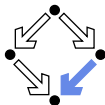
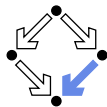


Logic

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
<http://www.risc.jku.at>





Take signature $\Sigma = (S, \Omega)$.

■ Variables:

- Family $V = (V_s)_{s \in S}$ of infinite sets disjoint with Ω and each other.
 - $V_s \dots$ the set of variables of sort s .
- Any family $X \subseteq V$ is called a **set of variables** for Σ .

■ Terms:

- Family $T_{\Sigma(X)} = (T_{\Sigma(X),s})_{s \in S}$ of terms with set of variables X for Σ .
 - Variables are terms: $X_s \subseteq T_{\Sigma(X),s}$.
 - Constants are terms: if $n : \rightarrow s \in \Omega$, then $n \in T_{\Sigma(X),s}$.
 - Applications are terms: if $n : s_1 \times \dots \times s_k \rightarrow s \in \Omega$ and, for $1 \leq i \leq k$, $t_i \in T_{\Sigma(X),s_i}$, then $n(t_1, \dots, t_k) \in T_{\Sigma(X),s}$.

■ $\text{Var}(t) \subseteq X$:

- The set of variables occurring in term $t \in T_{\Sigma(X)}$.

■ Ground terms:

- Term t is a ground term, if $\text{Var}(t) = \emptyset$.
- The set of ground terms $T_{\Sigma} (= (T_{\Sigma,s})_{s \in S})$.



Example

- Signature $\text{NATBOOL} = (\{nat, bool\},$
 $\{True \rightarrow bool, False \rightarrow bool,$
 $\neg : bool \rightarrow bool, \wedge : bool \times bool \rightarrow bool,$
 $0 \rightarrow nat, Succ : nat \rightarrow nat,$
 $+$ $: nat \times nat \rightarrow nat,$
 $\leq : nat \times nat \rightarrow bool\})$
- Variable set X with $X_{bool} = \{b, c\}$ and $X_{nat} = \{m, n\}$.
- Terms in $T_{\text{NATBOOL}(X), bool}$:

$$\begin{aligned} & c \\ & \wedge(\wedge(True, b), False) \\ & \leq(0, +(m, Succ(n))) \end{aligned}$$

All terms are strongly typed.



Term Semantics

Take signature $\Sigma = (S, \Omega)$, set of variables X for Σ , Σ -algebra A .

- **Assignment** $\alpha : X \rightarrow A$ of X in A :
 - Family $\alpha = (\alpha_s)_{s \in S}$ of functions $\alpha_s : X_s \rightarrow A(s)$.
 - Every variable is mapped to an A -value of the corresponding sort.
- **Value** $A(\alpha)(t)$ of term t for assignment α :
 - If $t = x$ with $x \in X_s$, then $\alpha_s(x)$.
 - If $t = n$ with $\omega = n : \rightarrow s \in \Omega$, then $A(\omega)$.
 - If $t = n(t_1, \dots, t_k)$ with $\omega = n : s_1 \times \dots \times s_k \rightarrow s \in \Omega$ and, for $1 \leq i \leq k$, $t_i \in T_{\Sigma(X), s_i}$, then $A(\omega)(A(\alpha)(t_1), \dots, A(\alpha)(t_k))$.
- **Value** $A(t)$ of ground term t .
 - $A(\alpha)(t)$ for any assignment α .
 - Value of ground term does not depend on assignment.

Semantics maps terms to algebra values.



General logical framework for specifying ADTs.

- (Algebra) Logic L : for each signature Σ ,
 - a set $L(\Sigma)$ of Σ -formulas.
 - a relation $\models_{\Sigma} \subseteq \text{Alg}(\Sigma) \times L(\Sigma)$ between Σ -algebras and Σ -formulas (the **satisfaction relation** for Σ).
 - If $A \models_{\Sigma} \varphi$, we say “ φ is valid in A ” or “ A satisfies φ ”.
- L must satisfy the **isomorphism condition**:
 - If $A \simeq B$, then $(A \models_{\Sigma} \varphi \text{ iff } B \models_{\Sigma} \varphi)$.
 - For any signature Σ , Σ -formula φ , Σ -algebras A and B .
 - L cannot distinguish between isomorphic algebras.
 - L has no more information about A and B than visible in Σ .

We will investigate three specific logics.



■ Formulas $EL(\Sigma)$:

- $EL(\Sigma) = \{\forall X.t = u \mid X \text{ is a set of variables for } \Sigma, \\ t, u \in T_{\Sigma(X),s} \text{ for some sort } s \text{ of } \Sigma\}.$
- May drop “ $\forall X$ ”, if $X = \text{Var}(t) \cup \text{Var}(u)$.

■ Satisfaction Relation \models_{Σ} :

- $A \models_{\Sigma} \forall X.t = u$ iff
for all assignments $\alpha : X \rightarrow A$:
 $A(\alpha)(t) = A(\alpha)(u)$
- For each Σ -algebra A and equation $\forall X.t = u \in EL(\Sigma)$.

The logic of universally quantified equations.



Example

Take “classical” NATBOOL-algebra A (with $A(\text{nat}) = \mathbb{N}$).

$$A \models x + 1 = 1 + x$$

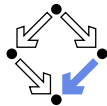
$$A \models (x \leq 0 \wedge \neg x \leq 0) = \textit{False}$$

$$A \models x = x$$

$$A \not\models x = y$$

Note: predicate \leq is operation of sort *bool*.

Conditional Equational Logic *CEL*



■ Formulas *CEL*(Σ):

- $CEL(\Sigma) = \{ \forall X. t_1 = u_1 \wedge \dots \wedge t_k = u_k \Rightarrow t_{k+1} = u_{k+1} \mid$
 X is a set of variables for Σ ,
 $t_i, u_i \in T_{\Sigma(X), s_i}$, for some sort s_i }.

- Drop “ $\forall X$ ”, if $X = \text{Var}(t_1) \cup \text{Var}(u_1) \cup \dots \cup \text{Var}(t_{k+1}) \cup \text{Var}(u_{k+1})$.

■ Satisfaction Relation \models_{Σ} :

- $A \models_{\Sigma} \forall X. t_1 = u_1 \wedge \dots \wedge t_k = u_k \Rightarrow t_{k+1} = u_{k+1}$ iff
for all assignments $\alpha : X \rightarrow A$:
if $A(\alpha)(t_1) = A(\alpha)(u_1)$ and ... and $A(\alpha)(t_k) = A(\alpha)(u_k)$ then
 $A(\alpha)(t_{k+1}) = A(\alpha)(u_{k+1})$.

The logic of universally quantified conditional equations.



Example

Take “classical” NATBOOL-algebra A (with $A(\text{nat}) = \mathbb{N}$) augmented by operation $- : \text{nat} \times \text{nat} \rightarrow \text{nat}$.

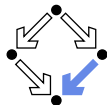
$$A \models x \leq y = \text{True} \Rightarrow (y - x) + x = y$$

$$A \models x + y = z \Rightarrow z - y = x$$

$$A \models x \leq y = \text{False} \Rightarrow y \leq x = \text{True}$$

Note: only equalities allowed as atomic predicates.

First-Order Predicate Logic PL



■ Formulas $PL(\Sigma)$:

- If $t, u \in T_{\Sigma(X),s}$ for some sort s of Σ , then $t = u \in PL(\Sigma)$.
- If $\varphi \in PL(\Sigma)$, then $\neg\varphi \in PL(\Sigma)$.
- If $\varphi_1, \varphi_2 \in PL(\Sigma)$, then $\varphi_1 \wedge \varphi_2 \in PL(\Sigma)$.
- If s is a sort of Σ , x is a variable of sort s , and $\varphi \in PL(\Sigma)$, then $(\forall x : s . \varphi) \in PL(\Sigma)$.

■ Value $A(\alpha)(\varphi)$ of formula φ for assignment $\alpha : free(\varphi) \rightarrow A$: ($free(\varphi)$... the set of free variables of φ)

- $A(\alpha)(t = u) = true$ iff $A(\alpha)(t) = A(\alpha)(u)$.
- $A(\alpha)(\neg\varphi) = true$ iff $A(\alpha)(\varphi) = false$.
- $A(\alpha)(\varphi_1 \wedge \varphi_2) = true$ iff $A(\alpha)(\varphi_1) = A(\alpha)(\varphi_2) = true$.
- $A(\alpha)(\forall x : s . \varphi) = true$ iff $A(\alpha[a/x])(\varphi) = true$ for all $a \in A(s)$.
 - $\alpha[a/x](x) = a$; $\alpha[a/x](y) = \alpha(y)$, if $x \neq y$.

■ Satisfaction Relation \models_{Σ} :

- $A \models_{\Sigma} (\varphi)$ iff $A(\alpha)(\varphi) = true$ for all assignments $\alpha : free(\varphi) \rightarrow A$.

Classical predicate logic in a typed framework.



Example

Take “classical” NATBOOL-algebra A (with $A(\text{nat}) = \mathbb{N}$).

$$A \models (\forall x : \text{nat} . (0 \leq x) = \text{True})$$

$$A \models \neg(\forall x : \text{nat} . (\forall y : \text{nat} . (x \leq y) = \text{True})).$$

$$A \models (\forall x : \text{nat} . (\forall y : \text{nat} . (x \leq y) = \text{True}) \Rightarrow x = 0)$$

The connectives \vee , \Rightarrow , \Leftrightarrow and the quantifier \exists can be introduced as abbreviations of formulas that use \neg , \wedge , \forall (e.g. $a \vee b := \Leftrightarrow \neg(\neg a \wedge \neg b)$).



- A **model** of a set of formulas $\Phi \subseteq L(\Sigma)$:
 - A Σ -algebra A is a model of Φ iff $A \models_{\Sigma} \Phi$.
 - $A \models_{\Sigma} \Phi$ iff $A \models_{\Sigma} \varphi$ for all $\varphi \in \Phi$.
- **Domain (universe)** for a signature Σ (a **Σ -domain**):
 - A class \mathcal{U} of Σ -algebras closed under isomorphism.
 - Note: a domain is an abstract datatype.
- **$Mod_{\mathcal{U},\Sigma}(\Phi) \subseteq \mathcal{U}$:**
 - The class of all algebras of domain \mathcal{U} that are models of Φ .
 - If Σ is clear, then we write $Mod_{\mathcal{U}}(\Phi)$.
 - If $\mathcal{U} = Alg(\Sigma)$, then we write $Mod_{\Sigma}(\Phi)$.
 - If both holds, then we simply write $Mod(\Phi)$.
- **Theorem:** $Mod_{\mathcal{U},\Sigma}(\Phi)$ is an abstract datatype.
 - Logic L , signature Σ , formula set $\Phi \subseteq L(\Sigma)$, Σ -domain \mathcal{U} .

A set of formulas specifies a subset of a given Σ -domain as an ADT.



Example

- $\Sigma = (\{s\}, \{0 : \rightarrow s, + : s \times s \rightarrow s\})$.
- $\Phi = \{x + (y + z) = (x + y) + z,$
 $x + 0 = x,$
 $0 + x = x,$
 $\forall x : s . \exists y : s . x + y = 0 \wedge y + x = 0\}$.
- $Mod_{\Sigma}(\Phi) = \{A \in Alg(\Sigma) \mid A(s) \text{ and } A(+)$
form a group with neutral element $A(0)\}$.

Specification of the abstract datatype “group” (polymorphic, because the group may or may not be commutative).