma)))$$

Here the binary function symbol $\mathtt{A}$ stands for term application, and the binary predicate symbol $\mathtt{R}$ for CL-reducibility. Notice that every ground combinatory term is reflected in its type.

Now, if we were able to compute a finite set of potentially relevant *closed types* for an arbitrary reducibility problem in this theory, this would readily let us decide the problem itself, as we would thus obtain the set of potentially relevant *ground terms*. Since ground reducibility in CL is undecidable, complete monomorphisation in $\mathbf{FOL}_{\mathrm{T}}$ is undecidable, too. □

3 Eliminating Polymorphic Types

Being unable to select just a relevant monomorphic subset of a polymorphic problem, we have to resort to some form of encoding, converting the polymorphic problem to an equisatisfiable monomorphic one. Such conversion inevitably implies merging many types into few sorts or just a single sort. This is undesirable if we target an automated prover equipped with special techniques (decision procedures, unification modulo, etc.) for particular types, such as integers, booleans or arrays. These types ought to be separated from the rest, protected against this "type fusion", expelled from polymorphism in the problem.

To this purpose, we slightly extend our language in order to be able to select the terms that will keep their (monomorphic) type through polymorphism elimination. To every sort S in $\mathcal{T}(\mathbb{F}_{\mathrm{T}})$ we associate a new *protected sort* $\bar{S}$. The use of protected sorts is restricted: a protected sort can appear in the type signature of a symbol or as a type of a term, but it cannot occur under a type constructor or

in the range of a type substitution. In other words, the only type that matches a protected sort $\bar{S}$ is $\bar{S}$ itself.

For example, $\mathtt{get}(v : \overline{\mathsf{M}(\mathsf{I},\mathsf{I})}, c : \mathsf{I}) : \mathsf{I}$ is a malformed term, since the type signature of $\mathtt{get}$ is $\langle \mathsf{M}(\alpha,\beta), \alpha, \beta\rangle$ and $\mathsf{M}(\alpha,\beta)$ does not match $\overline{\mathsf{M}(\mathsf{I},\mathsf{I})}$. Similarly, the term $\mathtt{get}(v : \mathsf{M}(\mathsf{I},\bar{\mathsf{I}}), c : \mathsf{I}) : \bar{\mathsf{I}}$ is malformed, because β does not match $\bar{\mathsf{I}}$ and also because $\mathsf{M}(\mathsf{I},\bar{\mathsf{I}})$ is a malformed type expression. However, if we consider a "protected specialization" of $\mathtt{get}$, denoted $\overline{\mathtt{get}}$, with the type signature $\langle \overline{\mathsf{M}(\mathsf{I},\mathsf{I})}, \bar{\mathsf{I}}, \bar{\mathsf{I}}\rangle$, the application $\overline{\mathtt{get}}(v : \overline{\mathsf{M}(\mathsf{I},\mathsf{I})}, c : \bar{\mathsf{I}}) : \bar{\mathsf{I}}$ is a well-formed term.

Concerning interpretation, every protected sort $\bar{S}$ has its proper non-empty domain $\mathcal{D}^{\mathfrak{I}}_{\bar{S}}$. As with any type substitution, we restrict type valuations to non-protected sorts. Thus, a set $\{\forall(u:\alpha)\,\forall(v:\alpha)\,u \approx v,\ \exists(a:\bar{\mathsf{I}})\,\exists(b:\bar{\mathsf{I}})\,a \not\approx b\}$ is satisfiable. Indeed, the first formula requires the domain of every non-protected sort to be a singleton, but does not constrain the domains of protected sorts.

Using protected sorts, we can define a general three-stage scheme of encoding of polymorphic formulas, explained below from the end to the beginning.

The final, "type-fusing" stage takes a set of polymorphic formulas with protected sorts and converts it into an equisatisfiable set of monomorphic formulas. A common requirement to the methods on this stage is preservation of terms with protected types: monomorphic protected fragments of the problem, e.g., arithmetic expressions, must be sent to a prover as is. We present two "type-fusing" transformations, Dec and Exp, in Sections 3.4 and 3.5. Both methods have been previously described in the literature [8, 9, 12, 5, 10]. Our presentation is more general in that it permits to protect arbitrarily complex monomorphic types, such as "list of integers" or "integer-to-real map". The ability to preserve such sorts is of more than purely theoretical interest: as we have already mentioned, Z3, CVC3, and Yices provide built-in support for access and update operations on integer-indexed arrays.

The intermediate, "type-protecting" stage takes a set of polymorphic formulas without protected sorts and converts it into an equisatisfiable set of formulas with protection. The methods on this stage take as a parameter the set of sorts that we wish to protect; we expect them to put protection over every occurrence of every sort from this set in the problem. In this paper, we present two type-protecting transformations, Par and Tw, in Sections 3.2 and 3.3. The latter method was introduced in [5]; we generalize it to complex sorts.

The first stage can be figuratively called "type-revealing". Even if our type-protecting and type-fusing transformations are not limited to sort constants and can protect arbitrarily complex sorts, say, arrays of integers, we cannot readily benefit from this capacity. In an initial $\mathbf{FOL}_{\mathrm{T}}$-problem, arrays are most probably formalized as a polymorphic type, with premises that apply to arrays of any type and with polymorphic function symbols for access and update. In order to produce interpreted monomorphic operations for Z3, CVC3, or Yices in the end, we must start by replacing, wherever possible, these function symbols with their monomorphic specializations. This is the purpose of the Dis transformation, presented in Section 3.1. We show in Section 4 that this "type revealing" brings a considerable improvement to provers' results.

3.1 Symbol Discrimination

The Dis transformation involves producing a sufficient number of type instances of formulas in an initial problem Γ with subsequent discrimination of function and predicate symbols.

Let f be a function symbol of type signature $\mathbf{S}$ in the initial problem Γ. Let τ be a type substitution in the type variables of $\mathbf{S}$. A fresh function symbol f_τ with the type signature $\mathbf{S}\tau$ is called a *specialization* of f. We call f_τ a *monomorphic specialization* if $\mathbf{S}\tau$ is monomorphic. Specializations of predicate symbols are defined in the same way.

Let W be a set of monomorphic specializations of function and predicate symbols in Γ. These are the instances that we want to "reveal" in the problem. The set W is fixed for the rest of this section; the Dis transformation is implicitly parametrized by it.

First of all, the Dis transformation modifies the signature of Γ:

1. For every variable symbol u and type T, we add a new variable symbol u_T.
2. We add every function and predicate symbol from W.

Given an arbitrary type substitution θ, the discriminating transformation $\textsc{Dis}_\theta$ instantiates and converts terms and formulas into the new signature:

1. Given a variable $u:T$, $\textsc{Dis}_\theta(u:T) \triangleq u_T:T\theta$.
2. Consider a term $t = f(t_1:T_1,\ldots,t_n:T_n):T$. Let τ be the type substitution that instantiates the type signature of the symbol f to $\langle T_1\theta,\ldots,T_n\theta,T\theta\rangle$. If f_τ belongs to W, then $\textsc{Dis}_\theta(t) \triangleq f_\tau(\textsc{Dis}_\theta(t_1):T_1\theta,\ldots,\textsc{Dis}_\theta(t_n):T_n\theta):T\theta$. Otherwise, $\textsc{Dis}_\theta(t) \triangleq f(\textsc{Dis}_\theta(t_1):T_1\theta,\ldots,\textsc{Dis}_\theta(t_n):T_n\theta):T\theta$.
3. Consider an atomic formula $F = p(t_1:T_1,\ldots,t_n:T_n)$. Let τ be the type substitution that instantiates the type signature of p to $\langle T_1\theta,\ldots,T_n\theta\rangle$. If p_τ is in W, then $\textsc{Dis}_\theta(F) \triangleq p_\tau(\textsc{Dis}_\theta(t_1):T_1\theta,\ldots,\textsc{Dis}_\theta(t_n):T_n\theta)$. Otherwise, $\textsc{Dis}_\theta(F) \triangleq p(\textsc{Dis}_\theta(t_1):T_1\theta,\ldots,\textsc{Dis}_\theta(t_n):T_n\theta)$.

Equalities and complex formulas are converted in a natural way:

$$\textsc{Dis}_\theta(t_1 \approx t_2) \triangleq \textsc{Dis}_\theta(t_1) \approx \textsc{Dis}_\theta(t_2) \qquad \textsc{Dis}_\theta(F \wedge G) \triangleq \textsc{Dis}_\theta(F) \wedge \textsc{Dis}_\theta(G)$$
$$\textsc{Dis}_\theta(\neg F) \triangleq \neg\,\textsc{Dis}_\theta(F) \qquad \textsc{Dis}_\theta(\forall x\,F) \triangleq \forall(\textsc{Dis}_\theta(x))\,\textsc{Dis}_\theta(F)$$

Now, let F be a closed formula. The set of monomorphic type substitutions $\Theta(F)$ is defined as follows:

$$\begin{aligned}\Theta(F) \triangleq \{\,\theta \mid\ & F \text{ contains a term } f(t_1:T_1,\ldots,t_n:T_n):T \text{ such that}\\ & \theta \text{ only instantiates the variables of } \mathbf{T} = \langle T_1,\ldots,T_n,T\rangle \text{ and}\\ & W \text{ contains a specialization of } f \text{ with the type signature } \mathbf{T}\theta\,\}\\ \cup\,\{\,\theta \mid\ & F \text{ contains an atomic formula } p(t_1:T_1,\ldots,t_n:T_n) \text{ such that}\\ & \theta \text{ only instantiates the variables of } \mathbf{T} = \langle T_1,\ldots,T_n\rangle \text{ and}\\ & W \text{ contains a specialization of } p \text{ with the type signature } \mathbf{T}\theta\,\}\end{aligned}$$

We call two monomorphic type substitutions *compatible* if they do not substitute two different sorts for the same type variable. The *union* of two compatible monomorphic type substitutions is their composition (the order is irrelevant). We define $\Theta^\star(F)$ as the closure of $\Theta(F)$ with respect to finite unions of compatible type substitutions. The empty union, i.e., the identity type substitution, also belongs to $\Theta^\star(F)$.

Finally, Dis translates a closed formula F into a set of formulas:

$$\textsc{Dis}(F) \triangleq \{\, \textsc{Dis}_\theta(F) \mid \theta \in \Theta^\star(F) \,\}$$

On our running example, assuming $W = \{\mathtt{get}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}, \mathtt{set}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}\}$, Dis produces the following two formulas:

$$\forall(m_{\mathsf{M}(\alpha,\mathsf{I})} : \mathsf{M}(\alpha,\mathsf{I}))\, \forall(c_\alpha : \alpha)\, \mathtt{get}(\mathtt{set}(m_{\mathsf{M}(\alpha,\mathsf{I})}, c_\alpha, 6), c_\alpha) * 7 \approx 42$$
$$\forall(m_{\mathsf{M}(\alpha,\mathsf{I})} : \mathsf{M}(\mathsf{I},\mathsf{I}))\, \forall(c_\alpha : \mathsf{I})\, \mathtt{get}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}(\mathtt{set}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}(m_{\mathsf{M}(\alpha,\mathsf{I})}, c_\alpha, 6), c_\alpha) * 7 \approx 42$$

The new symbol $\mathtt{get}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}$ has the monomorphic type signature $\langle \mathsf{M}(\mathsf{I},\mathsf{I}), \mathsf{I}, \mathsf{I}\rangle$.

Lemma 3. *Let θ be a type substitution, t a term of type T, and F a formula. Then $\textsc{Dis}_\theta(t)$ is a well-formed term of type $T\theta$ and $\textsc{Dis}_\theta(F)$ is a well-formed formula such that $\mathrm{FV}(\textsc{Dis}_\theta(F)) = \{\textsc{Dis}_\theta(x) \mid x \in \mathrm{FV}(F)\}$ and $\mathrm{FV}_\mathrm{T}(\textsc{Dis}_\theta(F)) = \mathrm{FV}_\mathrm{T}(F)\backslash\mathrm{dom}(\theta)$.*

Proof. The proof goes by induction on the structure of terms and formulas. The only interesting case is the application of a function symbol (predicate symbols are treated in the same way). Let t be a term of the form $f(t_1 : T_1, \ldots, t_n : T_n) : T$. Let τ be the type substitution that instantiates the type signature of f to $\langle T_1\theta, \ldots, T_n\theta, T\theta\rangle$. For every argument t_i, $\textsc{Dis}_\theta(t_i)$ is a well-formed term of type $T_i\theta$ by the induction hypothesis. If f_τ belongs to W, then its type signature is exactly $\langle T_1\theta, \ldots, T_n\theta, T\theta\rangle$. Otherwise, as τ witnesses, the type signature of f matches $\langle T_1\theta, \ldots, T_n\theta, T\theta\rangle$. □

Theorem 2 (Soundness of Dis). *If a set of closed formulas Γ is satisfiable then $\textsc{Dis}(\Gamma)$ is satisfiable.*

Proof. Let $\mathfrak{M}$ be a model of Γ. We are going to construct an interpretation $\mathfrak{I}$ of $\textsc{Dis}(\Gamma)$ such that, for every type substitution θ, type valuation π, and variable valuation φ in $\mathfrak{I}$, there exist a type valuation π' and a variable valuation φ' in the original model $\mathfrak{M}$, such that:

$$\mathfrak{I}_{\pi,\varphi}(\textsc{Dis}_\theta(t)) = \mathfrak{M}_{\pi',\varphi'}(t) \qquad\qquad \mathfrak{I}_{\pi,\varphi}(\textsc{Dis}_\theta(F)) = \mathfrak{M}_{\pi',\varphi'}(F)$$

for every term t and formula F occurring in Γ. Since for every closed $F \in \Gamma$ and every $\theta \in \Theta^\star(F)$, $\mathfrak{I}$ satisfies $\textsc{Dis}_\theta(F)$, $\mathfrak{I}$ is a model of $\textsc{Dis}(\Gamma)$.

For any sort S in the new signature, we define the domain $\mathcal{D}^{\mathfrak{I}}_S \triangleq \mathcal{D}^{\mathfrak{M}}_S$. Moreover, every function and predicate symbol in Γ is interpreted in $\mathfrak{I}$ exactly as in $\mathfrak{M}$. Every new function symbol f_τ with the type signature $\mathbf{S}$ in $\textsc{Dis}(\Gamma)$ is

interpreted as f on $\mathbf{S}$ in Γ: $(f_\tau)^{\mathfrak{I}}_{\mathbf{S}} \triangleq f^{\mathfrak{M}}_{\mathbf{S}}$. New predicate symbols are interpreted in the same way.

Now, let us consider some type valuation π, a variable valuation φ in $\mathfrak{I}$, and a type substitution θ. We set π' to be $\theta \circ \pi$ and we define $\varphi'(u:T) \triangleq \varphi(u_T : T\theta)$. Notice that for any type T in Γ, $\mathcal{D}^{\mathfrak{I}}_{T\theta\pi} = \mathcal{D}^{\mathfrak{M}}_{T\theta\pi} = \mathcal{D}^{\mathfrak{M}}_{T\pi'}$

Using this definitions, we proceed to prove the two equalities above. The proof goes by induction on the structure of terms and formulas. We consider three cases: a variable, a function application, and a universally quantified formula (other cases can be handled along the same lines).

1. Consider a variable $u:T$ occurring in Γ. We obtain:
$$\mathfrak{I}_{\pi,\varphi}(\textsc{Dis}_\theta(u:T)) = \varphi(u_T : T\theta) = \varphi'(u:T) = \mathfrak{M}_{\pi',\varphi'}(u:T)$$
2. Consider a term $t = f(t_1 : T_1, \ldots, t_n : T_n) : T$ and let $\mathbf{T}$ denote $\langle T_1, \ldots, T_n, T\rangle$. Independently of whether W contains a specialization of f with the type signature $\mathbf{T}\theta$, the resulting function application is interpreted in $\mathfrak{I}$ exactly as f on $\mathbf{T}\theta\pi = \mathbf{T}\pi'$ in $\mathfrak{M}$. Therefore, $\mathfrak{I}_{\pi,\varphi}(\textsc{Dis}_\theta(t)) = \mathfrak{M}_{\pi',\varphi'}(t)$.
3. Consider a universally quantified formula $\forall(u:T)\,F$. As we have shown above, $\mathcal{D}^{\mathfrak{I}}_{T\theta\pi} = \mathcal{D}^{\mathfrak{M}}_{T\pi'}$. Also, for any domain element $a \in \mathcal{D}^{\mathfrak{I}}_{T\theta\pi}$, the variable valuation $\varphi[u_T : T\theta \mapsto a]$ in $\mathfrak{I}$ produces the valuation $\varphi'[u:T \mapsto a]$ in $\mathfrak{M}$. Thus:
$$\begin{aligned}\mathfrak{I}_{\pi,\varphi}(\textsc{Dis}_\theta(\forall(u:T)\,F)) &= \bigwedge_{a\in\mathcal{D}^{\mathfrak{I}}_{T\theta\pi}} \mathfrak{I}_{\pi,\varphi[u_T : T\theta \mapsto a]}(\textsc{Dis}_\theta(F)) \\ &= \bigwedge_{a\in\mathcal{D}^{\mathfrak{M}}_{T\pi'}} \mathfrak{M}_{\pi',\varphi'[u:T\mapsto a]}(F) = \mathfrak{M}_{\pi',\varphi'}(\forall(u:T)\,F)\end{aligned}$$

Thus, for every formula F in Γ, $\textsc{Dis}(F)$ is satisfied under $\mathfrak{I}$. □

Lemma 4. *For any closed formula F and an arbitrary type valuation π, there exists a unique maximal (with respect to the set of instantiated type variables) type substitution in $\Theta^\star(F)$ compatible with (i.e., included in) π.*

Proof. Consider the set S of all type substitutions in $\Theta^\star(F)$ included in π. This set is necessarily finite, since F contains a finite number of type variables. The type substitutions in S are pairwise compatible. Therefore, S is a complete upper semi-lattice and its maximal element is included in π. □

Theorem 3 (Completeness of Dis). *For every set of closed formulas Γ, if $\textsc{Dis}(\Gamma)$ is satisfiable then Γ is satisfiable.*

Proof. Let $\mathfrak{M}$ be a model of $\textsc{Dis}(\Gamma)$. We are going to construct an interpretation $\mathfrak{I}$ of Γ such that for every closed formula H in Γ, every type valuation π, and every variable valuation φ in $\mathfrak{I}$, there exist a type substitution $\theta \in \Theta^\star(H)$, a type valuation π', and a variable valuation φ' in the original model $\mathfrak{M}$ such that:
$$\mathfrak{I}_{\pi,\varphi}(t) = \mathfrak{M}_{\pi',\varphi'}(\textsc{Dis}_\theta(t)) \qquad \mathfrak{I}_{\pi,\varphi}(F) = \mathfrak{M}_{\pi',\varphi'}(\textsc{Dis}_\theta(F))$$

for every term t and formula F occurring in H. Since for every $\theta \in \Theta^\star(H)$, the model $\mathfrak{M}$ satisfies $\textsc{Dis}_\theta(H)$, the initial formula H is satisfied under $\mathfrak{I}$.

For any sort S in the signature of Γ, we set $\mathcal{D}^{\mathfrak{I}}_S \triangleq \mathcal{D}^{\mathfrak{M}}_S$. Consider a function symbol f in Γ, a vector of sorts $\mathbf{T} = \langle T_1, \ldots, T_n, T\rangle$, and a type substitution τ that instantiates the type signature of f to $\mathbf{T}$. If the monomorphic specialization f_τ belongs to W, we define $f^{\mathfrak{I}}_{\mathbf{T}} \triangleq (f_\tau)^{\mathfrak{M}}_{\mathbf{T}}$. Otherwise, $f^{\mathfrak{I}}_{\mathbf{T}} \triangleq f^{\mathfrak{M}}_{\mathbf{T}}$. The predicate symbols are interpreted in the same way.

Now, consider a formula H in Γ, a type valuation π, and a variable valuation φ in $\mathfrak{I}$. By Lemma 4, $\Theta^\star(H)$ contains a maximal monomorphic type substitution θ compatible with π (and thus included in π). We set $\pi' \triangleq \pi$, so that $\theta \circ \pi' = \pi$. For every variable $u : T$ occurring in H, we set $\varphi'(u_T : T\theta) \triangleq \varphi(u : T)$. Indeed for every type in F, $\mathcal{D}^{\mathfrak{I}}_{T\pi} = \mathcal{D}^{\mathfrak{M}}_{T\pi} = \mathcal{D}^{\mathfrak{M}}_{T\theta\pi'}$. By Lemma 1, it is sufficient to define φ' on variables that occur in $\textsc{Dis}(H)$.

Using this definitions, we proceed to prove the two equalities above. The proof goes by induction on the structure of terms and formulas. We consider three cases: a variable, a function application, and a universally quantified formula (other cases can be handled along the same lines).

1. Consider a variable $u : T$ occurring in H. We obtain:

$$\mathfrak{I}_{\pi,\varphi}(u : T) = \varphi(u : T) = \varphi'(u_T : T\theta)) = \mathfrak{M}_{\pi',\varphi'}(\textsc{Dis}_\theta(u : T))$$

2. Consider a term $t = f(t_1 : T_1, \ldots, t_n : T_n) : T$ occurring in H. Let $\mathbf{T}$ denote $\langle T_1, \ldots, T_n, T\rangle$ and let τ be the type substitution that instantiates the type signature of f to $\mathbf{T}\theta$. Suppose that the specialization f_τ belongs to W. Thus $\textsc{Dis}_\theta$ replaces f with f_τ and $\mathbf{T}\pi = \mathbf{T}\theta$. We obtain:

$$\begin{aligned}
\mathfrak{I}_{\pi,\varphi}(t) &= f^{\mathfrak{I}}_{\mathbf{T}\pi}(\mathfrak{I}_{\pi,\varphi}(t_1), \ldots, \mathfrak{I}_{\pi,\varphi}(t_n)) \\
&= (f_\tau)^{\mathfrak{M}}_{\mathbf{T}\theta\pi'}(\mathfrak{M}_{\pi',\varphi'}(\textsc{Dis}_\theta(t_1)), \ldots, \mathfrak{M}_{\pi',\varphi'}(\textsc{Dis}_\theta(t_n))) \\
&= \mathfrak{M}_{\pi',\varphi'}(f_\tau(\textsc{Dis}_\theta(t_1), \ldots, \textsc{Dis}_\theta(t_n))) \\
&= \mathfrak{M}_{\pi',\varphi'}(\textsc{Dis}_\theta(t))
\end{aligned}$$

   Now, let us assume that f_τ does not belong to W, so that $\textsc{Dis}_\theta$ keeps the symbol f in t. Then we can show that there is no monomorphic specialization of f in W with the type signature $\mathbf{T}\pi$. Indeed, if there is one, then the restriction of π to the type variables of $\mathbf{T}$ belongs to $\Theta(H)$. This restriction is compatible with θ, since both of them are included in π. As θ is maximal, this restriction is also included in θ. This gives us $\mathbf{T}\pi = \mathbf{T}\theta$, which leads to a contradiction. We obtain:

$$\begin{aligned}
\mathfrak{I}_{\pi,\varphi}(t) &= f^{\mathfrak{I}}_{\mathbf{T}\pi}(\mathfrak{I}_{\pi,\varphi}(t_1), \ldots, \mathfrak{I}_{\pi,\varphi}(t_n)) \\
&= f^{\mathfrak{M}}_{\mathbf{T}\theta\pi'}(\mathfrak{M}_{\pi',\varphi'}(\textsc{Dis}_\theta(t_1)), \ldots, \mathfrak{M}_{\pi',\varphi'}(\textsc{Dis}_\theta(t_n))) \\
&= \mathfrak{M}_{\pi',\varphi'}(f(\textsc{Dis}_\theta(t_1), \ldots, \textsc{Dis}_\theta(t_n))) \\
&= \mathfrak{M}_{\pi',\varphi'}(\textsc{Dis}_\theta(t))
\end{aligned}$$

3. Consider a universally quantified formula $\forall(u:T)\,F$. As we have seen above, $\mathcal{D}^{\mathfrak{I}}_{T\pi} = \mathcal{D}^{\mathfrak{M}}_{T\theta\pi'}$. Also, for any $a \in \mathcal{D}^{\mathfrak{M}}_{T\theta\pi'}$, the variable valuation $\varphi[u:T \mapsto a]$ in $\mathfrak{I}$ produces the valuation $\varphi'[u_T:T\theta \mapsto a]$ in $\mathfrak{M}$. We obtain:

$$\begin{aligned}\mathfrak{I}_{\pi,\varphi}(\forall(u:T)\,F) &= \bigwedge_{a\in\mathcal{D}^{\mathfrak{I}}_{T\pi}} \mathfrak{I}_{\pi,\varphi[u:T\mapsto a]}(F)\\ &= \bigwedge_{a\in\mathcal{D}^{\mathfrak{M}}_{T\theta\pi'}} \mathfrak{M}_{\pi',\varphi'[u_T:T\theta\mapsto a]}(\textsc{Dis}_\theta(F))\\ &= \mathfrak{M}_{\pi',\varphi'}(\forall(u_T:T\theta)\,\textsc{Dis}_\theta(F)) = \mathfrak{M}_{\pi',\varphi'}(\textsc{Dis}_\theta(\forall(u:T)\,F))\end{aligned}$$

Thus, for every formula H in Γ, H is satisfied under $\mathfrak{I}$. □

The definition of Dis can be generalized to a case where W admits polymorphic specializations. This requires W to be closed with respect to unification of type signatures, so that we can always choose the most refined specialization symbol during discrimination. The substitutions in the set $\Theta(F)$ must be considered modulo renaming of type variables in the signatures of specialization symbols. Finally, the union of two substitutions would be their most general common refinement. However, since our transformations target monomorphic theorem provers, we find this generalization of lesser practical interest and do not pursue it in this paper.

3.2 Partial Monomorphisation

It turns out that the above-presented approach of type instantiation with symbol discrimination can be advanced to actually protect types in the sense of the introduction to Section 3.

Let U be a set of sorts that we want to preserve across our type-eliminating transformations. The set U is fixed for the rest of this section and the Par transformation is parametrized by it. Given a set of polymorphic types M, we define a set of monomorphic substitutions $\Theta(M)$ as follows:

$$\begin{aligned}\Theta(M) \triangleq \{\,\theta \mid\ & M \text{ contains a type } T \text{ such that}\\ & \theta \text{ only instantiates the type variables of } T \text{ and}\\ & T\theta \text{ belongs to } U\,\}\end{aligned}$$

The closure of $\Theta(M)$ with respect to finite unions, denoted $\Theta^\star(M)$, is defined as in the previous section. Recall that the identity substitution always belongs to the closure.

Lemma 5. *Let M be a set of polymorphic types and π a type substitution. Then there exists a unique monomorphic type substitution $\theta \in \Theta^\star(M)$ such that:*

- *θ is included in π (that is, for every α, either $\alpha\theta = \alpha$ or $\alpha\theta = \alpha\pi$);*
- *for every type $T \in M$, $T\pi \in U$ if and only if $T\theta \in U$.*

Proof. Let M' be the set of types from M that are instantiated "into U" by π. Let θ be the restriction of π onto the type variables occurring in M'. It is easy to see that θ is a finite union of substitutions in $\Theta(M)$ that instantiate individual types in M' into U. Indeed, all of these individual substitutions are included in π and thus are compatible. Now, let T be an arbitrary type in M. If $T\pi$ belongs to U, then $T \in M'$ and hence $T\theta = T\pi$. If $T\theta$ belongs to U, then $T\pi \in U$, too, since all types in U are closed and θ is included in π.

Now, suppose that there is another monomorphic substitution θ' in $\Theta^\star(M)$ that satisfies both conditions. Since both θ and θ' are included in π, they cannot substitute two different sorts for the same type variable. Also, θ' must coincide with θ (and π) on every type variable in M'. Therefore, θ is included in θ'. If θ' differs from θ, then there exists some type variable α outside M' instantiated by θ'. By definition of $\Theta(M)$, there must therefore exist a type $T \in M \backslash M'$ such that $T\theta' \in U$. This contradicts the second condition. □

Given a type T, the transformed type $[T]$ is defined as follows: if $T \in U$, we set $[T] \triangleq \overline{T}$, otherwise $[T] \triangleq T$. Now, let f be a function symbol of type signature $\langle S_1, \ldots, S_n, S\rangle$ in Γ. Let τ be a type substitution in $\Theta^\star(\{S_1, \ldots, S_n, S\})$. A fresh function symbol $\bar{f}_\tau$ with the type signature $\langle [S_1\tau], \ldots, [S_n\tau], [S\tau]\rangle$ will be called a *protected specialization* of f. Protected specializations of predicate symbols are defined in the same way.

The idea is to produce every specialization that makes protected sorts appear in the type signature. For example, let M be $\{\mathsf{M}(\mathsf{I},\mathsf{I})\}$. Then the symbol $\mathtt{get}$ with the type signature $\langle \mathsf{M}(\alpha,\beta),\alpha,\beta\rangle$ induces five protected specializations:

$$\begin{aligned}
\mathtt{get}_{[]} &: \langle \mathsf{M}(\alpha,\beta),\alpha,\beta\rangle\\
\mathtt{get}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]} &: \langle \overline{\mathsf{M}(\mathsf{I},\mathsf{I})},\mathsf{I},\mathsf{I}\rangle\\
\mathtt{get}_{[\mathsf{M}(\mathsf{I},\mathsf{I})/\alpha]} &: \langle \mathsf{M}(\mathsf{M}(\mathsf{I},\mathsf{I}),\beta),\overline{\mathsf{M}(\mathsf{I},\mathsf{I})},\beta\rangle\\
\mathtt{get}_{[\mathsf{M}(\mathsf{I},\mathsf{I})/\beta]} &: \langle \mathsf{M}(\alpha,\mathsf{M}(\mathsf{I},\mathsf{I})),\alpha,\overline{\mathsf{M}(\mathsf{I},\mathsf{I})}\rangle\\
\mathtt{get}_{[\mathsf{M}(\mathsf{I},\mathsf{I})/\alpha,\mathsf{M}(\mathsf{I},\mathsf{I})/\beta]} &: \langle \mathsf{M}(\mathsf{M}(\mathsf{I},\mathsf{I}),\mathsf{M}(\mathsf{I},\mathsf{I})),\overline{\mathsf{M}(\mathsf{I},\mathsf{I})},\overline{\mathsf{M}(\mathsf{I},\mathsf{I})}\rangle
\end{aligned}$$

The Par transformation modifies the signature of a transformed theory:

1. For every variable symbol u and type T, we add a new variable symbol u_T.
2. We replace every function and predicate symbol with the set of all its protected specializations.

Given an arbitrary substitution θ, the transformation $\textsc{Par}_\theta$ instantiates and converts terms and formulas into the new signature:

1. Given a variable $u\!:\!T$, $\textsc{Par}_\theta(u\!:\!T) \triangleq u_T\!:\![T\theta]$.
2. Consider a term $t = f(t_1\!:\!T_1, \ldots, t_n\!:\!T_n)\!:\!T$. By Lemma 5, there is a unique protected specialization $\bar{f}_\tau$ whose signature matches $\langle [T_1\theta], \ldots, [T_n\theta], [T\theta]\rangle$. Indeed, if $T_i\theta$ is in U, then the only type that matches $[T_i\theta]$ is $[T_i\theta]$ itself. Then $\textsc{Par}_\theta(t) \triangleq \bar{f}_\tau(\textsc{Par}_\theta(t_1), \ldots, \textsc{Par}_\theta(t_n))\!:\![T\theta]$.

3. Consider an atomic formula $F = p(t_1 : T_1, \ldots, t_n : T_n)$. By Lemma 5, there is one protected specialization $\bar{p}_\tau$ whose signature matches $\langle [T_1\theta], \ldots, [T_n\theta] \rangle$. Then $\textsc{Par}_\theta(F) \triangleq \bar{p}_\tau(\textsc{Par}_\theta(t_1), \ldots, \textsc{Par}_\theta(t_n))$.

Equalities and complex formulas are converted in a natural way:

$$\textsc{Par}_\theta(t_1 \approx t_2) \triangleq \textsc{Par}_\theta(t_1) \approx \textsc{Par}_\theta(t_2) \qquad \textsc{Par}_\theta(F \wedge G) \triangleq \textsc{Par}_\theta(F) \wedge \textsc{Par}_\theta(G)$$
$$\textsc{Par}_\theta(\neg F) \triangleq \neg\, \textsc{Par}_\theta(F) \qquad \textsc{Par}_\theta(\forall x\, F) \triangleq \forall(\textsc{Par}_\theta(x))\, \textsc{Par}_\theta(F)$$

Finally, Par translates a closed formula F into a set of formulas:

$$\textsc{Par}(F) \triangleq \{\, \textsc{Par}_\theta(F) \mid \theta \in \Theta^\star(M) \,\}$$

where M is the set of types of all terms that occur in F.

On our running example, supposing that $U = \{\mathsf{I}\}$, Par produces the following two formulas (via $\textsc{Par}_{[\,]}$ and $\textsc{Par}_{[\mathsf{I}/\alpha]}$, respectively):

$$\forall(m_{\mathsf{M}(\alpha,\mathsf{I})} : \mathsf{M}(\alpha, \mathsf{I}))\, \forall(c_\alpha : \alpha)\, \mathtt{get}_{[\mathsf{I}/\beta]}(\mathtt{set}_{[\mathsf{I}/\beta]}(m_{\mathsf{M}(\alpha,\mathsf{I})}, c_\alpha, 6), c_\alpha) * 7 \approx 42$$
$$\forall(m_{\mathsf{M}(\alpha,\mathsf{I})} : \mathsf{M}(\mathsf{I}, \mathsf{I}))\, \forall(c_\alpha : \bar{\mathsf{I}})\, \mathtt{get}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}(\mathtt{set}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}(m_{\mathsf{M}(\alpha,\mathsf{I})}, c_\alpha, 6), c_\alpha) * 7 \approx 42$$

The new symbol $\mathtt{get}_{[\mathsf{I}/\beta]}$ has the type signature $\langle \mathsf{M}(\alpha, \mathsf{I}), \alpha, \bar{\mathsf{I}} \rangle$ and $\mathtt{get}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}$ has the monomorphic type signature $\langle \mathsf{M}(\mathsf{I}, \mathsf{I}), \bar{\mathsf{I}}, \bar{\mathsf{I}} \rangle$.

Lemma 6. *Let θ be a type substitution, t a term of type T, and F a formula. Then $\textsc{Par}_\theta(t)$ is a well-formed term of type $[T\theta]$ and $\textsc{Par}_\theta(F)$ is a well-formed formula such that $\mathrm{FV}(\textsc{Par}_\theta(F)) = \{\textsc{Par}_\theta(x) \mid x \in \mathrm{FV}(F)\}$ and $\mathrm{FV}_\mathrm{T}(\textsc{Par}_\theta(F)) = \mathrm{FV}_\mathrm{T}(F) \backslash \mathrm{dom}(\theta)$.*

Proof. The proof goes by induction on the structure of terms and formulas. The only interesting case is the application of a function symbol (predicate symbols are treated in the same way). Let t be a term of the form $f(t_1 : T_1, \ldots, t_n : T_n) : T$. For every t_i, $\textsc{Par}_\theta(t_i)$ is a well-formed term of type $[T_i\theta]$ by the induction hypothesis. By definition, the type signature of $\bar{f}_\tau$ matches $\langle [T_1\theta], \ldots, [T_n\theta], [T\theta] \rangle$. Thus, $\textsc{Par}_\theta(t)$ is a well-formed term of type $[T\theta]$. □

Theorem 4 (Soundness of Par). *If a set of closed formulas Γ is satisfiable then $\textsc{Par}(\Gamma)$ is satisfiable.*

Proof. Let $\mathfrak{M}$ be a model of Γ. We are going to construct an interpretation $\mathfrak{I}$ of $\textsc{Par}(\Gamma)$ such that, for every type substitution θ, type valuation π, and variable valuation φ in $\mathfrak{I}$, there exist a type valuation π' and a variable valuation φ' in the original model $\mathfrak{M}$, such that:

$$\mathfrak{I}_{\pi,\varphi}(\textsc{Par}_\theta(t)) = \mathfrak{M}_{\pi',\varphi'}(t) \qquad \mathfrak{I}_{\pi,\varphi}(\textsc{Par}_\theta(F)) = \mathfrak{M}_{\pi',\varphi'}(F)$$

for every term t and formula F occurring in Γ. Since for every closed $F \in \Gamma$ and every θ, $\mathfrak{I}$ satisfies $\textsc{Par}_\theta(F)$, $\mathfrak{I}$ is a model of $\textsc{Par}(\Gamma)$.

For any sort S in Γ, we define $\mathcal{D}^{\mathfrak{I}}_S \triangleq \mathcal{D}^{\mathfrak{M}}_S$ and $\mathcal{D}^{\mathfrak{I}}_{\bar{S}} \triangleq \mathcal{D}^{\mathfrak{M}}_S$. Then we give an interpretation to the function and predicate symbols in $\textsc{Par}(\Gamma)$. Consider a function symbol f from Γ and its protected specialization $\bar{f}_\tau$. Denoting the type signature of f with $\mathbf{S}$, the type signature of $\bar{f}_\tau$ is $[\mathbf{S}\tau]$. Let $\mathbf{T}$ be a vector of sorts such that $\mathbf{T} = [\mathbf{S}\tau]\tau'$ for some type substitution τ'. Then we interpret $\bar{f}_\tau$ on $\mathbf{T}$ in $\mathfrak{I}$ exactly as f is interpreted on $\mathbf{S}\tau\tau'$ in $\mathfrak{M}$. This is correct, as $\mathbf{S}\tau\tau'$ is just $\mathbf{T}$ with protection removed. Predicate symbols are interpreted in the same way.

Now, let us consider some type substitution θ, a type valuation π, and a variable valuation φ in $\mathfrak{I}$. We set $\pi' \triangleq \theta \circ \pi$ and we set $\varphi'(u:T) \triangleq \varphi(u_T:[T\theta])$. Notice that for any type T in Γ, $\mathcal{D}^{\mathfrak{I}}_{[T\theta]\pi} = \mathcal{D}^{\mathfrak{M}}_{T\theta\pi} = \mathcal{D}^{\mathfrak{M}}_{T\pi'}$.

Using this definitions, we proceed to prove the two equalities above. The proof goes by induction on the structure of terms and formulas. We consider three cases: a variable, a function application, and a universally quantified formula (other cases can be handled along the same lines).

1. Consider a variable $u:T$ occurring in Γ. We obtain:

$$\mathfrak{I}_{\pi,\varphi}(\textsc{Par}_\theta(u:T)) = \varphi(u_T:[T\theta]) = \varphi'(u:T) = \mathfrak{M}_{\pi',\varphi'}(u:T)$$

2. Consider a term $t = f(t_1:T_1,\ldots,t_n:T_n):T$. Let f_τ be the protected specialization of f whose type signature matches $[\mathbf{T}\theta]$, where $\mathbf{T} = \langle T_1,\ldots,T_n,T\rangle$.

$$\begin{aligned}
\mathfrak{I}_{\pi,\varphi}(\textsc{Par}_\theta(t)) &= \mathfrak{I}_{\pi,\varphi}(\bar{f}_\tau(\textsc{Par}_\theta(t_1),\ldots,\textsc{Par}_\theta(t_n)):[T\theta]) \\
&= (f_\tau)^{\mathfrak{I}}_{[\mathbf{T}\theta]\pi}(\mathfrak{I}_{\pi,\varphi}(\textsc{Par}_\theta(t_1)),\ldots,\mathfrak{I}_{\pi,\varphi}(\textsc{Par}_\theta(t_n))) \\
&= f^{\mathfrak{M}}_{\mathbf{T}\theta\pi}(\mathfrak{M}_{\pi',\varphi'}(t_1),\ldots,\mathfrak{M}_{\pi',\varphi'}(t_n)) \\
&= f^{\mathfrak{M}}_{\mathbf{T}\pi'}(\mathfrak{M}_{\pi',\varphi'}(t_1),\ldots,\mathfrak{M}_{\pi',\varphi'}(t_n)) \\
&= \mathfrak{M}_{\pi',\varphi'}(t)
\end{aligned}$$

3. Consider a universally quantified formula $\forall(u:T)\,F$. As we have shown above, $\mathcal{D}^{\mathfrak{I}}_{[T\theta]\pi} = \mathcal{D}^{\mathfrak{M}}_{T\pi'}$. Also, for any domain element $a \in \mathcal{D}^{\mathfrak{I}}_{[T\theta]\pi}$, the variable valuation $\varphi[u_T:[T\theta] \mapsto a]$ in $\mathfrak{I}$ produces the valuation $\varphi'[u:T \mapsto a]$ in $\mathfrak{M}$.

$$\begin{aligned}
\mathfrak{I}_{\pi,\varphi}(\textsc{Par}_\theta(\forall(u:T)\,F)) &= \bigwedge_{a\in\mathcal{D}^{\mathfrak{I}}_{[T\theta]\pi}} \mathfrak{I}_{\pi,\varphi[u_T:[T\theta]\mapsto a]}(\textsc{Par}_\theta(F)) \\
&= \bigwedge_{a\in\mathcal{D}^{\mathfrak{M}}_{T\pi'}} \mathfrak{M}_{\pi',\varphi'[u:T\mapsto a]}(F) = \mathfrak{M}_{\pi',\varphi'}(\forall(u:T)\,F)
\end{aligned}$$

Thus, for every formula F in Γ, $\textsc{Par}(F)$ is satisfied under $\mathfrak{I}$. □

Theorem 5 (Completeness of Par). *For every set of closed formulas Γ, if $\textsc{Par}(\Gamma)$ is satisfiable then Γ is satisfiable.*

Proof. Let $\mathfrak{M}$ be a model of $\textsc{Par}(\Gamma)$. We are going to construct an interpretation $\mathfrak{I}$ of Γ such that for every formula H in Γ, every type valuation π, and every variable interpretation φ in $\mathfrak{I}$, there exist a type substitution

$\theta \in \Theta^\star(\{T \mid T \text{ is the type of a term in } H\})$, a type valuation π', and a variable valuation φ' in the original model $\mathfrak{M}$ such that:

$$\mathfrak{I}_{\pi,\varphi}(t) = \mathfrak{M}_{\pi',\varphi'}(\textsc{Par}_\theta(t)) \qquad \mathfrak{I}_{\pi,\varphi}(F) = \mathfrak{M}_{\pi',\varphi'}(\textsc{Par}_\theta(F))$$

for every term t and formula F occurring in H.

For any sort S in the signature of Γ, we set $\mathcal{D}^{\mathfrak{I}}_S \triangleq \mathcal{D}^{\mathfrak{M}}_{[S]}$. Consider a function symbol f in Γ and a vector of sorts $\mathbf{T} = \langle T_1, \ldots, T_n, T\rangle$ matched by the type signature of f. By Lemma 5, there is a unique protected specialization $\bar{f}_\tau$ whose type signature matches the sort vector $[\mathbf{T}]$. Then we define $f^{\mathfrak{I}}_{\mathbf{T}} \triangleq (\bar{f}_\tau)^{\mathfrak{M}}_{[\mathbf{T}]}$. The predicate symbols are interpreted in the same way.

Now, consider a formula H from Γ, a type valuation π, and a variable valuation φ in $\mathfrak{I}$. Let M be the set of types of all terms occurring in H. By Lemma 5, there exists a unique monomorphic type substitution $\theta \in \Theta^\star(M)$ such that θ is included in π and for every type $T \in M$, $T\pi \in U$ if and only if $T\theta \in U$. We set $\pi' \triangleq \pi$, so that $\theta \circ \pi' = \pi$. Let a variable $u\!:\!T$ occur in H. If it occurs in H as a term, then $T \in M$ and we set $\varphi'(u_T\!:\![T\theta]) \triangleq \varphi(u\!:\!T)$. Indeed, for every type T in M, $\mathcal{D}^{\mathfrak{I}}_{T\pi} = \mathcal{D}^{\mathfrak{M}}_{[T\pi]} = \mathcal{D}^{\mathfrak{M}}_{[T\theta]\pi'}$ by definition of θ. Otherwise, if $u\!:\!T$ is only bound by a quantifier in H but does not occur under this quantifier, we can safely ignore it in variable valuations by Lemma 1.

Using this definitions, we proceed to prove the two equalities above. The proof goes by induction on the structure of terms and formulas. We consider three cases: a variable, a function application, and a universally quantified formula (other cases can be handled along the same lines).

1. Consider a variable $u\!:\!T$ occurring in H as a term.

$$\mathfrak{I}_{\pi,\varphi}(u\!:\!T) = \varphi(u\!:\!T) = \varphi'(u_T\!:\![T\theta]) = \mathfrak{M}_{\pi',\varphi'}(\textsc{Par}_\theta(u\!:\!T))$$

2. Consider a term $t = f(t_1\!:\!T_1, \ldots, t_n\!:\!T_n)\!:\!T$ occurring in H and let $\mathbf{T}$ denote $\langle T_1, \ldots, T_n, T\rangle$. Let $\bar{f}_\tau$ be the protected specialization of f whose type signature matches $[\mathbf{T}\theta]$. By definition of θ, $[\mathbf{T}\theta]$ matches $[\mathbf{T}\pi]$. Then $\bar{f}_\tau$ is also the only protected specialization of f whose type signature matches $[\mathbf{T}\pi]$.

$$\begin{aligned}
\mathfrak{I}_{\pi,\varphi}(t) &= f^{\mathfrak{I}}_{\mathbf{T}\pi}(\mathfrak{I}_{\pi,\varphi}(t_1), \ldots, \mathfrak{I}_{\pi,\varphi}(t_n)) \\
&= (\bar{f}_\tau)^{\mathfrak{M}}_{[\mathbf{T}\pi]}(\mathfrak{M}_{\pi',\varphi'}(\textsc{Par}_\theta(t_1)), \ldots, \mathfrak{M}_{\pi',\varphi'}(\textsc{Par}_\theta(t_n))) \\
&= (\bar{f}_\tau)^{\mathfrak{M}}_{[\mathbf{T}\theta]\pi'}(\mathfrak{M}_{\pi',\varphi'}(\textsc{Par}_\theta(t_1)), \ldots, \mathfrak{M}_{\pi',\varphi'}(\textsc{Par}_\theta(t_n))) \\
&= \mathfrak{M}_{\pi',\varphi'}(\bar{f}_\tau(\textsc{Par}_\theta(t_1), \ldots, \textsc{Par}_\theta(t_n))) = \mathfrak{M}_{\pi',\varphi'}(\textsc{Par}_\theta(t))
\end{aligned}$$

3. Consider a universally quantified formula $\forall(u\!:\!T)\,F$ occurring in H. We can assume that $u\!:\!T$ occurs in F and $T \in M$. Then $\mathcal{D}^{\mathfrak{I}}_{T\pi} = \mathcal{D}^{\mathfrak{M}}_{[T\theta]\pi'}$. Also, for any element $a \in \mathcal{D}^{\mathfrak{M}}_{[T\theta]\pi'}$, the variable valuation $\varphi[u\!:\!T \mapsto a]$ in $\mathfrak{I}$ produces

the valuation $\varphi'[u_T : [T\theta] \mapsto a]$ in $\mathfrak{M}$. We obtain:

$$\begin{aligned}\mathfrak{I}_{\pi,\varphi}(\forall(u:T)\,F) &= \bigwedge_{a \in \mathcal{D}^{\mathfrak{I}}_{T\pi}} \mathfrak{I}_{\pi,\varphi[u:T \mapsto a]}(F) \\ &= \bigwedge_{a \in \mathcal{D}^{\mathfrak{M}}_{[T\theta]\pi'}} \mathfrak{M}_{\pi',\varphi'[u_T:[T\theta] \mapsto a]}(\textsc{Par}_\theta(F)) \\ &= \mathfrak{M}_{\pi',\varphi'}(\forall(u_T:[T\theta])\,\textsc{Par}_\theta(F)) = \mathfrak{M}_{\pi',\varphi'}(\textsc{Par}_\theta(\forall(u:T)\,F))\end{aligned}$$

Thus, for every formula H in Γ, H is satisfied under $\mathfrak{I}$. □

In practice, the Par transformation behaves reasonably well on small sets U, containing two or three sorts. If we try to protect more — say, every sort constant occurring in the problem — the exponential growth due to type instantiation (w.r.t. the number of type variables in a formula) quickly makes the proof search infeasible. Therefore, we do not consider Par as a practical alternative to the Tw method, presented in Section 3.3. Yet, in our view, the existence of an alternative type protection technique justifies the separation of type protection from type fusion and shows that the notion of protected sort gives a good independent characterization of the boundary between the two stages.

3.3 Twin Sorts

The Tw transformation converts a set of formulas into an equisatisfiable set with protected sorts. It applies a pair of conversion functions to pass, wherever necessary, from a protected sort to a non-protected one and vice versa.

Let U be a set of sorts that we want to protect as in the previous section. Given a type T, the transformed type $[T]$ is $\overline{T}$ if $T \in U$, and T otherwise. Then Tw modifies the signature of a transformed theory as follows:

1. We replace every function symbol f of type signature $\langle S_1, \ldots, S_n, S\rangle$ with a symbol $\bar{f}$ of type signature $\langle [S_1], \ldots, [S_n], [S]\rangle$.
2. We replace every predicate symbol p of type signature $\langle S_1, \ldots, S_n\rangle$ with a symbol $\bar{p}$ of type signature $\langle [S_1], \ldots, [S_n]\rangle$.
3. For every sort $T \in U$, we add a pair of "bridge" function symbols $\mathtt{to}_T : \langle \overline{T}, T\rangle$ and $\mathtt{from}_T : \langle T, \overline{T}\rangle$.

Then we convert terms and atomic formulas into the new signature. Our aim is to forbid a polymorphic type in a symbol's type signature being instantiated into a type from U. Whenever such instantiation takes place, a bridge function is applied. In more precise terms:

1. Given a variable $u:T$, $\textsc{Tw}(u:T) \triangleq u:[T]$.

2. Consider a term $t = f(t_1 : T_1, \ldots, t_n : T_n) : T$ and let $\langle S_1, \ldots, S_n, S \rangle$ be the type signature of f.

$$\text{For every } t_i,\ t_i' \triangleq \begin{cases} \mathtt{to}_{T_i}(\mathrm{Tw}(t_i)) : T_i & \text{if } T_i \in U \text{ and } S_i \notin U, \\ \mathrm{Tw}(t_i) & \text{if } T_i \notin U \text{ or } S_i \in U. \end{cases}$$

$$\text{Then } \mathrm{Tw}(t) \triangleq \begin{cases} \bar{f}(t_1', \ldots, t_n') : [T] & \text{if } T \notin U \text{ or } S \in U, \\ \mathtt{from}_T(\bar{f}(t_1', \ldots, t_n') : T) : \overline{T} & \text{if } T \in U \text{ and } S \notin U. \end{cases}$$

3. Consider a formula $p(t_1 : T_1, \ldots, t_n : T_n)$ and let $\langle S_1, \ldots, S_n \rangle$ be the type signature of p. For every argument t_i, we define t_i' as in the previous case. Then, $\mathrm{Tw}(p(t_1 : T_1, \ldots, t_n : T_n)) \triangleq \bar{p}(t_1', \ldots, t_n')$.

Equalities and complex formulas are converted in a natural way:

$$\mathrm{Tw}(t_1 \approx t_2) \triangleq \mathrm{Tw}(t_1) \approx \mathrm{Tw}(t_2) \qquad \mathrm{Tw}(F \wedge G) \triangleq \mathrm{Tw}(F) \wedge \mathrm{Tw}(G)$$
$$\mathrm{Tw}(\neg F) \triangleq \neg\, \mathrm{Tw}(F) \qquad \mathrm{Tw}(\forall x\, F) \triangleq \forall(\mathrm{Tw}(x))\, \mathrm{Tw}(F)$$

Finally, we convert the formulas in Γ and add axioms for the bridge functions:

$$\begin{aligned} \mathrm{Tw}(\Gamma) \triangleq & \{\, \mathrm{Tw}(F) \mid F \in \Gamma \,\} \\ & \cup \{\, \forall(v : \overline{T})\ \mathtt{from}_T(\mathtt{to}_T(v : \overline{T})) \approx v : \overline{T} \mid T \in U \,\} \\ & \cup \{\, \forall(u : T)\ \mathtt{to}_T(\mathtt{from}_T(u : T)) \approx u : T \mid T \in U \,\} \end{aligned}$$

Assuming $U = \{\mathsf{I}\}$, the running example is transformed as follows. Notice that `6`, `7`, and `42` have the type $\overline{\mathsf{I}}$ and the type signature of $*$ is $\langle \overline{\mathsf{I}}, \overline{\mathsf{I}}, \overline{\mathsf{I}} \rangle$.

$$\forall(m : \mathsf{M}(\alpha, \mathsf{I}))\, \forall(c : \alpha)\, \mathtt{from}_{\mathsf{I}}(\mathtt{get}(\mathtt{set}(m, c, \mathtt{to}_{\mathsf{I}}(\mathtt{6}) : \mathsf{I}), c) : \mathsf{I}) * \mathtt{7} \approx \mathtt{42}$$

Lemma 7. *For every term t of type T, $\mathrm{Tw}(t)$ is a well-formed term of type $[T]$, and for every formula F, $\mathrm{Tw}(F)$ is a well-formed formula with the same free variables (modulo conversion of their types) and type variables.*

Proof. The proof goes by induction on the structure of terms and formulas. The only interesting case is the application of a function symbol (predicate symbols are treated in the same way). The conservation of free (type) variables is trivial.

Let t be a term of the form $f(t_1 : T_1, \ldots, t_n : T_n) : T$. Let $\langle S_1, \ldots, S_n, S \rangle$ be the type signature of f and τ the type substitution that instantiates $\langle S_1, \ldots, S_n, S \rangle$ to $\langle T_1, \ldots, T_n, T \rangle$. For every argument t_i, $\mathrm{Tw}(t_i)$ is a well-formed term of type $[T_i]$ by the induction hypothesis, and there are three cases to consider:

1. $T_i \in U$ and $S_i \notin U$. Then $\mathrm{Tw}(t_i)$ has the type $\overline{T}_i$ and $\mathtt{to}_{T_i}(\mathrm{Tw}(t_i)) : T_i$ is a well-formed term. Notice that the i-th element of the type signature of $\bar{f}$ is $[S_i] = S_i$, and $S_i\tau = T_i$.
2. $S_i \in U$. Then S_i is closed, so $T_i = S_i$, and $[T_i] = [S_i] = [S_i]\tau$.
3. $S_i \notin U$ and $T_i \notin U$. Then $[S_i]\tau = S_i\tau = T_i = [T_i]$.

Then we apply $\bar{f}$ to $t'_1, \ldots, t'_n$ using τ to instantiate the type signature of $\bar{f}$. Again, we have three possible cases:

1. $T \in U$ and $S \notin U$. Since $S\tau = T$, we obtain $[S]\tau = S\tau = T$. Then the term $\mathtt{from}_T(\bar{f}(t'_1, \ldots, t'_n) : T) : \overline{T}$ is well-formed and has the type $[T] = \overline{T}$.
2. $S \in U$. Then S is closed, so $T = S$ and $[T] = [S] = [S]\tau$.
3. $S \notin U$ and $T \notin U$. Then $[S]\tau = S\tau = T = [T]$.

Thus, $\textsc{Tw}(t)$ is a well-formed term of type $[T]$. □

Theorem 6 (Soundness of Tw). *If a set of closed formulas Γ is satisfiable then $\textsc{Tw}(\Gamma)$ is satisfiable.*

Proof. Let $\mathfrak{M}$ be a model of Γ. We are going to construct an interpretation $\mathfrak{I}$ of $\textsc{Tw}(\Gamma)$ such that, for every type valuation π and every variable valuation φ in $\mathfrak{I}$, there exist a type valuation π' and a variable valuation φ' in the original model $\mathfrak{M}$, such that:

$$\mathfrak{I}_{\pi,\varphi}(\textsc{Tw}(t)) = \mathfrak{M}_{\pi',\varphi'}(t) \qquad \mathfrak{I}_{\pi,\varphi}(\textsc{Tw}(F)) = \mathfrak{M}_{\pi',\varphi'}(F)$$

for every term t and formula F occurring in Γ.

First of all, let us denote with $]\ [$ the "protection erasure" operation: for any protected sort $\bar{S}$, $]\bar{S}[\triangleq S$ and for any non-protected type T, $]\bar{T}[\triangleq T$. Obviously, for any type T in the initial signature, $][T][= T$. On the other hand, it is not generally true that $[]T[] = T$.

For any sort S in the initial signature, we define $\mathcal{D}^{\mathfrak{I}}_S \triangleq \mathcal{D}^{\mathfrak{M}}_S$ and $\mathcal{D}^{\mathfrak{I}}_{\bar{S}} \triangleq \mathcal{D}^{\mathfrak{M}}_S$. Then we give an interpretation to the function and predicate symbols in $\textsc{Tw}(\Gamma)$. Apart from the bridge function symbols, every symbol in the new signature originates from some initial symbol, as described above. Consider a function symbol f with the type signature $\mathbf{S}$ in the initial set Γ. In $\textsc{Tw}(\Gamma)$, it is replaced with the symbol $\bar{f}$ of type signature $[\mathbf{S}]$. Let $\mathbf{T}$ be a sort vector matched by $[\mathbf{S}]$. It is easy to see that the "unprotected" sort vector $]\mathbf{T}[$ is matched by $\mathbf{S}$. Thus we can define $\bar{f}^{\mathfrak{I}}_{\mathbf{T}} \triangleq f^{\mathfrak{M}}_{]\mathbf{T}[}$. Predicate symbols in $\textsc{Tw}(\Gamma)$ are interpreted in the same way. Finally, for every sort $T \in U$, the function symbols $\mathtt{to}_T$ and $\mathtt{from}_T$ are interpreted as identities. This finishes the definition of the interpretation $\mathfrak{I}$.

Now let us take a type valuation π and a variable valuation φ in $\mathfrak{I}$. We set $\pi' \triangleq \pi$ and $\varphi'(u : T) \triangleq \varphi(u : [T])$. By construction, $\mathcal{D}^{\mathfrak{I}}_{[T]\pi} = \mathcal{D}^{\mathfrak{I}}_{T\pi} = \mathcal{D}^{\mathfrak{M}}_{T\pi'}$.

Using this definitions, we can proceed to prove the two equalities above. The proof goes by induction on the structure of terms and formulas. We will consider three cases: a variable, a function application, and a universally quantified formula (other cases can be handled along the same lines).

1. Let $u : T$ be a variable occurring in Γ.

$$\mathfrak{I}_{\pi,\varphi}(\textsc{Tw}(u : T)) = \mathfrak{I}_{\pi,\varphi}(u : [T]) = \varphi(u : [T]) = \varphi'(u : T) = \mathfrak{M}_{\pi',\varphi'}(u : T)$$

2. Consider a term $t = f(t_1 : T_1, \ldots, t_n : T_n) : T$. The transformed term $\textsc{Tw}(t)$ is an application of the function symbol $\bar{f}$ to the transformed arguments $\textsc{Tw}(t_1), \ldots, \textsc{Tw}(t_n)$ with added, where necessary, functions $\mathtt{to}_\Box$ and $\mathtt{from}_\Box$. By the induction hypothesis and knowing that

– function symbols $\mathtt{to}_\square$ and $\mathtt{from}_\square$ are interpreted in $\mathfrak{I}$ as identities, and
– $\bar{f}$ on a vector sort $\mathbf{T}$ is interpreted in $\mathfrak{I}$ as f on $]\mathbf{T}[$ in $\mathfrak{M}$,

we can conclude that $\mathfrak{I}_{\pi,\varphi}(\mathrm{Tw}(t)) = \mathfrak{M}_{\pi',\varphi'}(t)$.

3. Consider a universally quantified formula $\forall(u:T)\,F$ from Γ. As we know, $\mathcal{D}^{\mathfrak{I}}_{[T]\pi} = \mathcal{D}^{\mathfrak{M}}_{T\pi'}$. Moreover, for any element $a \in \mathcal{D}^{\mathfrak{I}}_{[T]\pi}$, the variable valuation $\varphi[u:[T] \mapsto a]$ in $\mathfrak{I}$ produces the valuation $\varphi'[u:T \mapsto a]$ in $\mathfrak{M}$. We obtain:

$$\begin{aligned}\mathfrak{I}_{\pi,\varphi}(\mathrm{Tw}(\forall(u:T)\,F)) &= \bigwedge_{a\in\mathcal{D}^{\mathfrak{I}}_{[T]\pi}} \mathfrak{I}_{\pi,\varphi[u:[T]\mapsto a]}(\mathrm{Tw}(F)) \\ &= \bigwedge_{a\in\mathcal{D}^{\mathfrak{M}}_{T\pi'}} \mathfrak{M}_{\pi',\varphi'[u:T\mapsto a]}(F) = \mathfrak{M}_{\pi',\varphi'}(\forall(u:T)\,F)\end{aligned}$$

Thus, for every formula F in Γ, $\mathrm{Tw}(F)$ is satisfiedunder $\mathfrak{I}$. The bijection axioms for $\mathtt{to}_\square$ and $\mathtt{from}_\square$ are trivially satisfied, too, since both function symbols are interpreted as identities. □

Theorem 7 (Completeness of Tw). *For every set of closed formulas Γ, if $\mathrm{Tw}(\Gamma)$ is satisfiable then Γ is satisfiable.*

Proof. Let $\mathfrak{M}$ be a model of $\mathrm{Tw}(\Gamma)$. Due to the bijection axioms for bridge functions, we can assume without loss of generality that, for every sort $T \in U$, $\mathcal{D}^{\mathfrak{M}}_{T} = \mathcal{D}^{\mathfrak{M}}_{\overline{T}}$ and both $\mathtt{to}_T$ and $\mathtt{from}_T$ are interpreted in $\mathfrak{M}$ as identities.

We are going to construct an interpretation $\mathfrak{I}$ of Γ such that, for every type valuation π and every variable valuation φ in $\mathfrak{I}$, there exist a valuation π' and a variable valuation φ' in the original model $\mathfrak{M}$, such that:

$$\mathfrak{I}_{\pi,\varphi}(t) = \mathfrak{M}_{\pi',\varphi'}(\mathrm{Tw}(t)) \qquad\qquad \mathfrak{I}_{\pi,\varphi}(F) = \mathfrak{M}_{\pi',\varphi'}(\mathrm{Tw}(F))$$

for every term t and formula F occurring in Γ.

For any sort S in the signature of Γ, we define $\mathcal{D}^{\mathfrak{I}}_{S} \triangleq \mathcal{D}^{\mathfrak{M}}_{S}$. Then for any type T and its closed type instance $T\tau$, $\mathcal{D}^{\mathfrak{I}}_{T\tau} = \mathcal{D}^{\mathfrak{M}}_{[T]\tau}$. Indeed, if $T \in U$, then $\mathcal{D}^{\mathfrak{I}}_{T\tau} = \mathcal{D}^{\mathfrak{I}}_{T} = \mathcal{D}^{\mathfrak{M}}_{T} = \mathcal{D}^{\mathfrak{M}}_{\overline{T}} = \mathcal{D}^{\mathfrak{M}}_{[T]\tau}$, otherwise $[T] = T$.

Every function symbol f (predicate symbol p) of type signature $\mathbf{S}$ in Γ is interpreted on a closed sort vector $\mathbf{S}\tau$ in $\mathfrak{I}$ exactly as the corresponding symbol $\bar{f}$ (respectively, $\bar{p}$) on $[\mathbf{S}]\tau$ in $\mathfrak{M}$. This concludes the definition of $\mathfrak{I}$.

Now, consider a type valuation π and a variable valuation φ in $\mathfrak{I}$. We set $\pi' \triangleq \pi$ and $\varphi'(u:[T]) \triangleq \varphi(u:T)$ for every variable $u:T$ occurring in Γ. By Lemma 1, it suffices to define φ' on variables that occur in transformed formulas. Also recall that $\mathcal{D}^{\mathfrak{I}}_{T\pi} = \mathcal{D}^{\mathfrak{M}}_{[T]\pi} = \mathcal{D}^{\mathfrak{M}}_{[T]\pi'}$.

Using this definitions, we can proceed to prove the two equalities above. The proof goes by induction on the structure of terms and formulas. We will consider three cases: a variable, a function application, and a universally quantified formula (other cases can be handled along the same lines).

1. Let $u:T$ be a variable occurring in Γ. We obtain:

$$\mathfrak{I}_{\pi,\varphi}(u:T) = \varphi(u:T) = \varphi'(u:[T]) = \mathfrak{M}_{\pi',\varphi'}(u:[T]) = \mathfrak{M}_{\pi',\varphi'}(\mathrm{Tw}(u:T))$$

2. Consider a term $t = f(t_1 : T_1, \ldots, t_n : T_n) : T$. Let $\mathbf{S} = \langle S_1, \ldots, S_n, S \rangle$ be the type signature of f and τ a type substitution such that $\mathbf{S}\tau = \langle T_1, \ldots, T_n, T \rangle$. The term $\mathrm{Tw}(t)$ is of the form $\bar{f}(t'_1 :[S_1]\tau, \ldots, t'_n :[S_n]\tau) :[S]\tau$, possibly under $\mathtt{from}_T$, where every t'_i is $\mathrm{Tw}(t_i)$, possibly under $\mathtt{to}_{T_i}$. Knowing that:
   - bridge function symbols are interpreted in $\mathfrak{M}$ as identities, and
   - f is interpreted on $\mathbf{S}\tau$ in $\mathfrak{I}$ as $\bar{f}$ on $[\mathbf{S}]\tau$ in $\mathfrak{M}$,

   we can conclude that $\mathfrak{I}_{\pi,\varphi}(t) = \mathfrak{M}_{\pi',\varphi'}(\mathrm{Tw}(t))$.

3. Consider a universally quantified formula $\forall(u : T)\, F$. For any $a \in \mathcal{D}^{\mathfrak{I}}_{T\pi}$, the valuation $\varphi[u : T \mapsto a]$ in $\mathfrak{I}$ produces the valuation $\varphi'[u :[T] \mapsto a]$ in $\mathfrak{M}$.

$$\begin{aligned}\mathfrak{I}_{\pi,\varphi}(\forall(u : T)\, F) &= \bigwedge_{a \in \mathcal{D}^{\mathfrak{I}}_{T\pi}} \mathfrak{I}_{\pi,\varphi[u : T \mapsto a]}(F) \\ &= \bigwedge_{a \in \mathcal{D}^{\mathfrak{M}}_{[T]\pi'}} \mathfrak{M}_{\pi',\varphi'[u :[T] \mapsto a]}(\mathrm{Tw}(F)) \\ &= \mathfrak{M}_{\pi',\varphi'}(\forall(u :[T])\, \mathrm{Tw}(F)) = \mathfrak{M}_{\pi',\varphi'}(\mathrm{Tw}(\forall(u : T)\, F))\end{aligned}$$

Thus, for every formula F in Γ, F is satisfied under $\mathfrak{I}$. □

3.4 Decorated Terms

The Dec transformation converts a polymorphic problem with protected sorts into an equisatisfiable monomorphic problem. Roughly speaking, in order to preserve type information, it decorates every term with its type, which itself is transformed to a term of a special sort.

First of all, we introduce three fresh sort constants U, D, and T. The first one is assigned to undecorated terms, the second one to decorated terms, and the third one to the terms representing types. To transform the type signatures of function and predicate symbols, we use the following operations on types:

$$[T]^- \triangleq \begin{cases} T & \text{if } T \text{ is protected,} \\ \mathsf{D} & \text{otherwise} \end{cases} \qquad [T]^+ \triangleq \begin{cases} T & \text{if } T \text{ is protected,} \\ \mathsf{U} & \text{otherwise} \end{cases}$$

Now, the signature of the resulting theory is defined as follows:

1. The set of type constructors is extended with U, D, T.
2. We replace every function symbol f of type signature $\langle S_1, \ldots, S_n, S \rangle$ with a symbol $\hat{f}$ with the monomorphic type signature $\langle [S_1]^-, \ldots, [S_n]^-, [S]^+ \rangle$.
3. We replace every predicate symbol p of type signature $\langle S_1, \ldots, S_n \rangle$ with a symbol $\hat{p}$ with the monomorphic type signature $\langle [S_1]^-, \ldots, [S_n]^- \rangle$.
4. For every variable symbol u and type T, we add a new variable symbol u_T.
5. For every type variable $\alpha \in \mathbb{V}_{\mathrm{T}}$, we add a new variable symbol v^α.
6. For every type constructor $\mathsf{F} \in \mathbb{F}_{\mathrm{T}}$, we add a new function symbol $\mathtt{F}$ of the same arity and with type signature $\langle \mathsf{T}, \ldots, \mathsf{T}, \mathsf{T} \rangle$.
7. We add a new "decoration" function symbol $\mathtt{deco} : \langle \mathsf{T}, \mathsf{U}, \mathsf{D} \rangle$.

The DEC transformation applies to non-protected types, translating them to terms of type T:

$$\textsc{Dec}(\alpha) \triangleq v^{\alpha} : \mathsf{T} \qquad \textsc{Dec}(\mathsf{F}(T_1, \ldots, T_n)) \triangleq \mathtt{F}(\textsc{Dec}(T_1), \ldots, \textsc{Dec}(T_n)) : \mathsf{T}$$

In the next definition, $\mathbf{t}$ stands for a vector of terms, $\bar{S}$ for a protected sort, and T for a non-protected type. The DEC transformation applies to terms:

$$\begin{aligned}
\textsc{Dec}(u : \bar{S}) &\triangleq u_{\bar{S}} : \bar{S} \\
\textsc{Dec}(u : T) &\triangleq \mathtt{deco}(\textsc{Dec}(T), u_T : \mathsf{U}) : \mathsf{D} \\
\textsc{Dec}(f(\mathbf{t}) : \bar{S}) &\triangleq \hat{f}(\textsc{Dec}(\mathbf{t})) : \bar{S} \\
\textsc{Dec}(f(\mathbf{t}) : T) &\triangleq \mathtt{deco}(\textsc{Dec}(T), \hat{f}(\textsc{Dec}(\mathbf{t})) : \mathsf{U}) : \mathsf{D}
\end{aligned}$$

and formulas (here, $\{\alpha_1, \ldots, \alpha_m\} = \mathrm{FV}_{\mathrm{T}}(H)$):

$$\begin{aligned}
\textsc{Dec}(p(\mathbf{t})) &\triangleq \hat{p}(\textsc{Dec}(\mathbf{t})) & \textsc{Dec}(\neg F) &\triangleq \neg\, \textsc{Dec}(F) \\
\textsc{Dec}(t_1 \approx t_2) &\triangleq \textsc{Dec}(t_1) \approx \textsc{Dec}(t_2) & \textsc{Dec}(\forall(u : \bar{S})F) &\triangleq \forall(u_{\bar{S}} : \bar{S})\, \textsc{Dec}(F) \\
\textsc{Dec}(F \wedge G) &\triangleq \textsc{Dec}(F) \wedge \textsc{Dec}(G) & \textsc{Dec}(\forall(u : T)F) &\triangleq \forall(u_T : \mathsf{U})\, \textsc{Dec}(F)
\end{aligned}$$

$$\textsc{Dec}^{\circ}(H) \triangleq \forall(v^{\alpha_1} : \mathsf{T}) \ldots \forall(v^{\alpha_m} : \mathsf{T})\, \textsc{Dec}(H)$$

On the running example, assuming $U = \{\mathsf{I}\}$, the transformations PAR and DEC$^{\circ}$ produce the following monomorphic formulas (for the sake of clarity, we "forget" the variable subscripts added by PAR):

$$\begin{aligned}
&\forall(v^{\alpha} : \mathsf{T})\, \forall(m_{\mathsf{M}(\alpha,\mathsf{I})} : \mathsf{U})\, \forall(c_{\alpha} : \mathsf{U})\, \mathtt{get}_{[\mathsf{I}/\beta]}(\mathtt{deco}(\mathtt{M}(v^{\alpha}, \mathtt{I}), \\
&\quad \mathtt{set}_{[\mathsf{I}/\beta]}(\mathtt{deco}(\mathtt{M}(v^{\alpha}, \mathtt{I}), m_{\mathsf{M}(\alpha,\mathsf{I})}), \mathtt{deco}(v^{\alpha}, c_{\alpha}), 6)), \mathtt{deco}(v^{\alpha}, c_{\alpha})) * 7 \approx 42
\end{aligned}$$

$$\begin{aligned}
&\forall(m_{\mathsf{M}(\mathsf{I},\mathsf{I})} : \mathsf{U})\, \forall(c_{\bar{\mathsf{I}}} : \bar{\mathsf{I}})\, \mathtt{get}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}(\mathtt{deco}(\mathtt{M}(\mathtt{I}, \mathtt{I}), \\
&\qquad\qquad \mathtt{set}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}(\mathtt{deco}(\mathtt{M}(\mathtt{I}, \mathtt{I}), m_{\mathsf{M}(\mathsf{I},\mathsf{I})}), c_{\bar{\mathsf{I}}}, 6)), c_{\bar{\mathsf{I}}}) * 7 \approx 42
\end{aligned}$$

whereas TW and DEC$^{\circ}$ produce:

$$\begin{aligned}
&\forall(v^{\alpha} : \mathsf{T})\, \forall(m_{\mathsf{M}(\alpha,\mathsf{I})} : \mathsf{U})\, \forall(c_{\alpha} : \mathsf{U})\, \mathtt{from}_{\mathsf{I}}\big(\mathtt{deco}\big(\mathtt{I}, \mathtt{get}(\mathtt{deco}(\mathtt{M}(v^{\alpha}, \mathtt{I}), \\
&\qquad \mathtt{set}(\mathtt{deco}(\mathtt{M}(v^{\alpha}, \mathtt{I}), m_{\mathsf{M}(\alpha,\mathsf{I})}), \mathtt{deco}(v^{\alpha}, c_{\alpha}), \mathtt{deco}(\mathtt{I}, \mathtt{to}_{\mathsf{I}}(6)))), \\
&\qquad\qquad \mathtt{deco}(v^{\alpha}, c_{\alpha}))\big)\big) * 7 \approx 42
\end{aligned}$$

The second axiom of bridge functions $\mathtt{to}_{\mathsf{I}}$ and $\mathtt{from}_{\mathsf{I}}$ becomes

$$\forall(u_{\mathsf{I}} : \mathsf{U})\, \mathtt{deco}(\mathtt{I}, \mathtt{to}_{\mathsf{I}}(\mathtt{from}_{\mathsf{I}}(\mathtt{deco}(\mathtt{I}, u_{\mathsf{I}})))) \approx \mathtt{deco}(\mathtt{I}, u_{\mathsf{I}})$$

Due to the outer application of `deco` on the both sides of equality, our translation is sound even when we protect finite types, such as booleans. Without this additional decoration (as in [5, Eq. (8)]), the finiteness of a protected sort implies the finiteness of the whole sort U.

Lemma 8. *For every term t of type T, $\textsc{Dec}(t)$ is a well-formed monomorphic term of type $[T]^-$. For every formula F, $\textsc{Dec}(F)$ is a well-formed monomorphic formula. Also,* $\mathrm{FV}(\textsc{Dec}(t)) = \{u_T : [T]^+ \mid u : T \in \mathrm{FV}(t)\} \cup \{v^\alpha : \mathsf{T} \mid \alpha \in \mathrm{FV}_\mathrm{T}(t)\}$ *and* $\mathrm{FV}(\textsc{Dec}(F)) = \{u_T : [T]^+ \mid u : T \in \mathrm{FV}(F)\} \cup \{v^\alpha : \mathsf{T} \mid \alpha \in \mathrm{FV}_\mathrm{T}(F)\}$. *For every closed formula F, $\textsc{Dec}^\circ(F)$ is a well-formed closed monomorphic formula.*

Proof. The last claim easily follows from the second one. The proof of the first two statements goes by induction on the structure of terms and formulas. The only interesting case is the application of a function symbol (predicate symbols are treated in the same way).

Consider a term $t = f(t_1 : T_1, \ldots, t_n : T_n) : T$. Let S_i denote the i-th sort in the type signature of f. We have two possible cases: either S_i and T_i are protected and $S_i = T_i$, or both S_i and T_i are non-protected, by the definition of matching with protected types. Therefore, $[T_i]^- = [S_i]^-$ for every $i \in [1, n]$.

Now, let S be the type of the value in the type signature of f. By the same argument, either S and T are protected and $S = T$, or both S and T are non-protected. Then $[T]^+ = [S]^+$ and the term $t' = \hat{f}(\textsc{Dec}(t_1), \ldots, \textsc{Dec}(t_n)) : [T]^+$ is well-formed.

Finally, if T is protected, then $\textsc{Dec}(t) = t'$ and $[T]^- = T = [T]^+$. Notice that the type T does not contain type variables. Otherwise, if T is non-protected, then $[T]^+ = \mathsf{U}$ and $\textsc{Dec}(t) = \mathtt{deco}(\textsc{Dec}(T), t') : \mathsf{D}$ is a well-formed term of type $[T]^-$. Furthermore, every type variable α occurring in T appears as a free variable $v^\alpha : \mathsf{T}$ in the first argument of $\mathtt{deco}$ in $\textsc{Dec}(t)$. □

Theorem 8 (Soundness of $\textsc{Dec}$). *If a set of closed formulas with protected sorts Γ is satisfiable then $\textsc{Dec}^\circ(\Gamma)$ is satisfiable.*

Proof. Let $\mathfrak{M}$ be a model of Γ. We are going to construct an interpretation $\mathfrak{I}$ of $\textsc{Dec}^\circ(\Gamma)$ such that, for every variable valuation φ in $\mathfrak{I}$, there exist a type valuation π' and a variable valuation φ' in the original model $\mathfrak{M}$, such that:

$$\mathfrak{I}_\varphi(\textsc{Dec}(t : T)) = \langle T\pi', \mathfrak{M}_{\pi',\varphi'}(t)\rangle \qquad \mathfrak{I}_\varphi(\textsc{Dec}(F)) = \mathfrak{M}_{\pi',\varphi'}(F)$$

for every term t and formula F occurring in Γ. We do not need to consider type valuations in $\mathfrak{I}$, since the result of $\textsc{Dec}^\circ$ is entirely monomorphic. As soon as these two equalities are proved, it is easy to show that for every closed $H \in \Gamma$, the interpretation $\mathfrak{I}$ satisfies $\textsc{Dec}^\circ(H)$:

$$\mathfrak{I}(\textsc{Dec}^\circ(H)) = \bigwedge_\varphi \mathfrak{I}_\varphi(\textsc{Dec}(H)) = \bigwedge_\varphi \mathfrak{M}_{\pi',\varphi'}(H) = \top$$

For every sort S in the initial signature, protected or not, we keep its domain, pairing its elements with S:

$$\mathcal{D}^\mathfrak{I}_S \triangleq \{S\} \times \mathcal{D}^\mathfrak{M}_S$$

The domain of D is defined as a dependent sum over non-protected sorts:

$$\mathcal{D}^\mathfrak{I}_\mathsf{D} \triangleq \{\, \langle S, c\rangle \mid S \in \mathcal{T}(\mathbb{F}_\mathrm{T}),\ c \in \mathcal{D}^\mathfrak{M}_S \,\}$$

The domain of U is the set of all functions that map every non-protected sort S in $\mathcal{T}(\mathbb{F}_{\mathrm{T}})$ to an element of $\mathcal{D}^{\mathfrak{M}}_S$:

$$\mathcal{D}^{\mathfrak{I}}_{\mathsf{U}} \triangleq \{\, S \in \mathcal{T}(\mathbb{F}_{\mathrm{T}}) \longmapsto c \in \mathcal{D}^{\mathfrak{M}}_S \,\}$$

Finally, the domain of T is the set of all non-protected sorts: $\mathcal{D}^{\mathfrak{I}}_{\mathsf{T}} \triangleq \mathcal{T}(\mathbb{F}_{\mathrm{T}})$.

Then we give an interpretation to the function and predicate symbols in $\textsc{Dec}^{\circ}(\Gamma)$. Let f be a function symbol in Γ of type signature $\mathbf{S} = \langle S_1, \ldots, S_n, S\rangle$. The symbol $\hat{f}$ has the type signature $\mathbf{S}' = \langle [S_1]^-, \ldots, [S_n]^-, [S]^+\rangle$ and this is the only sort vector to interpret $\hat{f}$ on, since it is monomorphic. Firstly, suppose that S is protected. Then we define:

$$\hat{f}^{\mathfrak{I}}_{\mathbf{S}'}(\langle T_1, c_1\rangle, \ldots, \langle T_n, c_n\rangle) \triangleq \begin{cases} \langle S, f^{\mathfrak{M}}_{\langle T_1,\ldots,T_n,S\rangle}(c_1,\ldots,c_n)\rangle & \text{if } \mathbf{S} \text{ matches } \langle T_1,\ldots,T_n,S\rangle, \\ \langle S, e_S\rangle & \text{otherwise,} \end{cases}$$

where e_S is an arbitrary but fixed element of $\mathcal{D}^{\mathfrak{M}}_S$. Otherwise, if S is non-protected, we set:

$$\hat{f}^{\mathfrak{I}}_{\mathbf{S}'}(\langle T_1, c_1\rangle, \ldots, \langle T_n, c_n\rangle) \triangleq \lambda T \in \mathcal{T}(\mathbb{F}_{\mathrm{T}})\,. \begin{cases} f^{\mathfrak{M}}_{\langle T_1,\ldots,T_n,T\rangle}(c_1,\ldots,c_n) & \text{if } \mathbf{S} \text{ matches } \langle T_1,\ldots,T_n,T\rangle, \\ e_T & \text{otherwise,} \end{cases}$$

where e_T is an arbitrary but fixed element of $\mathcal{D}^{\mathfrak{M}}_T$. Similarly, let p be a predicate symbol in Γ of type signature $\mathbf{S} = \langle S_1, \ldots, S_n\rangle$. The symbol $\hat{p}$ has the type signature $\mathbf{S}' = \langle [S_1]^-, \ldots, [S_n]^-\rangle$, and we set:

$$\hat{p}^{\mathfrak{I}}_{\mathbf{S}'}(\langle T_1, c_1\rangle, \ldots, \langle T_n, c_n\rangle) \triangleq \begin{cases} p^{\mathfrak{M}}_{\langle T_1,\ldots,T_n\rangle}(c_1,\ldots,c_n) & \text{if } \mathbf{S} \text{ matches } \langle T_1,\ldots,T_n\rangle, \\ \top & \text{otherwise.} \end{cases}$$

The symbol `deco` is interpreted as follows:

$$\mathtt{deco}^{\mathfrak{I}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D}\rangle}(T, h) \triangleq \langle T, h(T)\rangle$$

Finally, the type-representing function symbols are interpreted exactly as the original type constructors:

$$\mathsf{F}^{\mathfrak{I}}_{\langle \mathsf{T},\ldots,\mathsf{T}\rangle}(T_1,\ldots,T_n) \triangleq \mathsf{F}(T_1,\ldots,T_n)$$

Now, consider a variable valuation φ in $\mathfrak{I}$. For every type variable α, we set $\alpha\pi' \triangleq \varphi(v^{\alpha} : \mathsf{T})$. Every type in $\mathcal{D}^{\mathfrak{I}}_{\mathsf{T}}$ is a non-protected sort, so π' is a type valuation. Moreover, for every non-protected type T, we have $\mathfrak{I}_{\varphi}(\textsc{Dec}(T)) = T\pi'$.

Then, for any variable $u : T$ in Γ, we set:

$$\varphi'(u : T) \triangleq \begin{cases} c & \text{if } T \text{ is protected and } \varphi(u_T : T) = \langle T, c\rangle, \\ \varphi(u_T : \mathsf{U})(T\pi') & \text{otherwise.} \end{cases}$$

We proceed to prove the two main equalities, and, once again, the proof goes by induction on the structure of terms and formulas. We consider four cases: a variable, a function application, an equality, and a universally quantified formula.

1. Let $u:T$ be a variable in Γ. If T is protected, then $\mathfrak{I}_\varphi(\textsc{Dec}(u:T)) = \mathfrak{I}_\varphi(u_T:T) = \varphi(u_T:T) = \langle T, \varphi'(u:T)\rangle = \langle T, \mathfrak{M}_{\pi',\varphi'}(u:T)\rangle$. Otherwise, if T is a non-protected type, we have:

$$\begin{aligned}\mathfrak{I}_\varphi(\textsc{Dec}(u:T)) = \mathfrak{I}_\varphi(\mathtt{deco}(\textsc{Dec}(T), u_T:\mathsf{U}):\mathsf{D}) = \\ \mathtt{deco}^{\mathfrak{I}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D}\rangle}(\mathfrak{I}_\varphi(\textsc{Dec}(T)), \mathfrak{I}_\varphi(u_T:\mathsf{U})) = \langle T\pi', \varphi(u_T:\mathsf{U})(T\pi')\rangle = \\ \langle T\pi', \varphi'(u:T)\rangle = \langle T\pi', \mathfrak{M}_{\pi',\varphi'}(u:T)\rangle\end{aligned}$$

2. Consider a term $t = f(t_1:T_1,\ldots,t_n:T_n):T$. Let $\mathbf{S} = \langle S_1,\ldots,S_n,S\rangle$ be the type signature of f and $\mathbf{S}' = \langle [S_1]^-,\ldots,[S_n]^-,[S]^+\rangle$. If T is protected, then

$$\begin{aligned}\mathfrak{I}_\varphi(\textsc{Dec}(t)) = \hat{f}^{\mathfrak{I}}_{\mathbf{S}'}\big(\mathfrak{I}_\varphi(\textsc{Dec}(t_1)),\ldots,\mathfrak{I}_\varphi(\textsc{Dec}(t_n))\big) = \\ \hat{f}^{\mathfrak{I}}_{\mathbf{S}'}\big(\langle T_1\pi', \mathfrak{M}_{\pi',\varphi'}(t_1)\rangle,\ldots,\langle T_n\pi', \mathfrak{M}_{\pi',\varphi'}(t_n)\rangle\big) = \\ \langle T, f^{\mathfrak{M}}_{\langle T_1\pi',\ldots,T_n\pi',T\rangle}(\mathfrak{M}_{\pi',\varphi'}(t_1),\ldots,\mathfrak{M}_{\pi',\varphi'}(t_n))\rangle = \langle T\pi', \mathfrak{M}_{\pi',\varphi'}(t)\rangle\end{aligned}$$

   Now, suppose that T is non-protected. We obtain:

$$\begin{aligned}\mathfrak{I}_\varphi(\textsc{Dec}(t)) = \mathfrak{I}_\varphi(\mathtt{deco}(\textsc{Dec}(T), \hat{f}(\textsc{Dec}(t_1),\ldots,\textsc{Dec}(t_n)):\mathsf{U}):\mathsf{D}) = \\ \mathtt{deco}^{\mathfrak{I}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D}\rangle}\big(T\pi', \hat{f}^{\mathfrak{I}}_{\mathbf{S}'}\big(\mathfrak{I}_\varphi(\textsc{Dec}(t_1)),\ldots,\mathfrak{I}_\varphi(\textsc{Dec}(t_n))\big)\big) = \\ \langle T\pi', \hat{f}^{\mathfrak{I}}_{\mathbf{S}'}\big(\langle T_1\pi', \mathfrak{M}_{\pi',\varphi'}(t_1)\rangle,\ldots,\langle T_n\pi', \mathfrak{M}_{\pi',\varphi'}(t_n)\rangle\big)(T\pi')\rangle = \\ \langle T\pi', f^{\mathfrak{M}}_{\langle T_1\pi',\ldots,T_n\pi',T\pi'\rangle}(\mathfrak{M}_{\pi',\varphi'}(t_1),\ldots,\mathfrak{M}_{\pi',\varphi'}(t_n))\rangle = \langle T\pi', \mathfrak{M}_{\pi',\varphi'}(t)\rangle\end{aligned}$$

3. Consider an equality formula $t_1 \approx t_2$. Let T be the type of t_1 and t_2. By induction hypothesis, $\mathfrak{I}_\varphi(\textsc{Dec}(t_1)) = \langle T\pi', \mathfrak{M}_{\pi',\varphi'}(t_1)\rangle$ and $\mathfrak{I}_\varphi(\textsc{Dec}(t_2)) = \langle T\pi', \mathfrak{M}_{\pi',\varphi'}(t_2)\rangle$. Obviously, these two are equal if and only if $\mathfrak{M}_{\pi',\varphi'}(t_1)$ and $\mathfrak{M}_{\pi',\varphi'}(t_2)$ are equal.
4. Consider a universally quantified formula $\forall(u:S)\,F$, where S is protected. By construction, $\mathcal{D}^{\mathfrak{I}}_S = \{S\}\times\mathcal{D}^{\mathfrak{M}}_S$. Let us take an arbitrary element $\langle S, c\rangle \in \mathcal{D}^{\mathfrak{I}}_S$. According to the definition above, the variable valuation $\varphi[u_S : S \mapsto \langle S, c\rangle]$ will produce in $\mathfrak{M}$ the same type valuation π' and the variable valuation $\varphi'[u:S\mapsto c]$. We obtain:

$$\begin{aligned}\mathfrak{I}_\varphi(\textsc{Dec}(\forall(u:S)\,F)) &= \bigwedge_{\langle S,c\rangle\in\mathcal{D}^{\mathfrak{I}}_S} \mathfrak{I}_{\varphi[u_S:S\mapsto\langle S,c\rangle]}(\textsc{Dec}(F)) \\ &= \bigwedge_{c\in\mathcal{D}^{\mathfrak{M}}_S} \mathfrak{M}_{\pi',\varphi'[u:S\mapsto c]}(F) = \mathfrak{M}_{\pi',\varphi'}(\forall(u:S)\,F)\end{aligned}$$

5. Consider a formula $\forall(u:T)\,F$, where T is a non-protected type. After the transformation, the variable symbol u_T acquires the type U. Any element of

$\mathcal{D}^{\mathfrak{I}}_{\mathsf{U}}$ is a function that maps any non-protected sort S to some element of $\mathcal{D}^{\mathfrak{M}}_{S}$. However, we are only interested in the value of this function on $T\pi'$, since for any two functions $h, h' \in \mathcal{D}^{\mathfrak{I}}_{\mathsf{U}}$ that coincide on $T\pi'$, $\mathfrak{I}_{\varphi[u_T : \mathsf{U} \mapsto h]}(\textsc{Dec}(F)) = \mathfrak{I}_{\varphi[u_T : \mathsf{U} \mapsto h']}(\textsc{Dec}(F))$. Indeed, $u_T : \mathsf{U}$ can only occur in $\textsc{Dec}(F)$ inside a term $\mathtt{deco}(\textsc{Dec}(T), u_T : \mathsf{U}) : \mathsf{D}$. Since $\textsc{Dec}(F)$ does not contain quantifiers over T, the term $\textsc{Dec}(T)$ will always evaluate to $T\pi'$. Therefore, the term $\mathtt{deco}(\textsc{Dec}(T), u_T : \mathsf{U}) : \mathsf{D}$ will evaluate to $\langle T\pi', h(T\pi')\rangle$ in both cases.
For any $c \in \mathcal{D}^{\mathfrak{M}}_{T\pi'}$, let h_c denote an arbitrary but fixed function in $\mathcal{D}^{\mathfrak{I}}_{\mathsf{U}}$ that maps $T\pi'$ to c. The valuation $\varphi[u_T : \mathsf{U} \mapsto h_c]$ produces in $\mathfrak{M}$ the same type valuation π' and the variable valuation $\varphi'[u : T \mapsto c]$. We obtain:

$$\begin{aligned}
\mathfrak{I}_{\varphi}(\textsc{Dec}(\forall(u : T)\, F)) &= \bigwedge_{h \in \mathcal{D}^{\mathfrak{I}}_{\mathsf{U}}} \mathfrak{I}_{\varphi[u_T : \mathsf{U} \mapsto h]}(\textsc{Dec}(F)) \\
&= \bigwedge_{c \in \mathcal{D}^{\mathfrak{M}}_{T\pi'}} \mathfrak{I}_{\varphi[u_T : \mathsf{U} \mapsto h_c]}(\textsc{Dec}(F)) \\
&= \bigwedge_{c \in \mathcal{D}^{\mathfrak{M}}_{T\pi'}} \mathfrak{M}_{\pi', \varphi'[u : T \mapsto c]}(F) = \mathfrak{M}_{\pi', \varphi'}(\forall(u : T)\, F)
\end{aligned}$$

Thus, for every formula F in Γ, $\textsc{Dec}^{\circ}(F)$ is satisfied under $\mathfrak{I}$. □

Theorem 9 (Completeness of $\textsc{Dec}$**).** *For every set of closed formulas with protected sorts Γ, if $\textsc{Dec}^{\circ}(\Gamma)$ is satisfiable then Γ is satisfiable.*

Proof. Let $\mathfrak{M}$ be a model of $\textsc{Dec}^{\circ}(\Gamma)$. We are going to construct an interpretation $\mathfrak{I}$ of Γ such that, for every type valuation π and every variable valuation φ in $\mathfrak{I}$, there exists a variable valuation φ' in the original model $\mathfrak{M}$ such that:

$$\mathfrak{I}_{\pi,\varphi}(t) = \mathfrak{M}_{\varphi'}(\textsc{Dec}(t)) \qquad\qquad \mathfrak{I}_{\pi,\varphi}(F) = \mathfrak{M}_{\varphi'}(\textsc{Dec}(F))$$

for every term t and formula F occurring in Γ. Since $\textsc{Dec}^{\circ}(F)$ is the universal closure of $\textsc{Dec}(F)$, the second equality suffices to show $\mathfrak{I}_{\pi}(F)$ for every $F \in \Gamma$.

For any sort S in the signature of Γ, we set:

$$\mathcal{D}^{\mathfrak{I}}_{S} \triangleq \begin{cases} \mathcal{D}^{\mathfrak{M}}_{S} & \text{if } S \text{ is protected,} \\ \mathtt{deco}^{\mathfrak{M}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D}\rangle}(\mathfrak{M}(\textsc{Dec}(S)), \mathcal{D}^{\mathfrak{M}}_{\mathsf{U}}) & \text{otherwise,} \end{cases}$$

In the second case, we speak about the image of $\mathtt{deco}^{\mathfrak{M}}$ on the domain of U. We omit the variable valuation when evaluating $\textsc{Dec}(S)$, as this term does not contain variables. Notice that for every non-protected sort S, $\mathcal{D}^{\mathfrak{I}}_{S} \subseteq \mathcal{D}^{\mathfrak{M}}_{\mathsf{D}}$.

Consider a function symbol f in Γ of type signature $\mathbf{S} = \langle S_1, \ldots, S_n, S\rangle$. Let $\mathbf{S}'$ stand for the type signature of $\hat{f}$, $\langle [S_1]^-, \ldots, [S_n]^-, [S]^+\rangle$. The symbol f is then interpreted in $\mathfrak{I}$ on a sort vector $\mathbf{S}\tau$ as follows:

$$f^{\mathfrak{I}}_{\mathbf{S}\tau}(\mathbf{c}) \triangleq \begin{cases} \hat{f}^{\mathfrak{M}}_{\mathbf{S}'}(\mathbf{c}) & \text{if } S \text{ is protected,} \\ \mathtt{deco}^{\mathfrak{M}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D}\rangle}(\mathfrak{M}(\textsc{Dec}(S\tau)), \hat{f}^{\mathfrak{M}}_{\mathbf{S}'}(\mathbf{c})) & \text{otherwise.} \end{cases}$$

Notice that every argument c_i from $\mathbf{c}$ belongs either to $\mathcal{D}^{\mathfrak{M}}_{S_i}$ (if the sort S_i is protected) or to $\mathcal{D}^{\mathfrak{M}}_{\mathsf{D}}$ (if it is not), and thus the interpretation is well-defined. Furthermore, every predicate symbol p of type signature $\mathbf{S} = \langle S_1, \ldots, S_n \rangle$ is interpreted in $\mathfrak{I}$ on $\mathbf{S}\tau$ as $\hat{p}^{\mathfrak{M}}_{\langle [S_1]^-, \ldots, [S_n]^- \rangle}$ on the same arguments. This concludes the definition of the interpretation $\mathfrak{I}$.

Now, consider some type valuation π and a variable valuation φ in $\mathfrak{I}$. For every variable occurring in $\textsc{Dec}^{\circ}(\Gamma)$, we define (assuming that S is a protected sort and T is a non-protected type):

$$\begin{aligned} \varphi'(v^{\alpha} : \mathsf{T}) &\triangleq \mathfrak{M}(\textsc{Dec}(\alpha\pi)) \\ \varphi'(u_S : S) &\triangleq \varphi(u : S) \\ \varphi'(u_T : \mathsf{U}) &\triangleq b_{\varphi(u:T)} \end{aligned}$$

where, for any $a \in \mathcal{D}^{\mathfrak{I}}_{T\pi}$, b_a is an arbitrary but fixed element of $\mathcal{D}^{\mathfrak{M}}_{\mathsf{U}}$ such that

$$\mathtt{deco}^{\mathfrak{M}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D} \rangle}(\mathfrak{M}(\textsc{Dec}(T\pi)), b_a) = a$$

By definition of $\mathcal{D}^{\mathfrak{I}}_{T\pi}$, there exists at least one such element. Notice that as T is non-protected, $T\pi$ is non-protected, too. Also, $\mathfrak{M}_{\varphi'}(\textsc{Dec}(T)) = \mathfrak{M}(\textsc{Dec}(T\pi))$.

The proof of the equalities above goes by induction on the structure of terms and formulas. We will consider three cases: a variable, a function application, and a universally quantified formula.

1. Let $u : T$ be a variable occurring in Γ. If T is a protected sort, we have $\mathfrak{I}_{\pi,\varphi}(u : T) = \varphi(u : T) = \varphi'(u_T : T) = \mathfrak{M}_{\varphi'}(\textsc{Dec}(u : T))$. If T is a non-protected type, we obtain:

$$\begin{aligned} \mathfrak{I}_{\pi,\varphi}(u : T) = \varphi(u : T) &= \mathtt{deco}^{\mathfrak{M}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D} \rangle}(\mathfrak{M}(\textsc{Dec}(T\pi)), b_{\varphi(u:T)}) \\ &= \mathtt{deco}^{\mathfrak{M}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D} \rangle}(\mathfrak{M}_{\varphi'}(\textsc{Dec}(T)), \varphi'(u_T : \mathsf{U})) \\ &= \mathfrak{M}_{\varphi'}(\mathtt{deco}(\textsc{Dec}(T), u_T : \mathsf{U}) : \mathsf{D}) = \mathfrak{M}_{\varphi'}(\textsc{Dec}(u : T)) \end{aligned}$$

2. Consider a term $t = f(t_1 : T_1, \ldots, t_n : T_n) : T$ in Γ. Let $\mathbf{S} = \langle S_1, \ldots, S_n, S \rangle$ be the type signature of f and $\mathbf{S}'$ be the type signature of $\hat{f}$. If S is a protected sort, then f is interpreted on $\langle T_1, \ldots, T_n, T \rangle\pi$ in $\mathfrak{I}$ exactly as $\hat{f}$ on $\mathbf{S}'$ in $\mathfrak{M}$, thus $\mathfrak{I}_{\pi,\varphi}(t) = \mathfrak{M}_{\varphi'}(\textsc{Dec}(t))$. Assume that S is non-protected. Denoting the list of arguments with $\mathbf{t}$, we get:

$$\begin{aligned} \mathfrak{I}_{\pi,\varphi}(t) = f^{\mathfrak{I}}_{\mathbf{T}\pi}(\mathfrak{I}_{\pi,\varphi}(\mathbf{t})) = \mathtt{deco}^{\mathfrak{M}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D} \rangle}(\mathfrak{M}(\textsc{Dec}(T\pi)), \hat{f}^{\mathfrak{M}}_{\mathbf{S}'}(\mathfrak{I}_{\pi,\varphi}(\mathbf{t}))) = \\ \mathtt{deco}^{\mathfrak{M}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D} \rangle}(\mathfrak{M}_{\varphi'}(\textsc{Dec}(T)), \hat{f}^{\mathfrak{M}}_{\mathbf{S}'}(\mathfrak{M}_{\varphi'}(\textsc{Dec}(\mathbf{t})))) = \mathfrak{M}_{\varphi'}(\textsc{Dec}(t)) \end{aligned}$$

3. Consider a universal formula $\forall(u : T)\, F$. The case of protected T is trivial, so we assume that T is non-protected. Let b and b' be two elements of $\mathcal{D}^{\mathfrak{M}}_{\mathsf{U}}$ such that $\mathtt{deco}^{\mathfrak{M}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D} \rangle}(\mathfrak{M}(\textsc{Dec}(T\pi)), b) = \mathtt{deco}^{\mathfrak{M}}_{\langle \mathsf{T},\mathsf{U},\mathsf{D} \rangle}(\mathfrak{M}(\textsc{Dec}(T\pi)), b')$. We can show that $\mathfrak{M}_{\varphi'[u_T : \mathsf{U} \mapsto b]}(\textsc{Dec}(F)) = \mathfrak{M}_{\varphi'[u_T : \mathsf{U} \mapsto b']}(\textsc{Dec}(F))$. Indeed, the variable $u_T : \mathsf{U}$ can only occur in $\textsc{Dec}(F)$ in terms $\mathtt{deco}(\textsc{Dec}(T), u_T : \mathsf{U}) : \mathsf{D}$.

Since $\textsc{Dec}(F)$ does not contain quantifiers over T, the term $\textsc{Dec}(T)$ will always evaluate to $\mathfrak{M}(\textsc{Dec}(T\pi))$.
Let a be an arbitrary element of $\mathcal{D}^{\mathfrak{I}}_{T\pi}$. The type valuation π and the variable valuation $\varphi[u:T \mapsto a]$ produce in $\mathfrak{M}$ the variable valuation $\varphi'[u_T : \mathsf{U} \mapsto b_a]$.

$$\begin{aligned}
\mathfrak{I}_{\pi,\varphi}(\forall(u:T)\,F) &= \bigwedge_{a \in \mathcal{D}^{\mathfrak{I}}_{T\pi}} \mathfrak{I}_{\pi,\varphi[u:T \mapsto a]}(F) \\
&= \bigwedge_{a \in \mathcal{D}^{\mathfrak{I}}_{T\pi}} \mathfrak{M}_{\varphi'[u_T : \mathsf{U} \mapsto b_a]}(\textsc{Dec}(F)) \\
&= \bigwedge_{b \in \mathcal{D}^{\mathfrak{M}}_{\mathsf{U}}} \mathfrak{M}_{\varphi'[u_T : \mathsf{U} \mapsto b]}(\textsc{Dec}(F)) \\
&= \mathfrak{M}_{\varphi'}(\forall(u_T : \mathsf{U})\,\textsc{Dec}(F)) = \mathfrak{M}_{\varphi'}(\textsc{Dec}(\forall(u:T)\,F))
\end{aligned}$$

Thus, for every formula F in Γ, F is satisfied under $\mathfrak{I}$. □

3.5 Explicit Polymorphism

The Exp transformation is similar to Dec except that instead of attaching an explicit type annotation to every term, we add type-representing arguments to polymorphic symbols. This allows for much lighter modifications in the original problem. However, the method is only sound on problems that admit a model where every non-protected sort has an infinite domain.

We introduce fresh sort constants U and T. The first one replaces non-protected types and the second one, as in Dec, is the sort of type-representing terms. For any type T, we define $[T]$ to be T if T is protected, and U otherwise. Then Exp modifies the signature of a transformed theory in the following way:

1. The set of type constructors is extended with U and T.
2. Let f be a function symbol of signature $\mathbf{S} = \langle S_1, \ldots, S_n, S\rangle$ and $\alpha_1, \ldots, \alpha_r$ be the free type variables of $\mathbf{S}$. We replace f with a function symbol $\hat{f}$ of arity $r+n$ with monomorphic type signature $\langle \mathsf{T}, \ldots, \mathsf{T}, [S_1], \ldots, [S_n], [S]\rangle$.
3. Let p be a predicate symbol of signature $\mathbf{S} = \langle S_1, \ldots, S_n\rangle$ and $\alpha_1, \ldots, \alpha_r$ be the free type variables of $\mathbf{S}$. We replace p with a predicate symbol $\hat{p}$ of arity $r+n$ with monomorphic type signature $\langle \mathsf{T}, \ldots, \mathsf{T}, [S_1], \ldots, [S_n]\rangle$.
4. For every variable symbol u and type T, we add a new variable symbol u_T.
5. For every type variable $\alpha \in \mathbb{V}_{\mathrm{T}}$, we add a new variable symbol v^{α}.
6. For every type constructor $\mathsf{F} \in \mathbb{F}_{\mathrm{T}}$, we add a new function symbol $\mathtt{F}$ of the same arity and with type signature $\langle \mathsf{T}, \ldots, \mathsf{T}, \mathsf{T}\rangle$.

The Exp transformation applies to non-protected types, translating them to terms of type T, exactly as Dec:

$$\textsc{Exp}(\alpha) \triangleq v^{\alpha} : \mathsf{T} \qquad \textsc{Exp}(\mathsf{F}(T_1, \ldots, T_n)) \triangleq \mathtt{F}(\textsc{Exp}(T_1), \ldots, \textsc{Exp}(T_n)) : \mathsf{T}$$

The Exp transformation applies to terms and formulas. In the definition below, $\mathbf{t}$ stands for a list of terms; $\alpha_1, \ldots, \alpha_r$ are the type variables of the type

signature of f and p; the type signature of f and p is instantiated with a type substitution τ; and $\beta_1, \ldots, \beta_m$ are the type variables of H:

$$\begin{aligned}
\textsc{Exp}(u : T) &\triangleq u_T : [T] \\
\textsc{Exp}(f(\mathbf{t}) : T) &\triangleq \hat{f}(\textsc{Exp}(\alpha_1\tau), \ldots, \textsc{Exp}(\alpha_r\tau), \textsc{Exp}(\mathbf{t})) : [T] \\
\textsc{Exp}(p(\mathbf{t})) &\triangleq \hat{p}(\textsc{Exp}(\alpha_1\tau), \ldots, \textsc{Exp}(\alpha_r\tau), \textsc{Exp}(\mathbf{t})) \\
\textsc{Exp}(t_1 \approx t_2) &\triangleq \textsc{Exp}(t_1) \approx \textsc{Exp}(t_2) \\
\textsc{Exp}(\neg F) &\triangleq \neg\, \textsc{Exp}(F) \\
\textsc{Exp}(F \wedge G) &\triangleq \textsc{Exp}(F) \wedge \textsc{Exp}(G) \\
\textsc{Exp}(\forall x\, F) &\triangleq \forall(\textsc{Exp}(x))\, \textsc{Exp}(F) \\
\textsc{Exp}^\circ(H) &\triangleq \forall(v^{\beta_1} : \mathsf{T}) \ldots \forall(v^{\beta_m} : \mathsf{T})\, \textsc{Exp}(H)
\end{aligned}$$

On the running example, assuming $U = \{\mathsf{I}\}$, the transformations Par and Exp$^\circ$ produce the following formulas (for the sake of clarity, we "forget" the variable subscripts added by Par):

$$\begin{aligned}
&\forall(v^\alpha : \mathsf{T})\, \forall(m_{\mathsf{M}(\alpha,\mathsf{I})} : \mathsf{U})\, \forall(c_\alpha : \mathsf{U})\, \mathtt{get}_{[\mathsf{I}/\beta]}(v^\alpha, \\
&\qquad\qquad \mathtt{set}_{[\mathsf{I}/\beta]}(v^\alpha, m_{\mathsf{M}(\alpha,\mathsf{I})}, c_\alpha, \mathsf{6}), c_\alpha) * \mathsf{7} \approx \mathsf{42} \\
&\forall(m_{\mathsf{M}(\mathsf{I},\mathsf{I})} : \mathsf{U})\, \forall(c_{\bar{\mathsf{I}}} : \bar{\mathsf{I}})\, \mathtt{get}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}(\mathtt{set}_{[\mathsf{I}/\alpha,\mathsf{I}/\beta]}(m_{\mathsf{M}(\mathsf{I},\mathsf{I})}, c_{\bar{\mathsf{I}}}, \mathsf{6}), c_{\bar{\mathsf{I}}}) * \mathsf{7} \approx \mathsf{42}
\end{aligned}$$

and Tw and Exp$^\circ$ give:

$$\begin{aligned}
&\forall(v^\alpha : \mathsf{T})\, \forall(m_{\mathsf{M}(\alpha,\mathsf{I})} : \mathsf{U})\, \forall(c_\alpha : \mathsf{U})\, \mathtt{from}_{\mathsf{I}}(\mathtt{get}(v^\alpha, \mathtt{I}, \\
&\qquad\qquad \mathtt{set}(v^\alpha, \mathtt{I}, m_{\mathsf{M}(\alpha,\mathsf{I})}, c_\alpha, \mathtt{to}_{\mathsf{I}}(\mathsf{6})), c_\alpha)) * \mathsf{7} \approx \mathsf{42}
\end{aligned}$$

Lemma 9. *For every term t of type T,* Exp(t) *is a well-formed monomorphic term of type $[T]$. For every formula F,* Exp(F) *is a well-formed monomorphic formula. Also,* $\mathrm{FV}(\textsc{Exp}(t)) = \{u_T : [T] \mid u : T \in \mathrm{FV}(t)\} \cup \{v^\alpha : \mathsf{T} \mid \alpha \in \mathrm{FV}_\mathrm{T}(t)\}$ *and* $\mathrm{FV}(\textsc{Exp}(F)) = \{u_T : [T] \mid u : T \in \mathrm{FV}(F)\} \cup \{v^\alpha : \mathsf{T} \mid \alpha \in \mathrm{FV}_\mathrm{T}(F)\}$. *Finally, for every closed formula F,* Exp$^\circ(F)$ *is a well-formed closed monomorphic formula.*

Proof. The last claim easily follows from the second one. The proof of the first two statements goes by induction on the structure of terms and formulas. The only interesting case is the application of a function symbol (predicate symbols are treated in the same way).

Consider a term $t = f(t_1 : T_1, \ldots, t_n : T_n) : T$. Let S_i denote the i-th sort in the type signature of f. We have two possible cases: either S_i and T_i are protected and $S_i = T_i$, or both S_i and T_i are non-protected, by the definition of matching with protected types. Therefore, $[T_i] = [S_i]$ for every $i \in [1, n]$.

Now, let S be the type of the value in the type signature of f. By the same argument, either S and T are protected and $S = T$, or both S and T are non-protected. Therefore, $[T] = [S]$ and the application is well-formed.

Finally, every type variable α occurring in $\langle T_1, \ldots, T_n, T\rangle$ appears as a free variable $v^\alpha : \mathsf{T}$ in the supplementary "type" arguments of $\hat{f}$ in Exp(t). □

Theorem 10 (Soundness of Exp). *If a set of closed formulas with protected sorts Γ is satisfiable so that every non-protected sort has an infinite domain in the model, then* $\mathrm{Exp}^\circ(\Gamma)$ *is satisfiable.*

Proof. Every model of Γ is a model of the set of monomorphic type instances $\mathrm{MI}(\Gamma)$ and vice versa. In this set, protected sorts require no special treatment and the whole problem is reduced to many-sorted logic. Let $\mathfrak{M}$ be a model of Γ such that, for every non-protected sort S, the domain $\mathcal{D}_S^{\mathfrak{M}}$ is infinite. By Löwenheim-Skolem theorem for many-sorted logic [11, Sect. 7.6], we can assume that the domain of every non-protected sort is simply $\mathbb{N}$.

We are going to construct an interpretation $\mathfrak{I}$ of $\mathrm{Exp}^\circ(\Gamma)$ such that, for every variable valuation φ in $\mathfrak{I}$, there exist a type valuation π' and a variable valuation φ' in the original model $\mathfrak{M}$, such that:

$$\mathfrak{I}_\varphi(\mathrm{Exp}(t)) = \mathfrak{M}_{\pi',\varphi'}(t) \qquad\qquad \mathfrak{I}_\varphi(\mathrm{Exp}(F)) = \mathfrak{M}_{\pi',\varphi'}(F)$$

for every term t and formula F occurring in Γ. We do not need to consider type valuations in $\mathfrak{I}$, since the result of Exp° is entirely monomorphic. As soon as these two equalities are proved, it is easy to show that for every closed formula $H \in \Gamma$, the interpretation $\mathfrak{I}$ satisfies $\mathrm{Exp}^\circ(H)$. Indeed,

$$\mathfrak{I}(\mathrm{Exp}^\circ(H)) = \bigwedge_\varphi \mathfrak{I}_\varphi(\mathrm{Exp}(H)) = \bigwedge_\varphi \mathfrak{M}_{\pi',\varphi'}(H) = \top$$

For every sort S in the initial signature, protected or not, we keep its domain: $\mathcal{D}_S^{\mathfrak{I}} \triangleq \mathcal{D}_S^{\mathfrak{M}}$. The domain of U is the set of natural numbers: $\mathcal{D}_{\mathsf{U}}^{\mathfrak{I}} \triangleq \mathbb{N}$. The domain of T is the set of non-protected sorts: $\mathcal{D}_{\mathsf{T}}^{\mathfrak{I}} \triangleq \mathcal{T}(\mathbb{F}_{\mathrm{T}})$.

Then we give an interpretation to the function and predicate symbols in $\mathrm{Exp}^\circ(\Gamma)$. Let f be a function symbol in Γ of type signature $\mathbf{S} = \langle S_1, \ldots, S_n, S\rangle$ and $\alpha_1, \ldots, \alpha_r$ be the type variables in $\mathbf{S}$. The symbol $\hat{f}$ has the arity $r+n$ and the type signature $\mathbf{S}' = \langle \mathsf{T}, \ldots, \mathsf{T}, [S_1], \ldots, [S_n], [S]\rangle$ and this is the only sort vector to interpret $\hat{f}$ on, since it is monomorphic. We define:

$$\hat{f}^{\mathfrak{I}}_{\mathbf{S}'}(T_1, \ldots, T_r, c_1, \ldots, c_n) \triangleq f^{\mathfrak{M}}_{\mathbf{S}\tau}(c_1, \ldots, c_n)$$

where τ is the type substitution $[T_1/\alpha_1, \ldots, T_r/\alpha_r]$. Notice that every type S in $\mathbf{S}$ verifies $\mathcal{D}^{\mathfrak{I}}_{[S]} = \mathcal{D}^{\mathfrak{M}}_{S\tau}$. Indeed, either S is protected and $[S] = S = S\tau$, or both S and $S\tau$ are non-protected and $\mathcal{D}^{\mathfrak{I}}_{[S]} = \mathcal{D}^{\mathfrak{I}}_{\mathsf{U}} = \mathbb{N} = \mathcal{D}^{\mathfrak{M}}_{S\tau}$.

Similarly, let p be a predicate symbol in Γ of type signature $\mathbf{S} = \langle S_1, \ldots, S_n\rangle$ and $\alpha_1, \ldots, \alpha_r$ be the type variables in $\mathbf{S}$. The symbol $\hat{p}$ has the arity $r+n$ and the type signature $\mathbf{S}' = \langle \mathsf{T}, \ldots, \mathsf{T}, [S_1], \ldots, [S_n]\rangle$. Then we set:

$$\hat{p}^{\mathfrak{I}}_{\mathbf{S}'}(T_1, \ldots, T_r, c_1, \ldots, c_n) \triangleq p^{\mathfrak{M}}_{\mathbf{S}[T_1/\alpha_1, \ldots, T_r/\alpha_r]}(c_1, \ldots, c_n)$$

Finally, the type-representing function symbols are interpreted exactly as the original type constructors:

$$\mathsf{F}^{\mathfrak{I}}_{\langle \mathsf{T}, \ldots, \mathsf{T}\rangle}(T_1, \ldots, T_n) \triangleq \mathsf{F}(T_1, \ldots, T_n)$$

Now, let us consider a variable valuation φ in $\mathfrak{I}$. For any type variable α, we set $\alpha\pi' \triangleq \varphi(v^\alpha : \mathsf{T})$. For any variable $u:T$ in Γ, we set $\varphi'(u:T) \triangleq \varphi(u_T:[T])$. As we have shown above, $\mathcal{D}^{\mathfrak{M}}_{T\pi'} = \mathcal{D}^{\mathfrak{I}}_{[T]}$. Also, for every non-protected type T, we have $\mathfrak{I}_\varphi(\textsc{Exp}(T)) = T\pi'$.

The proof of the two main equalities goes by induction on the structure of terms and formulas. We consider three cases: a variable, a function application, and a universally quantified formula.

1. Let $u:T$ be a variable in Γ. We obtain:

$$\mathfrak{I}_\varphi(\textsc{Exp}(u:T)) = \mathfrak{I}_\varphi(u_T:[T]) = \varphi(u_T:[T]) = \varphi'(u:T) = \mathfrak{M}_{\pi',\varphi'}(u:T)$$

2. Consider a term $t = f(t_1:T_1,\ldots,t_n:T_n):T$. Let $\mathbf{S} = \langle S_1,\ldots,S_n,S\rangle$ be the type signature of f and $\alpha_1,\ldots,\alpha_r$ be the type variables of $\mathbf{S}$. Let $\mathbf{S}'$ be the type signature of $\hat{f}$. Finally, let τ be the type substitution that instantiates $\mathbf{S}$ to $\langle T_1,\ldots,T_n,T\rangle$. We obtain:

$$\begin{aligned}\mathfrak{I}_\varphi(\textsc{Exp}(t)) &= \mathfrak{I}_\varphi(\hat{f}(\textsc{Exp}(\alpha_1\tau),\ldots,\textsc{Exp}(\alpha_r\tau),\textsc{Exp}(t_1),\ldots,\textsc{Exp}(t_n)):[S])\\ &= \hat{f}^{\mathfrak{I}}_{\mathbf{S}'}(\alpha_1\tau\pi',\ldots,\alpha_r\tau\pi',\mathfrak{M}_{\pi',\varphi'}(t_1),\ldots,\mathfrak{M}_{\pi',\varphi'}(t_n))\\ &= f^{\mathfrak{M}}_{\mathbf{S}\tau\pi'}(\mathfrak{M}_{\pi',\varphi'}(t_1),\ldots,\mathfrak{M}_{\pi',\varphi'}(t_n)) = \mathfrak{M}_{\pi',\varphi'}(t)\end{aligned}$$

3. Consider a universally quantified formula $\forall(u:T)\,F$. Let us take an arbitrary element $c \in \mathcal{D}^{\mathfrak{I}}_{[T]} = \mathcal{D}^{\mathfrak{M}}_{T\pi'}$. The variable valuation $\varphi[u_T:[T]\mapsto c]$ produces in $\mathfrak{M}$ the same type valuation π' and the variable valuation $\varphi'[u:T\mapsto c]$.

$$\begin{aligned}\mathfrak{I}_\varphi(\textsc{Exp}(\forall(u:T)\,F)) &= \bigwedge_{c\in\mathcal{D}^{\mathfrak{I}}_{[T]}} \mathfrak{I}_{\varphi[u_T:[T]\mapsto c]}(\textsc{Exp}(F))\\ &= \bigwedge_{c\in\mathcal{D}^{\mathfrak{M}}_{T\pi'}} \mathfrak{M}_{\pi',\varphi'[u:T\mapsto c]}(F) = \mathfrak{M}_{\pi',\varphi'}(\forall(u:T)\,F)\end{aligned}$$

Thus, for every formula F in Γ, $\textsc{Exp}^\circ(F)$ is satisfied under $\mathfrak{I}$. □

Theorem 11 (Completeness of Exp). *For every set of closed formulas with protected sorts Γ, if $\textsc{Exp}^\circ(\Gamma)$ is satisfiable then Γ is satisfiable.*

Proof. Let $\mathfrak{M}$ be a model of $\textsc{Exp}^\circ(\Gamma)$. We are going to construct an interpretation $\mathfrak{I}$ of Γ such that, for every type valuation π and every variable valuation φ in $\mathfrak{I}$, there exists a variable valuation φ' in the original model $\mathfrak{M}$, such that:

$$\mathfrak{I}_{\pi,\varphi}(t) = \mathfrak{M}_{\varphi'}(\textsc{Exp}(t)) \qquad\qquad \mathfrak{I}_{\pi,\varphi}(F) = \mathfrak{M}_{\varphi'}(\textsc{Exp}(F))$$

for every term t and formula F occurring in Γ. Since $\textsc{Exp}^\circ(F)$ is the universal closure of $\textsc{Exp}(F)$, the second equality suffices to show $\mathfrak{I}_\pi(F)$ for every $F \in \Gamma$.

For any sort S in the signature of Γ, we set $\mathcal{D}^{\mathfrak{I}}_S \triangleq \mathcal{D}^{\mathfrak{M}}_{[S]}$.

Consider a function symbol f in Γ of type signature $\mathbf{S} = \langle S_1,\ldots,S_n,S\rangle$ and let $\alpha_1,\ldots,\alpha_r$ be the type variables of $\mathbf{S}$. Let $\mathbf{S}'$ stand for the type signature of

$\hat{f}$, $\langle \mathsf{T}, \ldots, \mathsf{T}, [S_1], \ldots, [S_n], [S] \rangle$. The symbol f is then interpreted in $\mathfrak{I}$ on a sort vector $\mathbf{S}\tau$ as follows:

$$f^{\mathfrak{I}}_{\mathbf{S}\tau}(\mathbf{c}) \triangleq \hat{f}^{\mathfrak{M}}_{\mathbf{S}'}(\mathfrak{M}(\textsc{Exp}(\alpha_1\tau)), \ldots, \mathfrak{M}(\textsc{Exp}(\alpha_r\tau)), \mathbf{c}).$$

We omit the variable valuation when evaluating $\textsc{Exp}(\alpha_i\tau)$, as these terms do not contain variables. Notice that every argument c_i from $\mathbf{c}$ belongs either to $\mathcal{D}^{\mathfrak{M}}_{S_i}$ (if the sort S_i is protected) or to $\mathcal{D}^{\mathfrak{M}}_{\mathsf{U}}$ (if it is not), and thus the interpretation is well-defined.

Similarly, let p be a predicate symbol in Γ of type signature $\mathbf{S} = \langle S_1, \ldots, S_n \rangle$. Let $\alpha_1, \ldots, \alpha_r$ be the type variables of $\mathbf{S}$ and $\mathbf{S}'$ be the type signature of $\hat{p}$. The symbol p is then interpreted in $\mathfrak{I}$ on a sort vector $\mathbf{S}\tau$ as follows:

$$p^{\mathfrak{I}}_{\mathbf{S}\tau}(\mathbf{c}) \triangleq \hat{p}^{\mathfrak{M}}_{\mathbf{S}'}(\mathfrak{M}(\textsc{Exp}(\alpha_1\tau)), \ldots, \mathfrak{M}(\textsc{Exp}(\alpha_r\tau)), \mathbf{c}).$$

This concludes the definition of interpretation $\mathfrak{I}$.

Now, let us consider some type valuation π and a variable valuation φ in $\mathfrak{I}$. For every variable occurring in $\textsc{Exp}^{\circ}(\Gamma)$, we define:

$$\varphi'(v^{\alpha} : \mathsf{T}) \triangleq \mathfrak{M}(\textsc{Exp}(\alpha\pi)) \qquad \varphi'(u_T : [T]) \triangleq \varphi(u : T)$$

Notice that $\mathfrak{M}_{\varphi'}(\textsc{Exp}(T)) = \mathfrak{M}(\textsc{Exp}(T\pi))$ and $\mathcal{D}^{\mathfrak{I}}_{T\pi} = \mathcal{D}^{\mathfrak{M}}_{[T]}$.

The proof of the equalities above goes by induction on the structure of terms and formulas. We consider three cases: a variable, a function application, and a universally quantified formula.

1. Let $u : T$ be a variable occurring in Γ. We obtain:

$$\mathfrak{I}_{\pi,\varphi}(u : T) = \varphi(u : T) = \varphi'(u_T : [T]) = \mathfrak{M}_{\varphi'}(u_T : [T]) = \mathfrak{M}_{\varphi'}(\textsc{Exp}(u : T))$$

2. Consider a term $t = f(t_1 : T_1, \ldots, t_n : T_n) : T$ in Γ. Let $\mathbf{S} = \langle S_1, \ldots, S_n, S \rangle$ be the type signature of f, let $\alpha_1, \ldots, \alpha_r$ be the type variables of $\mathbf{S}$, let $\mathbf{S}'$ be the type signature of $\hat{f}$, and τ the type substitution that instantiates $\mathbf{S}$ to $\langle T_1, \ldots, T_n, T \rangle$. Denoting the list of arguments with $\mathbf{t}$, we get:

$$\begin{aligned}
\mathfrak{I}_{\pi,\varphi}(t) &= f^{\mathfrak{I}}_{\mathbf{S}\tau\pi}(\mathfrak{I}_{\pi,\varphi}(\mathbf{t})) \\
&= \hat{f}^{\mathfrak{M}}_{\mathbf{S}'}(\mathfrak{M}(\textsc{Exp}(\alpha_1\tau\pi)), \ldots, \mathfrak{M}(\textsc{Exp}(\alpha_r\tau\pi)), \mathfrak{I}_{\pi,\varphi}(\mathbf{t})) \\
&= \hat{f}^{\mathfrak{M}}_{\mathbf{S}'}(\mathfrak{M}_{\varphi'}(\textsc{Exp}(\alpha_1\tau)), \ldots, \mathfrak{M}_{\varphi'}(\textsc{Exp}(\alpha_r\tau)), \mathfrak{M}_{\varphi'}(\textsc{Exp}(\mathbf{t}))) \\
&= \mathfrak{M}_{\varphi'}(\hat{f}(\textsc{Exp}(\alpha_1\tau), \ldots, \textsc{Exp}(\alpha_r\tau), \textsc{Exp}(\mathbf{t})) : [S]) = \mathfrak{M}_{\varphi'}(\textsc{Exp}(t))
\end{aligned}$$

3. Consider a universal formula $\forall(u : T)\, F$. Let a be some element of $\mathcal{D}^{\mathfrak{I}}_{T\pi}$. The type valuation π and the variable valuation $\varphi[u : T \mapsto a]$ produces in $\mathfrak{M}$ the variable valuation $\varphi'[u_T : [T] \mapsto a]$. We obtain:

$$\begin{aligned}
\mathfrak{I}_{\pi,\varphi}(\forall(u : T)\, F) &= \bigwedge_{a \in \mathcal{D}^{\mathfrak{I}}_{T\pi}} \mathfrak{I}_{\pi,\varphi[u : T \mapsto a]}(F) \\
&= \bigwedge_{a \in \mathcal{D}^{\mathfrak{M}}_{[T]}} \mathfrak{M}_{\varphi'[u_T : [T] \mapsto a]}(\textsc{Exp}(F)) \\
&= \mathfrak{M}_{\varphi'}(\forall(u_T : [T])\, \textsc{Exp}(F)) = \mathfrak{M}_{\varphi'}(\textsc{Exp}(\forall(u : T)\, F))
\end{aligned}$$

Thus, for every formula F in Γ, F is satisfied under $\mathfrak{I}$. □

From a practical point of view, Theorem 10 is not comforting. Given a $\mathbf{FOL}_{\mathrm{T}}$-problem Γ, we cannot effectively decide which sorts admit infinite models and which do not (one can postulate a bijection between a given sort and the domain of a partial-recursive function). A practical way out could consist in a small language extension: for every type/sort, we specify explicitly whether it is finite or infinite. We proceed from the assumption that the author of any given problem knows the intended model of every type.

In Why3 [4], every type is declared either as abstract or algebraic (i.e., a sum of products). We postulate that abstract types are all infinite and we analyse the definitions of algebraic types to find out which of their monomorphic instances admit infinite models. For example, given the standard algebraic definitions of booleans (B), lists ($\mathsf{L}(\alpha)$), and pairs ($\mathsf{P}(\alpha, \beta)$), we can conclude that the sorts B and $\mathsf{P}(\mathsf{B}, \mathsf{B})$ are finite and I, $\mathsf{L}(\mathsf{B})$, and $\mathsf{P}(\mathsf{I}, \mathsf{B})$ are infinite.

Once we know the finite sorts, can we transform a problem to eliminate them, so that Exp (or a similar method) can be applied? Meng and Paulson propose to filter out the premises implying the finiteness of sorts [12, Sect. 2.8]; however, we need an infallible filter to ensure the soundness of type erasure. We have implemented an alternative solution which consists in putting a special "projection" function $\mathtt{proj}_T$ over every variable and function symbol of a finite type T. Thus, the premise $\forall(x : \mathsf{B})(x \approx \mathtt{True} \vee x \approx \mathtt{False})$ becomes $\forall(x : \mathsf{B})(\mathtt{proj}_{\mathsf{B}}(x) \approx \mathtt{proj}_{\mathsf{B}}(\mathtt{True}) \vee \mathtt{proj}_{\mathsf{B}}(x) \approx \mathtt{proj}_{\mathsf{B}}(\mathtt{False}))$ and the domain of B does not need to be finite anymore, as we confine ourselves to the range of $\mathtt{proj}_{\mathsf{B}}$. This method is still potentially unsound, as one can state the finiteness of a sort with a polymorphic axiom, where no projection would apply. Precisely, let $\mathtt{isUnit} : \langle\alpha\rangle$ be a unary predicate. Then the formulas $\forall(x : \alpha)(\mathtt{isUnit}(x) \supset \forall(y : \alpha)(y \approx x))$ and $\forall(x : \mathsf{A})\, \mathtt{isUnit}(x)$ imply that the sort A has a single inhabitant. Today, we know of no way to use Exp soundly on polymorphic problems with finite sorts.

The last remark to make is that Exp provides a path towards one-sorted languages. Indeed, in a monomorphic setting, Theorem **??** comes to: "if every sort admits an infinite domain, then we can safely erase the sort annotations". Thus, if we want to use a TPTP prover such as Vampire [16] or SPASS [18], we start by translating a proof task to the many-sorted language, using any of the methods described above. Then we eliminate the protected finite sorts (if any) using projections; in absence of polymorphism, this is a sound and complete transformation. And finally, we apply Exp assuming that all types are non-protected, which amounts to simply erasing all sorts.

4 Experiments and Conclusion

In our experiments, we wanted to compare the impact of different "paths" of polymorphism encoding on the performance of three well-established SMT solvers. We omit the Par transformation (see the remark in the end of Section 3.2) and we add the classical type encoding technique with per-variable

"type guards" [11]. Our implementation of this method (denoted Grd below) closely follows the description given in [10, Sect. 3.0].

We run our tests on 4123 verification conditions generated by the Why platform from 166 programs, which originate from Caduceus [7], Jessie [14], or directly from Why. Translated tasks were sent to Z3, CVC3, and Yices with a time limit of 60 seconds. On the whole, 3993 proof obligations were proved by at least one prover. The initial Why3 files and our results are available at `http://why3.lri.fr/download/polyfol_encoding.tar.gz`.

We have tested the encodings Tw+Dec, Tw+Exp, and Tw+Grd, both with and without Dis. In the latter case, these methods correspond to what is described in [5] and [10]; the set U of sorts to protect in Tw is set to contain only integers and reals, which are natively supported by the three provers. In presence of Dis, we put in the set W every monomorphic specialization that occurs in the goal formula along with the specializations of access and update operations on every monomorphic array type in the goal; we also protect every sort in the goal (as well as integers and reals). This configuration of Dis and Tw gives better results comparing to other configurations that we tried, e.g., collect the specializations and the sorts to protect from the whole proof task.

Z3 (3809)	Tw+Grd	Tw+Exp	Tw+Dec	Dis+Tw+Grd	Dis+Tw+Exp
Dis+Tw+Dec	+203 −36	+20 −49	+66 −37	+18 −5	+26 −30
Dis+Tw+Exp	+191 −20	+13 −38	+63 −30	+35 −18	
Dis+Tw+Grd	+195 −41	+11 −53	+59 −43		
Tw+Dec	+157 −19	+15 −73			
Tw+Exp	+211 −15				

CVC3 (3756)	Tw+Grd	Tw+Exp	Tw+Dec	Dis+Tw+Grd	Dis+Tw+Exp
Dis+Tw+Dec	+269 −20	+0 −26	+84 −19	+66 −4	+0 −6
Dis+Tw+Exp	+272 −17	+0 −20	+88 −17	+69 −1	
Dis+Tw+Grd	+204 −17	+1 −89	+46 −43		
Tw+Dec	+188 −4	+0 −91			
Tw+Exp	+275 −0				

Yices (3717)	Tw+Grd	Tw+Exp	Tw+Dec	Dis+Tw+Grd	Dis+Tw+Exp
Dis+Tw+Dec	+882 −6	+13 −276	+379 −79	+204 −2	+3 −272
Dis+Tw+Exp	+1149 −4	+39 −33	+574 −5	+472 −1	
Dis+Tw+Grd	+684 −10	+6 −471	+241 −143		
Tw+Dec	+577 −1	+5 −568			
Tw+Exp	+1140 −1				

Table 1. Experimental comparison of polymorphism encoding methods

Our results are given in Table 1. To the right of the prover's name, we put the number of goals proved by at least one encoding method. In every cell we specify the number of goals proved by one encoding but not by the other one.

For example, with CVC3, the encoding by Dis+Tw+Dec allows us to prove 84 goals that were not proved by Tw+Dec. On the other hand, with Tw+Dec, CVC3 proves 19 goals that were not proved with Dis+Tw+Dec.

On the average, symbol discrimination increases the number of premises by a factor of 1.8 (ranging from 1 to 10 on some examples). Nevertheless, adding the Dis phase allows us to prove more goals in every case except for Z3 and CVC3 with Exp. In particular, the Grd transformation is remarkably helped by Dis. Apart from the possibility to use the built-in support for arrays, the effectiveness of Dis is also explained by the fact that we protect the sorts that occur in the selected monomorphic specializations. Thus, the new premises generated by Dis are not only instantiated to the relevant sorts, they are also liberated from decorations imposed by the third, type-fusing, stage. This effect is less important in the case of Exp, because this transformation, unlike Dec and Grd, adds very little clutter to the encoded formulas in the first place.

The comparison between Exp, Dec, and Grd shows that Exp is generally more efficient than Dec which in its turn is more efficient than Grd. This is quite different from the results given in [10], where Exp and Grd have roughly the same performance. We have not yet identified whether this discrepancy comes from the difference in our test cases or in our implementations.

Conclusion. In the present paper, we described first-order logic with polymorphic types and introduced generic notions to define and reason about practical methods of polymorphism elimination. Using these notions, we generalized and proved two translation techniques known from literature. We also proposed to combine type protection with symbol discrimination. As our experiments show, this improves the performance of automated proof search and allows us to use built-in theories of complex types, such as arrays, in SMT solvers. One interesting problem we would like to resolve in the future is protection of polymorphic types, allowing to merge all monomorphic instances of a given complex type in a single protected sort. We also believe that better heuristics to choose the sets W and U can be devised, and further experiments are in order.

References

1. Barrett, C., Stump, A., Tinelli, C.: The SMT-LIB Standard: Version 2.0. Tech. rep., Department of Computer Science, The University of Iowa (2010)
2. Barrett, C., Tinelli, C.: CVC3. In: CAV'07. LNCS, vol. 4590, pp. 298–302. Springer (2007)
3. Bobot, F., Conchon, S., Contejean, E., Lescuyer, S.: Implementing polymorphism in SMT solvers. In: SMT'08. ACM ICPS, vol. 367, pp. 1–5 (2008)
4. Bobot, F., Filliâtre, J.C., Marché, C., Paskevich, A.: Why3, a theorem proving framework (2010), `http://why3.gforge.inria.fr/`
5. Couchot, J.F., Lescuyer, S.: Handling polymorphism in automated deduction. In: CADE-21. LNAI, vol. 4603, pp. 263–278. Springer (2007)
6. Dutertre, B., de Moura, L.: The YICES SMT solver. Tech. rep., SRI International (2006)

7. Filliâtre, J.C., Marché, C.: Multi-prover verification of C programs. In: ICFEM'04. LNCS, vol. 3308, pp. 15–29. Springer (2004)
8. Hurd, J.: An LCF-style interface between HOL and first-order logic. In: CADE-18. LNAI, vol. 2392, pp. 134–138. Springer (2002)
9. Hurd, J.: First-order proof tactics in higher-order logic theorem provers. In: Design and Application of Strategies/Tactics in Higher Order Logics. pp. 56–68. NASA Technical Report NASA/CP-2003-212448 (2003)
10. Leino, K.R.M., Rümmer, P.: A polymorphic intermediate verification language: Design and logical encoding. In: TACAS'10. LNCS, vol. 6015, pp. 312–327. Springer (2010)
11. Manzano, M.: Extensions of First-Order Logic, Cambridge Tracts in Theoretical Computer Science, vol. 19. Cambridge University Press (1996)
12. Meng, J., Paulson, L.C.: Translating higher-order clauses to first-order clauses. Journal of Automated Reasoning 40(1), 35–60 (2008)
13. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: TACAS'08. LNCS, vol. 4963, pp. 337–340. Springer (2008)
14. Moy, Y., Marché, C.: Jessie Plugin Tutorial, Lithium version. INRIA (2008), `http://www.frama-c.cea.fr/jessie.html`
15. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL — A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
16. Riazanov, A., Voronkov, A.: Vampire 1.1. In: IJCAR'01. LNCS, vol. 2083, pp. 376–380. Springer (2001)
17. The Coq Development Team: The Coq Proof Assistant Reference Manual – Version V8.0 (2004), `http://coq.inria.fr`
18. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: CADE-22. LNCS, vol. 5663, pp. 140–145. Springer (2009)