Formal Specification of Software

Propositional and Predicate Logic

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU

Propositional Logic: Syntax

Special symbols

$(\quad) \quad \neg \quad \wedge \quad \vee \quad \rightarrow \quad \leftrightarrow$

Signature

Propositional variables $\Sigma = \{p_0, p_1, \ldots\}$

Formulas

- **The propositional variables** $p \in \Sigma$ **are formulas**
- **If** A, B **are formulas, then**

$$\neg A \qquad (A \wedge B) \qquad (A \vee B) \qquad (A \rightarrow B) \qquad (A \leftrightarrow B)$$

are formulas

Propositional Logic: Unified Notation

Introduced by Smullyan, 1968

Conjunctive formulas **Type α**

$$\neg\neg A \qquad (A \wedge B) \qquad \neg(A \vee B) \qquad \neg(A \rightarrow B)$$

Disjunctive formulas **Type β**

$$\neg(A \wedge B) \qquad (A \vee B) \qquad (A \rightarrow B)$$

Propositional Logic: Unified Notation

Non-literal formulas and their corresponding “logical” sub-formulas

α	α_1	α_2
$A \wedge B$	A	B
$\neg(A \vee B)$	$\neg A$	$\neg B$
$\neg(A \rightarrow B)$	A	$\neg B$
$\neg\neg A$	A	A

β	β_1	β_2
$\neg(A \wedge B)$	$\neg A$	$\neg B$
$A \vee B$	A	B
$A \rightarrow B$	$\neg A$	B

Propositional Logic: Semantics

Interpretation

Function $I : \Sigma \rightarrow \{\underline{\textbf{true}}, \underline{\textbf{false}}\}$

Valuation

Extension of interpretation to formulas as follows:

$$\textbf{val}_I(p) \quad = \quad I(p)$$

$$\textbf{val}_I(\neg p) \quad = \quad \begin{cases} \underline{\textbf{true}} & \textbf{if } I(p) = \underline{\textbf{false}} \\ \underline{\textbf{false}} & \textbf{if } I(p) = \underline{\textbf{true}} \end{cases}$$

Propositional Logic: Semantics

$$\mathbf{val}_I(\alpha) = \begin{cases} \underline{\mathbf{true}} & \text{if } \mathbf{val}_I(\alpha_1) = \underline{\mathbf{true}} \\ & \text{and } \mathbf{val}_I(\alpha_2) = \underline{\mathbf{true}} \\ \underline{\mathbf{false}} & \text{if } \mathbf{val}_I(\alpha_1) = \underline{\mathbf{false}} \\ & \text{or } \mathbf{val}_I(\alpha_2) = \underline{\mathbf{false}} \end{cases}$$

$$\mathbf{val}_I(\beta) = \begin{cases} \underline{\mathbf{true}} & \text{if } \mathbf{val}_I(\beta_1) = \underline{\mathbf{true}} \\ & \text{or } \mathbf{val}_I(\beta_2) = \underline{\mathbf{true}} \\ \underline{\mathbf{false}} & \text{if } \mathbf{val}_I(\beta_1) = \underline{\mathbf{false}} \\ & \text{and } \mathbf{val}_I(\beta_2) = \underline{\mathbf{false}} \end{cases}$$

Propositional Logic: Semantics

$$\textbf{val}_I(A \leftrightarrow B) \quad = \quad \begin{cases} \underline{\textbf{true}} & \text{if } \textbf{val}_I(A) = \textbf{val}_I(B) \\ \underline{\textbf{false}} & \text{if } \textbf{val}_I(A) \neq \textbf{val}_I(B) \end{cases}$$

Predicate Logic: Syntax

Additional special symbols

"," "$\forall$" "$\exists$"

Object variables

Var $= \{x_0, x_1, \ldots\}$

Signature

Triple $\Sigma = \langle F_\Sigma, P_\Sigma, \alpha_\Sigma \rangle$ **consisting of**

- **set** F_Σ **of functions symbols**
- **set** P_Σ **of predicate symbols**
- **function** $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \rightarrow \mathbb{N}$
  assigning aritys to function and predicate symbols

Predicate Logic: Syntax

Terms

- **variables $x \in$ Var are terms**
- **if $f \in F_\Sigma$, $\alpha_\Sigma(f) = n$, and $t_1, \ldots, t_n$ terms, then $f(t_1, \ldots, t_n)$ is a term**

Atoms

If $p \in P_\Sigma$, $\alpha_\Sigma(p) = n$, and $t_1, \ldots, t_n$ terms, then $p(t_1, \ldots, t_n)$ is an atom

Predicate Logic: Syntax

Formulas

- **Atoms are formulas**
- **If A, B are formulas, $x \in$ Var, then**

$$\neg A, \quad (A \wedge B), \quad (A \vee B), \quad (A \rightarrow B), \quad (A \leftrightarrow B), \quad \forall x A, \quad \exists x A$$

are formulas

Literals

If A is an atom, then A and $\neg A$ are literals

Example

Signature

$$\Sigma_{\leq} = \langle \{0, a, b, f\}, \{in\_iv, leq\}, \alpha \rangle$$

with

$$\alpha(0) = \alpha(a) = \alpha(b) = 0$$

$$\alpha(f) = 1$$

$$\alpha(leq) = 2$$

$\alpha(in\_iv) = 3$ **(in interval)**

Formula

$$\phi = \neg \underbrace{leq(y, x)}_{\textbf{Atom}} \rightarrow \exists z \underbrace{(\neg leq(z, x) \wedge \neg leq(y, z))}_{\textbf{Scope of } \exists z}$$

Predicate Logic: Unified Notation

Extension of unified notation for propositional logic

Universal formulas **Type γ**

$$\forall x A \qquad \neg\exists x A$$

Existential formulas **Type δ**

$$\neg\forall x A \qquad \exists x A$$

Predicate Logic: Unified Notation

γ- and δ-formulas and their corresponding "logical" sub-formulas

γ	$\gamma_1(x)$
$\forall x A(x)$	$A(x)$
$\neg\exists x A(x)$	$\neg A(x)$

δ	$\delta_1(x)$
$\neg\forall x A(x)$	$\neg A(x)$
$\exists x A(x)$	$A(x)$

Predicate Logic: Semantics

Interpretation

A pair $\mathcal{D} = \langle D, I \rangle$ **where**

- D **an arbitrary non-empty set, the *universe***
- I **an interpretation function**

for $f \in F_\Sigma$**:** $I(f) : D^{\alpha(f)} \to D$

for $p \in P_\Sigma$**:** $I(p) : D^{\alpha(p)} \to \{\underline{\textbf{true}}, \underline{\textbf{false}}\}$

Variable assignment

A function $\lambda : \textbf{Var} \to D$

Predicate Logic: Semantics

Valuation

Extension of interpretation and variable assignment to formulas

val$_{\mathcal{D},\lambda}(x) = \lambda(x)$ **for** $x \in$ **Var**

val$_{\mathcal{D},\lambda}(f(t_1, \ldots, t_n)) = I(f)($**val**$_{\mathcal{D},\lambda}(t_1), \ldots,$ **val**$_{\mathcal{D},\lambda}(t_n))$

val$_{\mathcal{D},\lambda}(p(t_1, \ldots, t_n)) = I(p)($**val**$_{\mathcal{D},\lambda}(t_1), \ldots,$ **val**$_{\mathcal{D},\lambda}(t_n))$

$$\textbf{val}_{\mathcal{D},\lambda}(\forall x A) = \begin{cases} \underline{\textbf{true}} & \textbf{if val}_{\mathcal{D},\lambda_x^d}(A) = \underline{\textbf{true}} \quad \textbf{for all } d \in D \\ \underline{\textbf{false}} & \textbf{otherwise} \end{cases}$$

$$\textbf{val}_{\mathcal{D},\lambda}(\exists x A) = \begin{cases} \underline{\textbf{true}} & \textbf{if val}_{\mathcal{D},\lambda_x^d}(A) = \underline{\textbf{true}} \quad \textbf{for some } d \in D \\ \underline{\textbf{false}} & \textbf{otherwise} \end{cases}$$

val$_{\mathcal{D},\lambda}$ **defined for propositional operators in the same way as val**$_I$**.**

Predicate Logic: Semantics

Example

$$
\begin{array}{lll}
D & = & \mathbb{R} \\
I(0) & = & 0 \\
I(a) & = & -1 \\
I(b) & = & 1 \\
I(f) & = & \left\{ \begin{array}{l} \mathbb{R} \longrightarrow \mathbb{R} \\ x \mapsto x^2 \end{array} \right.
\end{array}
$$

$I(leq) =$ **true** **iff** $x \leq_{\mathbb{R}} y$

$I(in\_iv) =$ **true** **iff** $x \in [a, b]$

Predicate Logic: Semantics

Model

An interpretation $\mathcal{D}$ is model of a set Φ of formulas iff

val$_{\mathcal{D},\lambda}(A) =$ <u>true</u> for all λ and all $A \in \Phi$.

<u>Notation:</u> $\mathcal{D} \models \Phi$

Satisfiable

Φ ist satisfiable iff there are

an interpretation $\mathcal{D}$ and a variable assignment λ s.t.

val$_{\mathcal{D},\lambda}(A) =$ <u>true</u> for all $A \in \Phi$

Validity

A is valid iff

all interpretations are a model of A

Predicate Logic: Semantics

Consequence

A formula A is a consequence of Φ iff

all models of Φ are models of A as well

Notation: $\Phi \models A$

Equivalent formulas

Two formulas are equivalent iff

they are consequences of each other

Satisfiability equivalent formulas

Two formulas are satisfiability equivalent iff

they are either both satisfiable or both unsatisfiable

Substitutions

Substitution

Function $\sigma : \textbf{Var} \to \mathrm{Term}$

Written as: $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$

where $$\sigma(x) = \begin{cases} t_i & \textbf{if } x = x_i \textbf{ for } 1 \leq i \leq n \\ x & \textbf{otherwise} \end{cases}$$

Extension to terms and formulas

By replacing all *free* occurrences of variables x by $\sigma(x)$

Substitutions

Example:

$$
\begin{aligned}
\phi &= \neg leq(y, x) \rightarrow \exists z(\neg leq(z, x) \wedge \neg leq(y, z)) \\
\sigma &= \{x \leftarrow a,\ y \leftarrow w,\ z \leftarrow c\} \\
\phi\sigma &= \neg leq(w, a) \rightarrow \exists z(\neg leq(z, a) \wedge \neg leq(w, z))
\end{aligned}
$$

Note

Substitution forbidden in cases such as:

ϕ **as above** **and** $\sigma = \{y \leftarrow f(z)\}$

Typed Signatures

Definition

A *typed Signature* is a tuple

$$\Sigma = (S, \leq, F, P, \alpha),$$

where

- S **is a finite set of *types* (or *sorts*)**
- $\leq$ **is a partial ordering on** S
- F, P **are sets of function and predicate symbols (as before)**
- $\alpha : F \cup P \to S^*$ **assigns argument and domain types to function and predicate symbols**

Typed Signatures

The function α

$\alpha(f)$ = $Z_1 \ldots Z_n Z'$ **means:**

f **is a symbol for functions assigning to n-tuples of elements of type $Z_1 \ldots Z_n$ an element of type Z'**

$\alpha(p)$ = $Z_1, \ldots, Z_n$ **means:**

p **is a symbol for relations on n-tuples of elements of types $Z_1, \ldots, Z_n$**

Variables are typed as well

For each type $Z \in S$ there is an infinite set of variables of type Z

Typed Signatures: Terms

- **If x is a variable of type Z, then x is a term of type Z**
- **If**
  - $t_1, \ldots, t_n$ **are terms of types** $Y_1, \ldots, Y_n$
  - f **is a functions symbol with** $\alpha(f) = Z_1 \cdots Z_n Z'$
  - $Y_i \leq Z_i$ **for all** $1 \leq i \leq n$

  then $f(t_1, \ldots, t_n)$ **is a term of type** Z'**.**

Typed Signatures: Formulas

- **If**

  – $t_1, \ldots, t_n$ **are terms with types** $Y_1, \ldots, Y_n$
  – p **is a predicate symbol** $\alpha(p) = Z_1 \cdots Z_n$
  – $Y_i \leq Z_i$ **for all** $1 \leq i \leq n$

  then $p(t_1, \ldots, t_n)$ **is a typed (or well-sorted) formula**

- **If** t, s **are terms of sorts** X **and** Y **with** $X \leq Y$ **or** $Y \leq X$**, then** $t \doteq s$ **is a typed formula**

- **If** A, B **are typed formulas, then so are**

$$\neg A \qquad (A \wedge B) \qquad (A \vee B) \qquad (A \rightarrow B)$$

- **If** A **is a typed formula and** x **is a typed variable, then**

$$\forall x A \qquad \exists x A$$

  are typed formulas

Typed Interpretations

Given a signature $\Sigma = (S, \leq, F, P, \alpha)$

Interpretation

A pair (D, I) **such that**

- $\{D_Z \mid Z \in S\}$ **is a family of non-empty sets with**
  - $D = \bigcup\{D_Z \mid Z \in S\}$
  - $D_{Z_1} \subseteq D_{Z_2}$ **if** $Z_1 \leq Z_2$
- $I(f) : D_{Z_1} \times \cdots \times D_{Z_n} \to D_{Z'}$ **if** $\alpha(f) = Z_1 \cdots Z_n Z'$
- $I(p) \subseteq D_{Z_1} \times \cdots \times D_{Z_n}$ **if** $\alpha(p) = Z_1 \cdots Z_n$

Typed Substitutions

Typed substitution

A substitution is well-sorted if for each variable x,

the type of the term $\sigma(x)$ is a sub-type of the type of x

Special Type Structures

A type structure $(S, \leq)$ is

- ***discrete*,**

  in case $Z_1 \leq Z_2$ only if $Z_1 = Z_2$

- **a *tree structure*,**

  in case $U \leq Z_1$ and $U \leq Z_2$ implies $Z_2 \leq Z_1$ oder $Z_1 \leq Z_2$

- **a *lattice*,**

  in case that for any two sorts Z_1, Z_2 there is an infimum U, i.e.

  – $U \leq Z_1$ and $U \leq Z_2$

  – $W \leq U$ for every sort $W \in S$ with $W \leq Z_1$ and $W \leq Z_2$