A First-Order Logic with First-Class Types

Michael Walter
joint work with Peter H. Schmitt and Mattias Ulbrich

Institute for Theoretical Computer Science
University of Karlsruhe

The 18th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, 2009

Java Card DL

- modal logic behind KeY
- based on a typed first-order logic with subtyping, type predicates and casts [Gie05]

$$\forall x : \mathsf{Object}\ .\ x \sqsubseteq \mathsf{Array} \rightarrow \mathsf{length}((\mathsf{Array})x) \geq 0$$

- we focus only on this first-order part

Java Card DL

- modal logic behind KeY
- based on a typed first-order logic with subtyping, type predicates and casts [Gie05]

$$\forall x : \text{Object} \;.\; x \sqsubseteq \text{Array} \rightarrow \text{length}((\text{Array})x) \geq 0$$

- we focus only on this first-order part

Java Card DL

- modal logic behind KeY
- based on a typed first-order logic with subtyping, type predicates and casts [Gie05]

$$\forall x : \mathsf{Object} \; . \; x \sqsubseteq \mathsf{Array} \rightarrow \mathsf{length}((\mathsf{Array})x) \geq 0$$

- we focus only on this first-order part

JAVA CARD DL

- modal logic behind KeY
- based on a typed first-order logic with subtyping, type predicates and casts [Gie05]

$$\forall x : \mathrm{Object}\ .\ x \sqsubseteq \mathrm{Array} \rightarrow \mathrm{length}((\mathrm{Array})x) \geq 0$$

- we focus only on this first-order part

Java Card DL

- modal logic behind KeY
- based on a typed first-order logic with subtyping, type predicates and casts [Gie05]

$$\forall x : \mathrm{Object}\ .\ x \sqsubseteq \mathrm{Array} \rightarrow \mathrm{length}((\mathrm{Array})x) \geq 0$$

- we focus only on this first-order part

Java Generics

- classes parametrized by type parameters

```
public class Array<T>
{
  public T last();
}
```

- $\text{Array}\langle T \rangle \sqsubseteq \text{Array}\langle ? \rangle \sqsubseteq \text{Object}$
- what is the signature of last?

$$\{\, \text{last}_T : \text{Array}\langle T \rangle \to T \,\}$$

Java Generics

- classes parametrized by type parameters

```
public class Array<T>
{
  public T last();
}
```

- $\text{Array}\langle T\rangle \sqsubseteq \text{Array}\langle ?\rangle \sqsubseteq \text{Object}$
- what is the signature of last?

$$\{\, \text{last}_T : \text{Array}\langle T\rangle \to T \,\}$$

First-Class Types

$$\{\, \mathrm{last}_{T} : \mathrm{Array}\langle T \rangle \to T \,\}$$

how to reason about arrays without fixing the element type?

First-Class Types

$$\{\, \mathrm{last}_T : \mathrm{Array}\langle T\rangle \to T \,\}$$

how to reason about arrays without fixing the element type?

- single signature

$$\mathrm{last} : \mathrm{Array}\langle ?\rangle \to \top$$
$$\mathrm{T} : \mathrm{Array}\langle ?\rangle \to \mathbb{T}$$

  with type of all types $\mathbb{T}$

- need to assert that the return value has proper type

$$\forall a : \mathrm{Array}\langle ?\rangle \;.\; \mathrm{last}(a) \sqsubseteq T(a)$$

  with binary predicate $\sqsubseteq$

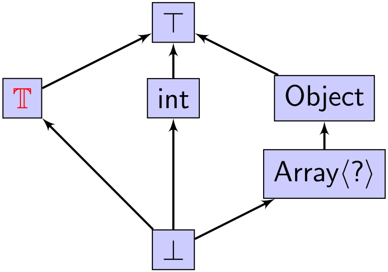
($\rightsquigarrow$ universal types)

Outline

Type Hierarchy

Definition (Type hierarchy)

- set of types $\mathcal{T}$
- subtype relation $\sqsubseteq$
- universal type $\top$ and empty type $\bot$
- greatest lower bounds ($\sqcap$)
- type of all types $\mathbb{T}$



Signature

Definition (Signature)

- predicate, function and variable symbols with types
- predefined symbols:
  - equality $\doteq : \top \times \top$
  - type predicate $\sqsubseteq : \top \times \mathbb{T}$
  - subtype predicate $\sqsubseteq : \mathbb{T} \times \mathbb{T}$

  - type intersection $\sqcap : \mathbb{T} \times \mathbb{T} \rightarrow \mathbb{T}$
  - type constants $T : \rightarrow \mathbb{T}$ (for each type $T \in \mathcal{T}$)
  - casts

Terms and Formulae

Definition (Term of type T)

- v if $v : T$ variable symbol
- $f(t_1, \ldots, t_n)$ if $f : T_1 \times \ldots \times T_n \to T$ function symbol, t_i term of type $T'_i \sqsubseteq T_i$

Definition (Formula)

- $p(t_1, \ldots, t_n)$ if ...
- $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \to \psi$
- $\forall v.\varphi$, $\exists v.\varphi$

Outline

Structure

Definition (Structure)

- domain $\mathcal{D}$
- dynamic typing function $\delta : \mathcal{D} \to \mathcal{T}$

$$\leadsto \mathcal{D}_T := \{x \in \mathcal{D} : \delta(x) \sqsubseteq T\}$$

- interpretation $\mathcal{I}$ of functions and predicates

$$\mathcal{I}(f) : \mathcal{D}_{T_1} \times \ldots \times \mathcal{D}_{T_n} \to \mathcal{D}_T$$
$$\mathcal{I}(p) \sqsubseteq \mathcal{D}_{T_1} \times \ldots \times \mathcal{D}_{T_n}$$

$\leadsto$ value of a term, validity of a formula...

Structure

Definition (Structure)

- domain $\mathcal{D}$
- dynamic typing function $\delta : \mathcal{D} \to \mathcal{T}$

$$\rightsquigarrow \mathcal{D}_T := \{x \in \mathcal{D} : \delta(x) \sqsubseteq T\}$$

- interpretation $\mathcal{I}$ of functions and predicates

$$\mathcal{I}(f) : \mathcal{D}_{T_1} \times \ldots \times \mathcal{D}_{T_n} \to \mathcal{D}_T$$
$$\mathcal{I}(p) \sqsubseteq \mathcal{D}_{T_1} \times \ldots \times \mathcal{D}_{T_n}$$

how about the predefined symbols?

$\rightsquigarrow$ value of a term, validity of a formula...

Interpretation

- $\mathcal{D}_{\mathbb{T}} = \mathcal{T}$
- predefined symbols shall agree with their type hierarchy counterpart:

$$\mathcal{I}(\sqsubseteq\!\!\!\in) \ni (x, T) \;\Leftrightarrow\; x \in \mathcal{D}_T \;\Leftrightarrow\; \delta(x) \sqsubseteq T$$
$$\mathcal{I}(\sqsubseteq) = \sqsubseteq,\ \mathcal{I}(T) = T, \ldots$$

Observation

If the type hierarchy is infinite then the logic has no sound and complete calculus. $\lightning$

Completeness and Compactness

Definition ((Strong) completeness)

$$\mathcal{A} \models \varphi \quad \Rightarrow \quad \mathcal{A} \vdash \varphi$$

Compactness Theorem

Every logic which has a sound and complete calculus is compact: If some set of formulae is not satisfiable then there exists a finite subset which is already not satisfiable.

Reasons for Noncompactness

two obstructions to compactness

1. constant symbols generate domain of $\mathbb{T}$

$$\{\neg(c \doteq T) \ : \ T \in \mathcal{T}\} \quad \lightning$$

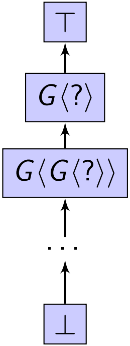
(for infinite $\mathcal{T}$; compare $\mathbb{N}$)

Reasons for Noncompactness

2. non-Noetherian type hierarchies

$$\{x \sqsubseteq G\langle ? \rangle, x \sqsubseteq G\langle G\langle ? \rangle\rangle, \ldots,$$
$$\neg(x \sqsubseteq \bot)\} \quad \lightning$$

(compare induction)



Theorem (Giese)

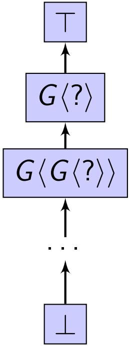
The logic of [Gie05] has a sound and complete calculus if and only if the type hierarchy is Noetherian.

Reasons for Noncompactness

2. non-Noetherian type hierarchies

$$\{x \sqsubseteq G\langle ?\rangle, x \sqsubseteq G\langle G\langle ?\rangle\rangle, \ldots, \\ \neg(x \sqsubseteq \bot)\} \quad \lightning$$

(compare induction)



Theorem (Giese)

The logic of [Gie05] has a sound and complete calculus if and only if the type hierarchy is Noetherian.

Interpretation – Modified

- require $\mathcal{D}_{\mathbb{T}}$ to be a type hierarchy that contains $(\mathcal{T}, \sqsubseteq)$
- predefined symbols shall extend their type hierarchy counterparts
- sanity conditions

Theorem

The modified logic has a sound and complete calculus if and only if the type hierarchy is Noetherian.

Outline

Conclusion

- characterized completeness of the logic of [Gie05]
- characterized completeness of first-class types

- first-class types are not useful on their own $\lightning$
  $\rightsquigarrow$ universal types, dependent types

Martin Giese.
A Calculus for Type Predicates and Type Coercion.
In Bernhard Becker, editor, *Proceedings of the 14th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2005)*, Lecture Notes in Artificial Intelligence, pages 123–137. Springer, 2005.