Axiomatization of Typed First-Order Logic

Peter H. Schmitt and Mattias Ulbrich

Karlsruhe Institute of Technology (KIT), Dept. of Informatics
Am Fasanengarten 5, 76131 Karlsruhe, Germany

Abstract. This paper contributes to the theory of typed first-order logic. We present a sound and complete axiomatization for a basic typed logic lifting restrictions imposed by previous results. As a second contribution, this paper provides complete axiomatizations for the type predicates $instance_T$, $exactInstance_T$, and functions $cast_T$ indispensable for reasoning about object-oriented programming languages.

1 Introduction

Typed first-order logics with sophisticated type systems are by now a tried-and-tested basis for program verification systems. The most common route to proof support for these logics is by translation to simply sorted or unsorted logics for which SMT solvers are available. Typical representatives of this approach are the translation of the Boogie type system explained in [7] and the translation of the type system used in the Why verification tool described in [3]. The alternative approach to implement provers for such typed first-order logics directly is less common, e.g., the prover integrated in the KeY system [1] and the Alt-Ergo SMT solver [2].

In this paper we present a logical framework for a hierarchically typed first order logic and a sound and complete calculus. This paper furthermore addresses an issue with the coincidence lemma that is folklore knowledge in the community though we could not find a published reference. Let $\mathcal{T}_1 = \{A, B\}$ be an example type hierarchy containing the two types A, B with $A \not\sqsubseteq B$ and $B \not\sqsubseteq A$. Let x be a variable of type A and y be a variable of type B. Then the formula $\neg\exists x.\exists y.(x \doteq y)$ is a tautology. If we extend $\mathcal{T}_1$ to the type hierarchy $\mathcal{T}_2 = (\{A, B, C\}, \sqsubseteq)$ with $C \sqsubseteq A$ and $C \sqsubseteq B$, the very same formula is no longer a tautology in the logic using $\mathcal{T}_2$. This phenomenon that universal validity of a formula depends on symbols not occurring in it, is highly undesirable. We adapt the notion of universal validity to exclude this deficiency in Definition 8 below.

There is a rich body of recent literature on the translation approach investigating various variations, optimizations, and tunings directly geared towards program verification, see again [7,3] for references. Typed, or many-sorted, calculi have a long tradition in mathematical logic, [8] may be counted among the earliest contributions in this line. More recently there was a short-lived flurry on order-sorted logic programming and resolution calculi, as witnessed e.g., by [10]. Despite this history there are not many recent papers on implemented calculi of typed first-order logic, let alone contributions aimed at program verification.

This paper falls into this category. The contribution closest to ours is [6] that presents a sound and complete sequent calculus under the restriction that the type hierarchy is closed under greatest lower bounds. An extended version has been published as [1, Chapter 2]. We improve on this by 1) lifting the restriction on the type hierarchy and 2) reducing the logic to a minimal predefined vocabulary, where equality $\doteq$ is the only built-in predicate, and defining the desired vocabulary axiomatically.

Plan of the paper: The main part of Section 2, after introducing the basic typed logic, is taken up by the proof of the soundness and completeness theorem, Theorem 1. In Section 3, an example of a theory in the basic typed logic is presented axiomatizing $instance_T$, $exactInstance_T$, and $cast_T$ culminating again in a soundness and completeness proof, Theorem 3. Section 4 contains a few hints how some Java-specific notions could be axiomatized. We close with concluding remarks in Section 5.

We thank an anonymous reviewer for the thorough reading of the first version of this paper and his or her expert and useful comments.

2 The Basic Typed Logic

2.1 Syntax

Definition 1. *a* type hierarchy $\mathcal{T} = (\mathrm{TSym}, \sqsubseteq)$ *consists of*

1. *a non-empty set* TSym *of type symbols,*
2. *a partial order relation* $\sqsubseteq$ *on* TSym *called the subtype relation,*
3. *the designated symbols* $\bot \in \mathrm{TSym}$ *for the* empty *type and* $\top \in \mathrm{TSym}$ *for the* universal *type,*
4. $\bot \sqsubseteq A \sqsubseteq \top$ *for all* $A \in \mathrm{TSym}$.

We point out that no further restrictions are placed on type hierarchies in contrast to other approaches requiring the existence of greatest lower bounds. The empty type $\bot$ only plays an *ornamental* role in this paper. We nevertheless kept it in the hope that it may find its uses in future developments.

Definition 2. *A* signature $\Sigma = (\mathrm{FSym}, \mathrm{PSym}, \mathrm{VSym})$ *for a given type hierarchy* $\mathcal{T}$ *is made up of*

1. *a set* FSym *of typed function symbols,*
   by $f : A_1, \ldots, A_n \to A$ *we declare the argument types of* $f \in \mathrm{FSym}$ *to be* $A_1, \ldots, A_n$ *in the given order and its result type to be* A*,*
2. *a set* PSym *of typed predicate symbols,*
   by $p : A_1, \ldots, A_n$ *we declare the argument types of* $p \in \mathrm{PSym}$ *to be* $A_1, \ldots, A_n$ *in the given order,*
3. *a set* VSym *of typed variable symbols,*
   by $v : A$ *for* $v \in \mathrm{VSym}$ *we declare* v *to be a variable of type* A.
4. PSym *contains the dedicated symbol* $\doteq\, : \top, \top$ *for equality.*

In the above all A, $A_1, \dots, A_n$ in TSym *are required to be different from $\bot$. We do not allow overloading: The same symbol may not occur in* FSym $\cup$ PSym $\cup$ VSym *with different typing.*

The next two definitions define the syntactic categories of terms and formulas of typed first-order logic, as usual.

Definition 3. *Let $\mathcal{T}$ be a type hierarchy, and Σ a signature for $\mathcal{T}$. The set* Trm_A *of terms of type $A \neq \bot$ is inductively defined such that*

1. *$v \in \mathrm{Trm}_A$ for each variable symbol $v : A \in \mathrm{VSym}$ of type A.*
2. *$f(t_1, \dots, t_n) \in \mathrm{Trm}_A$ for each $f : A_1, \dots, A_n \to A \in \mathrm{FSym}$, and terms $t_i \in \mathrm{Trm}_{B_i}$ with $B_i \sqsubseteq A_i$ for all $1 \leq i \leq n$.*

If $t \in \mathrm{Trm}_A$, we say that t is of (static) type A and write $\sigma(t) = A$.

Definition 4. *The set* Fml *is inductively defined as:*

1. *$p(t_1, \dots, t_n) \in \mathrm{Fml}$*
   for $p : A_1, \dots, A_n \in \mathrm{PSym}$, and $t_i \in \mathrm{Trm}_{B_i}$ with $B_i \sqsubseteq A_i$ for all $1 \leq i \leq n$.
   In particular $t_1 \doteq t_2 \in \mathrm{Fml}$ for arbitrary terms t_i.
2. *true, false $\in \mathrm{Fml}$*
3. *$\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \to \psi$ are in* Fml *for arbitrary $\phi, \psi \in \mathrm{Fml}$.*
4. *$\forall v.\phi$, $\exists v.\phi$ are in* Fml *for $\phi \in \mathrm{Fml}$ and $v : A \in \mathrm{VSym}$.*

If need arises, we will make dependence of these definitions on Σ and $\mathcal{T}$ explicit by writing $\mathrm{Trm}_{A,\Sigma}$, Fml_Σ or $\mathrm{Trm}_{A,\mathcal{T},\Sigma}$, $\mathrm{Fml}_{\mathcal{T},\Sigma}$. When convenient, we will also use the redundant notation $\forall\, v{:}A.\phi$, $\exists\, v{:}A.\phi$ for a variable $v : A \in \mathrm{VSym}$.

Free and bound variables are defined as usual as well as typed substitutions that allow one to replace a variable of type A by a term of type B if $B \sqsubseteq A$.

2.2 Semantics

Definition 5. *A* universe *or* domain *for a given type hierarchy $\mathcal{T}$ and signature Σ consists of*

1. *a non-empty set D,*
2. *a typing function $\delta : D \to \mathrm{TSym} \setminus \{\bot\}$.*

The sets $D^A = \{d \in D \mid \delta(d) \sqsubseteq A\}$ for $A \in \mathrm{TSym}$ are called type universe or type domain for A. We require that $D^A \neq \emptyset$ for each $A \in \mathrm{TSym}$ with $A \neq \bot$.

The typing function δ assigns to every element $o \in D$ of the universe its *dynamic type* $\delta(o)$ and the type domain D^A of a type A contains all elements of dynamic type A or of a dynamic type which is a subtype of A. Definition 5 implies that for different types A, $B \in \mathrm{TSym} \setminus \{\bot\}$, there is $o \in D^A \cap D^B$ only if there exists $C \in \mathrm{TSym}$, $C \neq \bot$ with $C \sqsubseteq A$ and $C \sqsubseteq B$.

Lemma 1. *The type domains for a universe (D, δ) share the following properties:*

1. $D^{\perp} = \emptyset$, $D^{\top} = D$,
2. $D^A \subseteq D^B$ *if* $A \sqsubseteq B$,
3. $D^C = D^A \cap D^B$ *in case the greatest lower bound* C *of* A *and* B *exists.*

Definition 6. *A first-order* structure $\mathcal{M}$ *for a given type hierarchy* $\mathcal{T}$ *and signature* Σ *consists of a domain* (D, δ) *and an interpretation* I *such that*

1. $I(f)$ *is a function from* $D^{A_1} \times \ldots \times D^{A_n}$ *into* D^A *for* $f : A_1, \ldots, A_n \to A$ *in* FSym,
2. $I(p)$ *is a subset of* $D^{A_1} \times \ldots \times D^{A_n}$ *for* $p : A_1, \ldots, A_n$ *in* PSym,
3. $I(\doteq) = \{(d, d) \mid d \in D\}$.

For a first-order structure $\mathcal{M}$ and variable assignment β the evaluation $\mathrm{val}_{\mathcal{M},\beta}(t)$ of a term t and the semantic truth relation $(\mathcal{M}, \beta) \models \phi$ for formulas ϕ are defined as usual.

2.3 Calculus

The following definition formalizes our concept of enlarging a type hierarchy: new types and subtype relations between old and new types may be added without adding new relations among the old types. This can be guaranteed by the restriction that new types can only be declared to be subtypes of old types, never supertypes.

Definition 7. *A type hierarchy* $\mathcal{T}_2 = (\mathrm{TSym}_2, \sqsubseteq_2)$ *is an* extension *of a type hierarchy* $\mathcal{T}_1 = (\mathrm{TSym}_1, \sqsubseteq_1)$, *in symbols* $\mathcal{T}_1 \sqsubseteq \mathcal{T}_2$, *if*

1. $\mathrm{TSym}_1 \subseteq \mathrm{TSym}_2$
2. $\sqsubseteq_2$ *is the smallest subtype relation containing* $\sqsubseteq_1 \cup \Delta$ *where* Δ *is a set of relations* (S, T) *with* $T \in \mathrm{TSym}_1$ *and* $S \in \mathrm{TSym}_2 \setminus \mathrm{TSym}_1$.

Note, that this definitions entails , $\perp \sqsubseteq_2 A \sqsubseteq_2 \top$ for all new types A. For later reference, we note the following lemma.

Lemma 2. *Let* $\mathcal{T}_2 = (\mathrm{TSym}_2, \sqsubseteq_2)$ *be an extension of* $\mathcal{T}_1 = (\mathrm{TSym}_1, \sqsubseteq_1)$ *with* $\sqsubseteq_2$ *being the smallest subtype relation containing* $\sqsubseteq_1 \cup \Delta$, *for some* $\Delta \subseteq (\mathrm{TSym}_2 \setminus \mathrm{TSym}_1) \times \mathrm{TSym}_1$. *Then for* $A, B \in \mathrm{TSym}_1$, $C \in \mathrm{TSym}_2 \setminus \mathrm{TSym}_1$, $D \in \mathrm{TSym}_2$:

1. $A \sqsubseteq_2 B \Leftrightarrow A \sqsubseteq_1 B$
2. $C \sqsubseteq_2 A \Leftrightarrow T \sqsubseteq_1 A$ *for some* $(C, T) \in \Delta$
3. $D \sqsubseteq_2 C \Leftrightarrow D = C$ *or* $D = \perp$.

Proof. This follows easily from the fact that no supertype relations of the form $A \sqsubseteq_2 C$ for new type symbols C are stipulated. □

The following adapted definition of universal validity resolves the undesirable phenomenon, already referred to in the introduction, that validity of a formula depends on symbols not occurring in it.

$$\textsc{andLeft}\ \frac{\Gamma, \phi, \psi \Longrightarrow \Delta}{\Gamma, \phi \wedge \psi \Longrightarrow \Delta} \qquad \textsc{andRight}\ \frac{\Gamma \Longrightarrow \phi, \Delta \qquad \Gamma \Longrightarrow \psi, \Delta}{\Gamma \Longrightarrow \phi \wedge \psi, \Delta}$$

$$\textsc{orRight}\ \frac{\Gamma \Longrightarrow \phi, \psi, \Delta}{\Gamma \Longrightarrow \phi \vee \psi, \Delta} \qquad \textsc{orLeft}\ \frac{\Gamma, \phi \Longrightarrow \Delta \qquad \Gamma, \psi \Longrightarrow \Delta}{\Gamma, \phi \vee \psi \Longrightarrow \Delta}$$

$$\textsc{impRight}\ \frac{\Gamma, \phi \Longrightarrow \psi, \Delta}{\Gamma \Longrightarrow \phi \to \psi, \Delta} \qquad \textsc{impLeft}\ \frac{\Gamma \Longrightarrow \phi, \Delta \qquad \Gamma, \psi \Longrightarrow \Delta}{\Gamma, \phi \to \psi \Longrightarrow \Delta}$$

$$\textsc{notLeft}\ \frac{\Gamma \Longrightarrow \phi, \Delta}{\Gamma, \neg\phi \Longrightarrow \Delta} \qquad \textsc{notRight}\ \frac{\Gamma, \phi \Longrightarrow \Delta}{\Gamma \Longrightarrow \neg\phi, \Delta}$$

$$\textsc{allRight}\ \frac{\Gamma \Longrightarrow [x/c](\phi), \Delta}{\Gamma \Longrightarrow \forall\, x{:}A.\phi, \Delta} \qquad \textsc{allLeft}\ \frac{\Gamma, \forall\, x{:}A.\phi, [x/t](\phi) \Longrightarrow \Delta}{\Gamma, \forall\, x{:}A.\phi \Longrightarrow \Delta}$$

$c : \to A$ a new constant $\qquad t \in \mathrm{Trm}_{A'}$ ground, $A' \sqsubseteq A$

$$\textsc{exLeft}\ \frac{\Gamma, [x/c](\phi) \Longrightarrow \Delta}{\Gamma, \exists\, x{:}A.\phi \Longrightarrow \Delta} \qquad \textsc{exRight}\ \frac{\Gamma \Longrightarrow \exists\, x{:}A.\phi, [x/t](\phi), \Delta}{\Gamma \Longrightarrow \exists\, x{:}A.\phi, \Delta}$$

$c : \to A$ a new constant $\qquad t \in \mathrm{Trm}_{A'}$ ground, $A' \sqsubseteq A$

$$\textsc{close}\ \frac{}{\Gamma, \phi \Longrightarrow \phi, \Delta}$$

$$\textsc{closeFalse}\ \frac{}{\Gamma, \mathrm{false} \Longrightarrow \Delta} \qquad \textsc{closeTrue}\ \frac{}{\Gamma \Longrightarrow \mathrm{true}, \Delta}$$

Fig. 1. First-order rules

Definition 8. *Let $\mathcal{T}$ be a type hierarchy and Σ a signature, $\phi \in \mathrm{Fml}_{\mathcal{T},\Sigma}$ a formula without free variables, and $\Phi \subseteq \mathrm{Fml}_{\mathcal{T},\Sigma}$ a set of formulas without free variables.*

1. *ϕ is a* logical consequence *of Φ, in symbols $\Phi \vdash \phi$, if for all type hierarchies $\mathcal{T}'$ with $\mathcal{T} \sqsubseteq \mathcal{T}'$, and all $\mathcal{T}'$-Σ-structures $\mathcal{M}$ such that $\mathcal{M} \models \Phi$ also $\mathcal{M} \models \phi$ holds.*
2. *ϕ is* universally valid *if it is a logical consequence of the empty set, i.e. $\emptyset \vdash \phi$.*
3. *ϕ is satisfiable if there is a type hierarchy $\mathcal{T}'$, with $\mathcal{T} \sqsubseteq \mathcal{T}'$ and a $\mathcal{T}'$-Σ-structure $\mathcal{M}$ with $\mathcal{M} \models \phi$.*

The notion of *logical consequence* from Definition 8 may be called *super logical consequence* to distinguish it from the concept $\Phi \vdash_{\mathcal{T},\Sigma} \phi$ that is true when for any $\mathcal{T}$-Σ-structure $\mathcal{M}$ with $\mathcal{M} \models \Phi$ also $\mathcal{M} \models \phi$ is true. For the above example type hierarchy $\mathcal{T}_1 = \{A, B\}$ which had unrelated types A and B ($A \not\sqsubseteq B, B \not\sqsubseteq A$), we obtain $\not\vdash \neg\exists\, x{:}A.\exists\, y{:}B.(x \doteq y)$, but $\vdash_{\mathcal{T}_1,\emptyset} \neg\exists\, x{:}A.\exists\, y{:}B.(x \doteq y)$.

Definition 7 forbids the introduction of subtype chains like $A \sqsubseteq B \sqsubseteq T$ into the type hierarchy. However, it can be shown that relaxing the definition in that respect results in an equivalent notion of logical consequence. We keep the restriction here since it simplifies reasoning about type hierarchy extensions.

The calculus of our choice is the *sequent calculus*. The basic data that is manipulated by the rules of the sequent calculus are *sequents*. These are of the form $\phi_1, \ldots, \phi_n \Longrightarrow \psi_1, \ldots, \psi_m$ The formulas $\phi_1, \ldots, \phi_n$ at the left-hand side of the sequent separator, $\Longrightarrow$, are the premises or the antecedent of the sequent, the formulas $\psi_1, \ldots, \psi_m$ on the right are the conclusions or the succedent. The intended meaning of a sequent is that the premises together imply at least one conclusion. In other words, a sequent $\phi_1, \ldots, \phi_n \Longrightarrow \psi_1, \ldots, \psi_m$ is valid iff the formula $\bigwedge_{1 \leq i \leq n} \phi_i \to \bigvee_{1 \leq j \leq m} \psi_j$ is valid.

Figures 1 and 2 show the usual set of rules of the sequent calculus with equality as it can be found in many text books, e.g. [5, Section 5.4]. The only exception is rule eqDynamicSort, which is the main innovation of our approach. Here, $[z/t_2](\phi)$ is used to denote the application of the substitution t_2 for free occurrences of variable z in ϕ. Note that the rules contain the schematic variables Γ, Δ for set of formulas, ψ, ϕ for formulas and t, c for terms and constants.

The rule eqDynamicSort is different than the other rules in that it introduces a new type. In that sense it is similar to the rules forallRight and exLeft that introduce new constant symbols. These constants are used to denote an unknown value whose existence is guaranteed. The rationale behind the new type C in eqDynamicSort is the same: Since the equality $t_1 \doteq t_2$ in the antecedent requires that the types $\sigma(t_1)$ and $\sigma(t_2)$ have a common element, there must also be a type to accommodate that value. In the rule that type is made explicit and named C.

Theorem 1 (Soundness and Completeness Theorem).
Let $\mathcal{T} = (\mathrm{TSym}, \sqsubseteq)$ be a type hierarchy and Σ a signature, $\Gamma, \Delta \subset \mathrm{Fml}_{\mathcal{T},\Sigma}$ without free variables. Assume that for every $A \in \mathrm{TSym} \setminus \{\bot\}$ there is a constant symbol of type A. Then:

$\Gamma \Longrightarrow \Delta$ is universally valid iff there is a closed proof tree for the $\Gamma \Longrightarrow \Delta$.

Proof. Soundness. We only present the proof for the new rule eqDynamicSort, the remaining cases follow the usual pattern.

To prove soundness of eqDynamicSort we assume that $\bigwedge \Gamma \wedge t_1 \doteq t_2 \wedge \exists x.(x \doteq t_1 \wedge x \doteq t_2) \to \bigvee \Delta$ is universally valid for type hierarchy $\mathcal{T}_C$ and need to show that $\bigwedge \Gamma \wedge t_1 \doteq t_2 \to \bigvee \Delta$ is universally valid for the hierarchy $\mathcal{T}$.

The type hierarchy $\mathcal{T}_C = (\mathrm{TSym} \cup \{C\}, \sqsubseteq_C)$ is an extension of the hierarchy $\mathcal{T} = (\mathrm{TSym}, \sqsubseteq)$ in the sense of Definition 7 , with $\sqsubseteq_C$ the least subtype relation containing $\sqsubseteq \cup \{(C, \sigma(t_1)), (C, \sigma(t_2))\}$.

Assume that $\bigwedge \Gamma \wedge t_1 \doteq t_2 \wedge \exists x.(x \doteq t_1 \wedge x \doteq t_2) \to \bigvee \Delta$ is universally valid for type hierarchy $\mathcal{T}_C$. To prove universal validity of $\bigwedge \Gamma \wedge t_1 \doteq t_2 \to \bigvee \Delta$ we need to consider a type hierarchy $\mathcal{T}^1 = (\mathrm{TSym}^1, \sqsubseteq^1)$ that is an arbitrary extension of $\mathcal{T} = (\mathrm{TSym}, \sqsubseteq)$ and an arbitrary $(\mathcal{T}^1, \Sigma)$-structure $\mathcal{M} = (M, \delta, I)$ satisfying $\mathcal{M} \models \Gamma \wedge t_1 \doteq t_2$ with the aim of showing $\mathcal{M} \models \bigvee \Delta$. Let T be the dynamic type of $t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$, i.e., $\delta(t_1^{\mathcal{M}}) = \delta(t_2^{\mathcal{M}}) = T$, which obviously satisfies $T \sqsubseteq \sigma(t_1)$ and $T \sqsubseteq \sigma(t_2)$. Set $\mathcal{T}_C^1 = (\mathrm{TSym}^1 \cup \{C\}, \sqsubseteq_C^1)$ with $\sqsubseteq_C^1$ being the smallest subtype relation containing $\sqsubseteq^1 \cup \{(C, \sigma(t_1)), (C, \sigma(t_2))\}$. We need a

$$\textsc{eqLeft}\ \frac{\Gamma, t_1 \doteq t_2, [z/t_1](\phi), [z/t_2](\phi) \Longrightarrow \Delta}{\Gamma, t_1 \doteq t_2, [z/t_1](\phi) \Longrightarrow \Delta}$$
if $\sigma(t_2) \sqsubseteq \sigma(t_1)$

$$\textsc{eqRight}\ \frac{\Gamma, t_1 \doteq t_2 \Longrightarrow [z/t_2](\phi), [z/t_1](\phi), \Delta}{\Gamma, t_1 \doteq t_2 \Longrightarrow [z/t_1](\phi), \Delta}$$
if $\sigma(t_2) \sqsubseteq \sigma(t_1)$

$$\textsc{eqSymmLeft}\ \frac{\Gamma, t_2 \doteq t_1 \Longrightarrow \Delta}{\Gamma, t_1 \doteq t_2 \Longrightarrow \Delta} \quad \textsc{eqReflLeft}\ \frac{\Gamma, t \doteq t \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta}$$

$$\textsc{eqDynamicSort}\ \frac{\Gamma, t_1 \doteq t_2, \exists x.(x \doteq t_1 \land x \doteq t_2) \Longrightarrow \Delta}{\Gamma, t_1 \doteq t_2 \Longrightarrow \Delta}$$
if $\sigma(t_1)$ and $\sigma(t_2)$ are incomparable,
the sort C of x is new and satisfies $C \sqsubset \sigma(t_1)$ and $C \sqsubset \sigma(t_2)$

Fig. 2. Equality rules

further extension $\mathcal{T}^2 = (\mathrm{TSym}^1 \cup \{C, C^2\}, \sqsubseteq^2)$ of $\mathcal{T}^1_C$, where $\sqsubseteq^2$ is the smallest subtype relation containing $\sqsubseteq^1_C \cup \Delta$ with $\Delta = \{(C^2, C), (C^2, T)\}$.

We proceed in the main proof by constructing a $(\mathcal{T}^2, \Sigma)$-structure $\mathcal{M}^2 = (M, \delta^2, I)$ that differs from $\mathcal{M}$ only in δ^2 which is given by

$$\delta^2(o) = \begin{cases} C^2 & \text{if } o = t_1^{\mathcal{M}} = t_2^{\mathcal{M}} \\ \delta(o) & \text{otherwise .} \end{cases}$$

This leads to $\mathcal{M}^2 \models \exists x.(x \doteq t_1 \land x \doteq t_2)$ – if we remember that x is a variable of the type C and $C^2 \sqsubseteq C$.

The crucial property of the type hierarchy $\mathcal{T}^2$ is

$$\text{for any } o \in M \text{ and any } A \in \mathrm{TSym}^1 \quad \delta(o) \sqsubseteq^1 A \Leftrightarrow \delta^2(o) \sqsubseteq^2 A\ . \tag{1}$$

Here are the arguments why (1) is true: In case $o \neq t_1^{\mathcal{M}}$ we have $\delta^2(o) = \delta(o) \in \mathrm{TSym}^1$ and $\delta(o) \sqsubseteq^1 A \Leftrightarrow \delta(o) \sqsubseteq^2 A$ by item 1 of Lemma 2. In case $o = t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$ we have $\delta(o) = T$ and $\delta^2(o) = C^2$. By item 2 of Lemma 2, $C^2 \sqsubseteq^2 A$ is equivalent to the disjunction of $T \sqsubseteq^1_C A$ or $C \sqsubseteq^1_C A$. Again by Lemma 2, this is equivalent to $T \sqsubseteq^1 A$ or $\sigma(t_1) \sqsubseteq^1 A$ or $\sigma(t_2) \sqsubseteq^1 A$. Since $T \sqsubseteq \sigma(t_1), \sigma(t_2)$, this is equivalent to $T \sqsubseteq^1 A$. In total, we have shown (1) by proving that $C^2 \sqsubseteq^2 A$ is equivalent to $T \sqsubseteq^1 A$.

We need to convince ourselves that $\mathcal{M}^2 \models \bigwedge \Gamma \land t_1 \doteq t_2$ is still true. We will prove the following auxiliary statement:

$$\begin{array}{c}\text{Let } \phi \text{ be an arbitrary } (\mathcal{T}, \Sigma)\text{-formula, } \beta \text{ a variable assignment, then} \\ (\mathcal{M}, \beta) \models \phi \quad \Leftrightarrow \quad (\mathcal{M}^2, \beta) \models \phi \end{array} \tag{2}$$

The proof of (2) proceeds by induction on the complexity of ϕ. The only non-trivial steps are the induction steps for quantifiers. So assume that the claim is true for $\phi(x_1, \dots, x_n)$[1] and we try to establish it for $(\exists x_1.\phi)(x_2, \dots, x_n)$, with x_1 a variable of type A. By choice of ϕ, the type A is different from C and C^2.

$$\begin{aligned}(\mathcal{M}, \beta) \models \exists x_1.\phi &\Leftrightarrow (\mathcal{M}, \beta^o_{x_1}) \models \phi(x_1) && \text{for } o \in M \text{ with } \delta(o) \sqsubseteq A \\ &\Leftrightarrow (\mathcal{M}^2, \beta^o_{x_1}) \models \phi(x_1) && \text{induction hypothesis} \\ &\Leftrightarrow (\mathcal{M}^2, \beta) \models \exists x_1.\phi && \text{by (1)}\end{aligned}$$

Now, we have established $\mathcal{M}^2 \models \bigwedge \Gamma \wedge t_1 \doteq t_2$. From the assumption we obtain $\mathcal{M}^2 \models \bigvee \Delta$, which entails $\mathcal{M} \models \bigvee \Delta$ by another appeal to (2), as desired.

Completeness. The completeness part of the proof proceeds by contradiction. Assume there is no closed proof tree with root labeled by $\Gamma \Longrightarrow \Delta$. We will eventually construct a $(\mathcal{T}', \Sigma)$-structure $\mathcal{M} = (H, \delta, I)$ that is a counterexample to the universal validity of $\Gamma \Longrightarrow \Delta$.

Let T be a proof tree with root labeled by $\Gamma \Longrightarrow \Delta$ such that all rules have been exhaustively applied, but T is not closed. Because of the rules $\mathsf{allLeft}$ and $\mathsf{exRight}$, T is necessarily infinite. By an appeal to König's Lemma there is an infinite branch B of T that is not closed.

Let H_0 be the set of all ground terms. We define the relation $\sim_B$ on H_0 by

$$\begin{aligned}t_1 \sim_B t_2 \text{ iff } t_1 = t_2 \text{ or}& \\ &\text{there is a sequent } \Gamma \Longrightarrow \Delta \text{ in } B \text{ with } t_1 \doteq t_2 \in \Gamma\end{aligned}$$

The relation $\sim_B$ is an equivalence relation. Reflexivity is assured by definition. If $t_1 \sim_B t_2$ with $t_1 \doteq t_2 \in \Gamma$ and $\Gamma \Longrightarrow \Delta \in B$, then somewhere in B the rule $\mathsf{eqSymmLeft}$ must have been applied since we assume exhaustive rule application. Thus there will be a sequent $\Gamma' \Longrightarrow \Delta' \in B$ with $t_2 \doteq t_1 \in \Gamma'$ and we arrive at $t_2 \sim_B t_1$. It remains to show transitivity. We start from $t_1 \sim_B t_2$ and $t_2 \sim_B t_3$. By definition of $\sim_B$ there are sequents $\Gamma_1 \Longrightarrow \Delta_1$ and $\Gamma_2 \Longrightarrow \Delta_2$ in B with $t_1 \doteq t_2 \in \Gamma_1$ and $t_2 \doteq t_3 \in \Gamma_2$. Since there is no rule that drops an equality in the antecedent, only the arguments may be swapped, there will be a sequent $\Gamma' \Longrightarrow \Delta'$ in B such that $t_1 \doteq t_2$ or $t_2 \doteq t_1$ and at the same time $t_2 \doteq t_3$ or $t_3 \doteq t_2$ occur in Γ'. We consider each case separately.

1. $t_1 \doteq t_2$ and $t_2 \doteq t_3$
2. $t_1 \doteq t_2$ and $t_3 \doteq t_2$
3. $t_2 \doteq t_1$ and $t_2 \doteq t_3$
4. $t_2 \doteq t_1$ and $t_3 \doteq t_2$

In case (1) we use eqLeft to replace the left-hand side t_2 of the second equation by its right-hand side in the first equation and obtain $t_1 \doteq t_3$.

In case (3) we replace, using eqLeft, the left-hand side t_2 of the first equation by its right-hand side in the second equation and obtain $t_1 \doteq t_3$.

In case (4) replace the left-hand side t_2 of the first equation by its right-hand side in the second equation and obtain $t_3 \doteq t_1$. Another application of $\mathsf{eqSymmLeft}$ yields $t_1 \doteq t_3$.

[1] i.e., a formula with at most the free variables $x_1, \dots, x_n$

Case (2) is the most involved. By exhaustiveness of B we know that eqSymmLeft will be applied again to both equations in focus. If it is first applied to the first equation, we obtain the situation in case (4). If it is first applied to the second equation, we obtain case (1).

Thus in any case $t_1 \sim_B t_3$ follows and transitivity is established. In total we know now that $\sim_B$ is an equivalence relation.

Next we aim to show that $\sim_B$ is also a congruence relation. This requires first to show that $t_i \sim_B t'_i$ for $1 \leq i \leq n$ implies $f(t_1, \ldots, t_n) \sim_B f(t'_1, \ldots, t'_n)$ for any n-place function symbol f. For simplicity we only present that case of a unary function symbol f. We need to show $f(t) \sim_B f(t')$ from $t \sim_B t'$. We first take on the case that $\sigma(t)$ and $\sigma(t')$ are comparable, e.g., $\sigma(t') \sqsubseteq \sigma(t)$. By assumption there is a sequent $\Gamma \Longrightarrow \Delta$ in branch B with $t \doteq t' \in \Gamma$. From the argument given above, we know that there is also sequent $\Gamma_1 \Longrightarrow \Delta_1$ on B with $f(t) \doteq f(t) \in \Gamma_1$ and $t \doteq t' \in \Gamma_1$. By rule eqLeft we obtain a sequent $\Gamma_2 \Longrightarrow \Delta_2$ on B with $f(t) \doteq f(t') \in \Gamma_2$, and thus $f(t) \sim_B f(t')$. It remains to deal with the case that $\sigma(t)$ and $\sigma(t')$ are incomparable. Then rule eqDynamicSort applies and yields a sequent $\Gamma_3 \Longrightarrow \Delta_3$ on B with $t \doteq t' \in \Gamma_3$ and also $\exists x.(x \doteq t \wedge x \doteq t') \in \Gamma_3$ with x a variable of the new type C, with $C \sqsubseteq \sigma(t)$ and $C \sqsubseteq \sigma(t')$. By exLeft there is a Skolem symbol sk of type C with $sk \sim_B t$ and $sk \sim_B t'$. By the comparable types case of the congruence property already established we obtain $f(sk) \sim_B f(t)$ and $f(sk) \sim_B f(t')$. In total $f(t) \sim_B f(t')$ as desired. The case of arbitrary n-place function symbols is only marginally more complicated.

To show that $\sim_B$ is a congruence relation requires to verify the second claim for any n-place predicate symbol q: if $t_i \sim_B t'_i$ for $1 \leq i \leq n$ and $q(t_1, \ldots, t_n) \in \Gamma$ for some $\Gamma \Longrightarrow \Delta$ in B then also $q(t'_1, \ldots, t'_n) \in \Gamma'$ for some $\Gamma' \Longrightarrow \Delta'$ in B. This follows easily from repeated application of the eqLeft rule.

By $[t]_B$ for $t \in H_0$ we denote the equivalence class of t with respect to $\sim_B$, i.e., $[t]_B = \{s \in H_0 \mid t \sim_B s\}$. The universe of the intended counterexample can be stated as:

$$H = \{[t]_B \mid t \in H_0\}$$

Next we need to decide what (dynamic) type an element $[t]_B$ should have in the structure to be constructed. We call an equivalence class $[t]_B$ *typed* if there is a type $T_0 \in \mathcal{T}$ such that there is an term $t_0 \in [t]_B$ with $\sigma(t_0) = T_0$ and for all $t' \in [t]_B$ the subtype relation $T_0 \sqsubseteq \sigma(t')$ holds true. For every equivalence class $[t]_B$ that is not typed, we introduce a new type constant $T_{[t]}$ and set

$$\begin{aligned} \Delta &= \{T_{[t]} \mid [t]_B \in H \text{ is not typed}\} \\ \Delta_R &= \{T_{[t]} \sqsubseteq \sigma(t') \mid t' \in [t]_B \text{ and } T_{[t]} \in \Delta\} \end{aligned}$$

The type hierarchy $\mathcal{T}' = (\text{TSym}', \sqsubseteq')$ extending $\mathcal{T} = (\text{TSym}, \sqsubseteq)$ is given by $\text{TSym}' = \text{TSym} \cup \Delta$ and $\sqsubseteq'$ the least subtype relation containing $\sqsubseteq \cup \Delta_R$. Obviously, $\mathcal{T} \sqsubseteq \mathcal{T}'$.

We are now ready to define a first-order $(\mathcal{T}', \Sigma)$-structure $\mathcal{M} = (H, \delta, I)$.

The (dynamic) typing function is given by

$$\delta([t]_B) = \begin{cases} T_0 & \text{if } [t]_B \text{ is typed by } T_0 \\ T_{[t]} & \text{the new type constant, otherwise} \end{cases}$$

For any n-place function symbol f, we set $I(f)([t_1]_B, \ldots, [t_n]_B) = [f(t_1, \ldots, t_n)]_B$. Since $\sim_B$ is a congruence relation this is an unambiguous definition.

For any n-place predicate symbol p we set

$$I(p) = \{([t_1]_B, \ldots, [t_n]_B) \mid \text{ a sequent } \Gamma, p(t_1, \ldots, t_n) \Longrightarrow \Delta \text{ occurs in } B\}$$

Again we have to argue that this definition is unambiguous. We do this again for the special case $n = 1$. The generalization to arbitrary n is left as an easy exercise to the reader. If $\Gamma, p(t) \Longrightarrow \Delta$ occurs in B and $t \sim_B s$ we need to show that also a sequent $\Gamma', p(s) \Longrightarrow \Delta'$ occurs in B. We observe first that there is no rule that removes or changes an atomic formula occurring in a sequent. Even in eqLeft and eqRight the substituted formula is added. Therefore we will have a sequent $\Gamma'', t \doteq s, p(t) \Longrightarrow \Delta''$ in B. An application of eqLeft now completes the argument.

This completes the definition of the structure $\mathcal{M} = (H, \delta, I)$.

$$I(t) = [t]_b \text{ for every ground term } t. \tag{3}$$

For 0-place function symbols c claim (3) is just the definition of $I(c)$. The rest of the claim follows by an easy induction on the structural complexity of t.

The next phase in the proof consists in the verification of the claim

$$\mathcal{M} \models \bigwedge \Gamma \wedge \neg \bigvee \Delta \text{ for all sequents } s = \Gamma \Longrightarrow \Delta \text{ in } B \tag{4}$$

The proof of claim (4) is reduced to the following

$$\begin{array}{l} \text{For every formula } \phi \\ \text{if there is } \Gamma \Longrightarrow \Delta \in B \text{ with } \phi \in \Gamma \text{ then } \mathcal{M} \models \phi \\ \text{if there is } \Gamma \Longrightarrow \Delta \in B \text{ with } \phi \in \Delta \text{ then } \mathcal{M} \not\models \phi \end{array} \tag{5}$$

Claim (5) is proved by induction on the structural complexity $n(\phi)$ of ϕ. If $n(\phi) = 0$ then ϕ is an atomic formula or an equation.

For an atomic formula $p(\bar{t}) \in \Gamma$ we know by definition of $\mathcal{M}$ that $\mathcal{M} \models p(\bar{t})$. Now, consider $p(\bar{t}) \in \Delta$. If $\mathcal{M} \models p(\bar{t})$ then there must by definition of $\mathcal{M}$ be a sequent $\Gamma' \Longrightarrow \Delta'$ in B with $p(\bar{t}) \in \Gamma'$. Since atomic formulas never get removed, we must have either $p(\bar{t}) \in \Delta$ and $p(\bar{t}) \in \Gamma$ or $p(\bar{t}) \in \Delta'$ and $p(\bar{t}) \in \Gamma'$. In both cases the branch B could be closed, contrary to assumption. Thus we must have $\mathcal{M} \models \neg p(\bar{t})$ for all $p(\bar{t}) \in \Delta$.

For $t_1 \doteq t_2 \in \Gamma$ we get $t_1 \sim_B t_2$ by definition of $\sim_B$. Thus $[t_1]_B = [t_2]_B$ which directly yields $\mathcal{M} \models t_1 \doteq t_2$.

For $t_1 \doteq t_2 \in \Delta$ we need to show $t_1 \not\sim_B t_2$. But, if $t_1 \sim_B t_2$ were true, there would by definition of $\sim_B$ be a sequent $\Gamma \Longrightarrow \Delta$ in branch B with $t_1 \doteq t_2 \in \Gamma$. Since atomic formulas never get erased there would also be a sequent $\Gamma' \Longrightarrow \Delta'$ with $t_1 \doteq t_2 \in \Gamma'$ and $t_1 \doteq t_2 \in \Delta'$ and the branch could be closed contrary to assumption.

The inductive step $n(\phi) > 0$ is split into a total of 12 cases. Since this part of the proof follows a well established pattern, we restrict our presentation to two exemplary cases.

Case A $\phi_1 \wedge \phi_2$ in Γ.
Since branch B is assumed to be *exhausted*, rule andLeft will have been applied. There is thus a sequent $\Gamma' \Longrightarrow \Delta'$ in B with $\phi_1, \phi_2 \in \Gamma'$. By induction hypothesis we know $\mathcal{M} \models \phi_1$ and $\mathcal{M} \models \phi_2$ thus $\mathcal{M} \models \phi_1 \wedge \phi_2$.

Case B $\exists x.\phi$ in Γ.
Since branch B is assumed to be *exhausted* rule exLeft will have been applied. There is thus a sequent $\Gamma' \Longrightarrow \Delta'$ in B with $[x/c]\phi \in \Gamma'$. By induction hypothesis we know $\mathcal{M} \models [x/c]\phi$ and thus also $\mathcal{M} \models \exists x.\phi$. $\square$

3 A Basic Theory in Typed Logic

In this section, we introduce the concept of a *basic theory* that may be useful in many application contexts.

Definition 9. *Let $\mathcal{T}$ be a type hierarchy, Σ a signature.*
A $(\mathcal{T}, \Sigma)$ theory T is called a basic theory *if*

- *Σ contains at least for each $A \in \mathcal{T}$, $A \neq \top, \bot$*
  - *The unary predicate symbols $instance_A : \top$ and $exactInstance_A : \top$*
  - *The function symbol $cast_A : \top \to A$*
  - *The constant symbol $default_A : A$.*

  Such Σ will be called a basic signature.
- *T contains at least the following axiom schemes $T_{\mathcal{T}}^{base}$*

$$
\begin{array}{rll}
1. & \forall x.(instance_A(x) \leftrightarrow \exists y.(y \doteq x)) \text{ with } y : A & \text{(Ax-I)} \\
2a. & \forall x.(exactInstance_A(x) \to instance_A(x)) & \text{(Ax-E}_1\text{)} \\
2b. & \forall x.(exactInstance_A(x) \to \neg instance_B(x)) \text{ with } A \not\sqsubseteq B & \text{(Ax-E}_2\text{)} \\
3. & \forall x.(\ (\ \ instance_A(x) \to cast_A(x) \doteq x) \wedge & \text{(Ax-C)} \\
 & \quad (\neg instance_A(x) \to cast_A(x) \doteq default_A)\ ) &
\end{array}
$$

A, B range over $\mathrm{TSym} \setminus \{\bot\}$ *and $x : \top$ is a variable of the universal sort $\top$.*

Definition 10. *Let Σ be a basic signature. A $(\mathcal{T}, \Sigma)$-structure $\mathcal{M} = (M, \delta, I)$ is called a* standard *structure if*

$$
\begin{array}{lll}
1. & instance_A^{\mathcal{M}} & = \{o \in M \mid \delta(o) \sqsubseteq A\} = M^A \\
2. & exactInstance_A^{\mathcal{M}} & = \{o \in M \mid \delta(o) = A\} \\
3. & cast_A^{\mathcal{M}}(o) & = \begin{cases} o & \text{if } o \in M^A \\ default_A^{\mathcal{M}} & \text{otherwise} \end{cases}
\end{array}
$$

There are at this point no restrictions on $default_A^{\mathcal{M}}$ except, of course, that it be an element of M^A.

Theorem 2.

1. *Let $\mathcal{M}$ be a standard $(\mathcal{T},\Sigma)$-structure for basic signature Σ.
Then $\mathcal{M} \models T^{base}_{\mathcal{T}}$.*
2. *Let $\mathcal{M}$ be a $(\mathcal{T},\Sigma)$-structure for basic signature Σ and $\mathcal{M} \models T^{base}_{\mathcal{T}}$.
Then $o \in exactInstance^{\mathcal{M}}_{A} \implies \delta(o) = A$.*

Proof. **ad 1**
Parts (1) and (3) of Definition 9 are direct formalization of the definitions of $instance_A$ and $cast_A$ in standard structures in Definition 10. Part (2a) is also an obvious consequence of the semantics of $exactInstance_A$. So let us turn to part (2b) and consider $o \in exactInstance^{\mathcal{M}}_{A}$ and a type B with $A \not\sqsubseteq B$. By the standard semantics definition this says $\delta(o) = A$ and $\delta(o) \not\sqsubseteq B$ and thus $o \notin instance^{\mathcal{M}}_{B}$. This proves $\mathcal{M} \models (2b)$.
ad 2
Since the first part of axiom (2) in $T^{base}_{\mathcal{T}}$ entails $exactInstance^{\mathcal{M}}_{A} \subseteq instance^{\mathcal{M}}_{A}$, we obtain $\delta(o) \sqsubseteq A$ from the definition of $instance_A$ in $T^{base}_{\mathcal{T}}$. If $\delta(o) \neq A$ axiom (2b) would yield $o \notin exactInstance^{\mathcal{M}}_{\delta(o)}$ contradicting (2) of Definition 10. □

If a $(\mathcal{T},\Sigma)$-structure $\mathcal{M}$ satisfies $\mathcal{M} \models T^{base}_{\mathcal{T}}$, then it need not be a standard structure, i.e., the reverse implication in Proposition 2 (2) need not hold. The axioms would, e.g., be true if $exactInstance^{\mathcal{M}}_{A} = \emptyset$ for all A. We will nevertheless be able to prove that the sentences derivable from $T^{base}_{\mathcal{T}}$ are exactly those universally valid in all standard structures. This needs the following preparatory definitions and lemma.

For an arbitrary extension $\mathcal{T}^* = (\mathrm{TSym}^*, \sqsubseteq^*)$ of $\mathcal{T} = (\mathrm{TSym}, \sqsubseteq)$ and signature Σ for the hierarchy $\mathcal{T}$ we construct for any $(\mathcal{T}^*,\Sigma^*)$-structure $\mathcal{M}$ with $\mathcal{M} \models T^{base}_{\mathcal{T}}$ an adapted structure $\mathcal{M}^a$ that is standard and not too far away from $\mathcal{M}$. $\mathcal{M}^a$ will be a $(\mathcal{T}^a,\Sigma^a)$ structure for an extension $\mathcal{T}^a$ of the hierarchy $\mathcal{T}^*$. The signatures Σ^* and Σ^a will at least contain all the symbols from Definition 9 relating to the new types not contained in TSym. In passing from $\mathcal{M}$ to $\mathcal{M}^a$ some elements of the universe need to be "relocated" into different types. We will do this using a partial type projection $\pi_{\mathcal{M},\mathcal{T}} : M \rightarrow\!\!\!\!+ \mathrm{TSym}$ which is characterized by

$$\pi_{\mathcal{M},\mathcal{T}}(o) = A \iff \begin{array}{l} \quad o \in exactInstance^{\mathcal{M}}_{A} \\ \text{and } \delta(o) \sqsubseteq A \\ \text{and } (\delta(o) \sqsubseteq B \implies A \sqsubseteq B) \text{ for all } B \in \mathrm{TSym}. \end{array} \tag{6}$$

Function π is well-defined: Assume there are two $A, A' \in \mathrm{TSym}$ for which the right-hand side of the above definition is true. By the third condition we have that $A \sqsubseteq A'$ and $A' \sqsubseteq A$, hence, ($\mathcal{T}$ is a poset) $A = A'$.

The idea behind it is that π maps element o to the type it appears to live in ($o \in exactInstance^{\mathcal{M}}_{A}$) when looking at it from the perspective of $\mathcal{T}$ only. The additional conditions make this well-defined and ensure that domains in the adapted structure remain the same.

Definition 11. *Let $\mathcal{T}^* = (\mathrm{TSym}^*, \sqsubseteq^*)$ be an extension of the type hierarchy $\mathcal{T} = (\mathrm{TSym}, \sqsubseteq)$, Σ a basic signature for hierarchy $\mathcal{T}$, and $\mathcal{M} = (M, \delta, I)$ a $(\mathcal{T}^*, \Sigma^*)$-structure.*
The adapted structure *$\mathcal{M}^a = (M, \delta^a, I^a)$ for $\mathcal{M}$ is the $(\mathcal{T}^a, \Sigma^a)$ structure with*

$$\begin{aligned}
\mathcal{T}^a &= (\mathrm{TSym}^a, \sqsubseteq^a)\\
\mathrm{TSym}^a &= \mathrm{TSym} \cup \{T_o \mid o \notin \mathrm{dom}\,\pi_{\mathcal{M},\mathcal{T}}\} \textit{ for new symbols } T_o\\
\sqsubseteq^a &= \textit{transitive closure of } \sqsubseteq_{\mathcal{T}} \cup\\
&\qquad \{(T_o, A) \mid o \notin \mathrm{dom}\,\pi_{\mathcal{M},\mathcal{T}}, A \in \mathrm{TSym}, \delta(o) \sqsubseteq A\}\\
\Sigma^a &= \Sigma^* \cup \{\mathit{instance}_C, \mathit{exactInstance}_C, \mathit{cast}_C, \mathit{default}_C \mid\\
&\qquad C \in \mathrm{TSym}^a \setminus \mathrm{TSym}^*\}\\
\delta^a(o) &= \begin{cases} A & \textit{if } \pi_{\mathcal{M},\mathcal{T}}(o) = A\\ T_o & \textit{if } o \notin \mathrm{dom}\,\pi_{\mathcal{M},\mathcal{T}} \end{cases}\\
I^a(f) &= I(f) \textit{ for symbols } f \in \Sigma\\
I^a(\mathit{instance}_{T_o}) &= I^a(\mathit{exactInstance}_{T_o}) = \{o\} \textit{ for all } o \notin \mathrm{dom}\,\pi_{\mathcal{M},\mathcal{T}}\\
I^a(\mathit{cast}_{T_o})(x) &= I^a(\mathit{default}_{T_o}) = o \textit{ for all } o \notin \mathrm{dom}\,\pi_{\mathcal{M},\mathcal{T}}
\end{aligned}$$

Lemma 3.

1. *$A^{\mathcal{M}^a} = A^{\mathcal{M}}$ for $A \in \mathrm{TSym}$.*
2. *If $\mathcal{M} \models T^{base}_{\mathcal{T}}$ then $\quad(\pi_{\mathcal{M},\mathcal{T}}(o) = A \iff o \in \mathit{exactInstance}^{\mathcal{M}}_A)$*

Property 1 is necessary for the construction of $\mathcal{M}^a$ to be well-defined. If domains had changed, e.g., $A^{\mathcal{M}^a} \neq A^{\mathcal{M}}$, the definition $I^a(f) = I(f)$ would not make sense.

Proof. **ad 1**

$\subseteq$ Assume $o \in A^{\mathcal{M}^a}$, that is $\delta^a(o) = B \sqsubseteq^a A$ for some $B \in \mathrm{TSym}^a$. If $B = T_o$, then $T_o \sqsubseteq^a A$ implies by the definition of $\sqsubseteq^a$ (see part (2) of Lemma 2) that there must be a type $C \in \mathrm{TSym}$ with $\delta(o) \sqsubseteq C$ and $C \sqsubseteq A$. Hence, also $\delta(o) \sqsubseteq A$, i.e., $o \in A^{\mathcal{M}}$. If $B = \pi_{\mathcal{M},\mathcal{T}}(o)$, then $\delta(o) \sqsubseteq A$ by definition of $\pi_{\mathcal{M},\mathcal{T}}$.

$\supseteq$ Assume $o \in A^{\mathcal{M}}$, that is $\delta(o) = B \sqsubseteq A$ for some $B \in \mathrm{TSym}^*$. If $o \in \mathrm{dom}\,\pi_{\mathcal{M},\mathcal{T}}$, then $\pi_{\mathcal{M},\mathcal{T}}(o)$ is the $\sqsubseteq$-smallest supertype in TSym covering B. Hence, $\delta^a(o) = \pi_{\mathcal{M},\mathcal{T}}(o) \sqsubseteq A$. Part (1) of Lemma 2 yields $\delta^a(o) \sqsubseteq^a A$ and so $o \in A^{\mathcal{M}^a}$. On the other hand, if $o \notin \mathrm{dom}\,\pi_{\mathcal{M},\mathcal{T}}$, then $\delta^a(o) = T_o$ and $T_o \sqsubseteq^a A$ (by definition of $\sqsubseteq^a$). Again, $\delta^a(o) \sqsubseteq^a A$ and $o \in A^{\mathcal{M}^a}$.

ad 2
We show that under the assumption of the axioms, the first condition in (6) implies the other two. Choose $o \in \mathit{exactInstance}^{\mathcal{M}}_A$ in the following.

Axiom (Ax-E$_1$) ensures that $o \in \mathit{instance}^{\mathcal{M}}_A$, and axiom (Ax-I) that $\delta(o) \sqsubseteq A$. (see later more details...)

Assume that the third condition were violated, that is, there is $B \in \mathrm{TSym}$ with $\delta(o) \sqsubseteq B$ and $A \not\sqsubseteq B$. But this allows us to use axiom (Ax-E$_2$) to obtain $o \notin \mathit{instance}_B^{\mathcal{M}}$ and (again by axiom (Ax-I)) that $\delta(o) \not\sqsubseteq B$. Contradiction. □

Lemma 4. *Let Σ, $\mathcal{T}$, $\mathcal{T}^*$, $\mathcal{T}^a$ be as in Definition 11 and $\mathcal{M}$ a $(\mathcal{T}^*, \Sigma^*)$-structure satisfying $\mathcal{M} \models T_{\mathcal{T}}^{base}$.*

1. *The* adapted *structure $\mathcal{M}^a$ of $\mathcal{M}$ is a standard structure and*
2. *$\mathcal{M} \models \varphi \iff \mathcal{M}^a \models \varphi$ for all $(\mathcal{T}, \Sigma)$-formulas φ.*

Proof.
ad 1. To argue that $\mathcal{M}^a$ is a standard structure we look separately at the three conditions of Definition 10, where A ranges of all type symbols in TSym^a.

1. $\mathit{instance}_A^{\mathcal{M}^a} = \{o \in M \mid \delta^a(o) \sqsubseteq^a A\}$
   We have already observed that we have for all $A \in \mathrm{TSym}$
   $$A^{\mathcal{M}^a} = \{o \in M \mid \delta^a(o) \sqsubseteq^a A\} = \{o \in M \mid \delta(o) \sqsubseteq A\} = A^{\mathcal{M}}.$$
   For $A \in \mathrm{TSym}$ we know $\mathcal{M} \models \forall x(\mathit{instance}_A(x) \leftrightarrow \exists y.(y \doteq x))$ from axiom (Ax-I). Thus $\mathit{instance}_A^{\mathcal{M}} = \{o \in M \mid \delta(o) \sqsubseteq A\}$. By definition of $\mathcal{M}^a$ we have $\mathit{instance}_A^{\mathcal{M}^a} = \mathit{instance}_A^{\mathcal{M}}$. Together with the initial observation this proves what we want.
   It remains to consider types T_o for $o \notin \operatorname{dom} \pi_{\mathcal{M},\mathcal{T}}$. By definition $\mathit{instance}_{T_o}^{\mathcal{M}^a} = \{o\}$.
   $\{o' \in M \mid \delta^a(o') \sqsubseteq^a T_o\} = \{o' \in M \mid \delta^a(o') = T_o\}$ Lemma 2(3)
   $\qquad\qquad\qquad\qquad\qquad\;\; = \{o' \in M \mid o' = o\}$ Def. of δ^a
2. $\mathit{exactInstance}_A^{\mathcal{M}^a} = \{o \in M \mid \delta^a(o) = A\}$
   For $A \in \mathrm{TSym}$ the valuation $\mathit{exactInstance}_A^{\mathcal{M}^a}$ is the same as $\mathit{exactInstance}_A^{\mathcal{M}}$
   From Lemma 3(2), we obtained that $\pi_{\mathcal{M},\mathcal{T}}(o) = A \iff o \in \mathit{exactInstance}_A^{\mathcal{M}^a}$.
   Let $o \in \mathit{exactInstance}_A^{\mathcal{M}^a}$ be given. The implication from right to left gives us that $\pi_{\mathcal{M},\mathcal{T}}(o) = A$ and also $\delta^a(o) = A$ (since $o \in \operatorname{dom} \pi_{\mathcal{M},\mathcal{T}}$).
   For the opposite direction, assume now that $\delta^a(o) = A$. Since $A \in \mathrm{TSym}$, it must be that $o \in \operatorname{dom} \pi_{\mathcal{M},\mathcal{T}}$ and $\pi_{\mathcal{M},\mathcal{T}}(o) = A$. The implication from right to left entails $o \in \mathit{exactInstance}_A^{\mathcal{M}^a}$.
   Finally, if $T_o \in \mathrm{TSym}^a \setminus \mathrm{TSym}$ is a type introduced in the adapted type system for $o \notin \operatorname{dom} \pi_{\mathcal{M},\mathcal{T}}$, then (by definition of δ^a) o is the only element of that type, and $I^a(\mathit{exactInstance})$ is defined accordingly.
3. $\mathit{cast}_A^{\mathcal{M}^a}(o) = \begin{cases} o & \text{if } o \in A^{\mathcal{M}^a} \\ \mathit{default}_A^{\mathcal{M}^a} & \text{otherwise} \end{cases}$
   For a type $A \in \mathrm{TSym}$, the domain $A^{\mathcal{M}^a} = A^{\mathcal{M}}$ has *not* changed; the definition of δ^a reveals that some elements o may now have a new dynamic type T_o which is a subtype of A, but this does not modify the extension of the type. We can use axiom (Ax-C) to show that the semantics of the cast is precisely the required. We can use the fact that $A^{\mathcal{M}} = \mathit{instance}_A^{\mathcal{M}^a}$ established in item 1 and leave the proof as an easy exercise.
   Again for the types not already present in TSym, the definition of I^a fixes the semantics of the cast symbols correctly.

ad 2.

For the evaluation of a formula, the adaptation $\mathcal{M}^a$ is indistinguishable from the original $\mathcal{M}$. Keep in mind that the syntactical material for φ is that of $(\mathcal{T}, \Sigma)$, i.e., neither the types in $\mathrm{TSym}^* \setminus \mathrm{TSym}, \mathrm{TSym}^a \setminus \mathrm{TSym}$ nor the corresponding function and predicate symbols will appear in φ.

Proof by structural induction over quantifications:

– For any quantifier-free φ we have that $\mathcal{M}, \beta \models \varphi \iff \mathcal{M}^a, \beta \models \varphi$. This is a direct consequence of the fact that functions and predicates in Σ are interpreted identically in $\mathcal{M}$ and $\mathcal{M}^a$.
– Let $\forall x{:}A.\ \varphi$ be a universally quantified formula for $A \in \mathrm{TSym}$. We have:

$$\begin{aligned} & \mathcal{M}, \beta \models \forall x.\ \varphi \\ \iff & \mathcal{M}, \beta_x^o \models \varphi \text{ for all } o \in A^{\mathcal{M}} \\ \iff & \mathcal{M}^a, \beta_x^o \models \varphi \text{ for all } o \in A^{\mathcal{M}} \quad \text{(induction hypothesis)} \\ \overset{(*)}{\iff} & \mathcal{M}^a, \beta_x^o \models \varphi \text{ for all } o \in A^{\mathcal{M}^a} \\ \iff & \mathcal{M}^a, \beta \models \forall x.\ \varphi \end{aligned}$$

The essential point is $(*)$ relying upon that quantifiers range over the same domains in $\mathcal{M}$ and $\mathcal{M}^a$. We have observed this already in the proof of the first point of this proposition.
The case for the existential quantifier is completely analogous.

The next lemma claims that $T_{\mathcal{T}}^{base}$ is a complete axiomatization of standard structures.

Theorem 3. *Let Σ be a basic signature, $\mathcal{T}$ an arbitrary type hierarchy, and ϕ a $(\mathcal{T}, \Sigma)$ sentence. Then*

$$\begin{aligned} T_{\mathcal{T}}^{base} \models \phi \Leftrightarrow & \textit{ for all extensions } \mathcal{T}^* \sqsupseteq \mathcal{T} \textit{ and } (\mathcal{T}^*, \Sigma) \textit{ standard structures } \mathcal{M} \\ & \mathcal{M} \models \phi \end{aligned}$$

Proof. For the implication $\Rightarrow$ from left to right we assume $T_{\mathcal{T}}^{base} \models \phi$ and fix an extension hierarchy $\mathcal{T}^* \sqsupseteq \mathcal{T}$ and a $(\mathcal{T}^*, \Sigma)$ standard structure $\mathcal{M}$ with the aim of showing $\mathcal{M} \models \phi$. We will succeed if we can show $\mathcal{M} \models T_{\mathcal{T}}^{base}$, which is Theorem 2(1)).

For the reverse implication, $\Leftarrow$, we assume the right-hand condition and fix an extension $\mathcal{T}^* \sqsupseteq \mathcal{T}$ and a $(\mathcal{T}^*, \Sigma)$-structure $\mathcal{M}$ with $\mathcal{M} \models T_{\mathcal{T}}^{base}$. We want to arrive at $\mathcal{M} \models \phi$. Let $\mathcal{M}^a$ be the adapted structure for $\mathcal{M}$ as in Definition 11. By the first part of Proposition 4 we know that $\mathcal{M}^a$ is a standard structure. By assumption this implies $\mathcal{M}^a \models \phi$. By the second part of Proposition 4, we get $\mathcal{M} \models \phi$. □

4 Towards a Java Theory

In this section we provide a few hints how theory $T_{\mathcal{T}}^{base}$ can be instantiated and extended to a theory T_J suitable for reasoning about a real Java program Π.

The type hierarchy $\mathcal{T}_J$ will consist of the classes occurring in Π with the subclass ordering plus possibly some abstract data types. One might wish to fix certain default elements by adding e.g., $default_{Object} = null$ and $default_{boolean} = false$ to the theory.

It will also be useful to fix certain properties of the type hierarchy, e.g., that *int* and *Object* are disjoint types. This can be done by adding $\neg\exists x.(instance_{int}(x) \wedge instance_{Object}(x))$ to T_J. As another example, one may want to formalize that *int* has no strict subtype. This is achieved by adding $\forall x.(instance_{int}(x) \rightarrow exactInstance_{int}(x))$ to T_J.

Martin Giese in [1, Chapter 2]. included from the start a distinction between abstract and non-abstract types, that he called dynamic types. In our setup we can define a type T to be abstract by the formula $\neg\exists x.(exactInstance_T(x))$, with x a variable of type *Object*.

5 Concluding Remarks

We point out that finiteness of TSym is not assumed for the completeness proof.

One might sum up the distinctive feature of our approach by noting that it allows us to convey typing information firstly in the way of syntax declarations. Thus, associating a unique static type with every term with the usual benefits. Secondly, typing information can be stated freely as axioms. Other approaches, e.g. [11] only offer the second possibility

A first-order theory for Java along the lines sketched in Section 4 but more expressive language has been implemented and used in the KeY system. This theory is, in fact, based on a logic that is richer than the one introduced in Section 2, contains e.g. conditional terms (if ϕ then t_1 else t_2) $\in \mathrm{Trm}_A$ for $\phi \in \mathrm{Fml}$ and $t_i \in \mathrm{Trm}_{A_i}$ such that $A_2 \sqsubseteq A_1 = A$ or $A_1 \sqsubseteq A_2 = A$.

References

1. B. Beckert, R. Hähnle, and P. H. Schmitt, editors. *Verification of Object-Oriented Software: The KeY Approach.* Number 4334 in Lecture Notes in Computer Science. Springer-Verlag, 2007.
2. F. Bobot, S. Conchon, E. Contejean, and S. Lescuyer. Implementing polymorphism in smt solvers. In *Proceedings of the Joint Workshops of the 6th International Workshop on Satisfiability Modulo Theories and 1st International Workshop on Bit-Precise Reasoning*, SMT '08/BPR '08, pages 1–5, New York, NY, USA, 2008. ACM.
3. F. Bobot and A. Paskevich. Expressing Polymorphic Types in a Many-Sorted Language. In Tinelli and Sofronie-Stokkermans [9], pages 87–102.
4. J. Esparza and R. Majumdar, editors. *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science.* Springer, 2010.

5. J. H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Wiley, 1987.
6. M. Giese. A calculus for type predicates and type coercion. In B. Beckert, editor, *TABLEAUX*, volume 3702 of *Lecture Notes in Computer Science*, pages 123–137. Springer, 2005.
7. K. R. M. Leino and P. Rümmer. A polymorphic intermediate verification language: Design and logical encoding. In Esparza and Majumdar [4], pages 312–327.
8. A. Schmidt. Über deduktive Theorien mit mehreren Sorten von Grunddingen. *Math. Annalen*, 115:485–506, 1938.
9. C. Tinelli and V. Sofronie-Stokkermans, editors. *Frontiers of Combining Systems, 8th International Symposium, FroCoS 2011, Saarbrücken, Germany, October 5-7, 2011. Proceedings*, volume 6989 of *Lecture Notes in Computer Science*. Springer, 2011.
10. C. Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation*. Pitman / Morgan Kaufmann, 1987.
11. C. Weidenbach. First-order tableaux with sorts. *Logic Journal of the IGPL*, 3(6):887–906, 1995.