Formal Specification and Verification

– First-order logic; Logical Theories (Part 2)

– Formal specification (generalities)

– Algebraic specification

22.05.2012

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Mathematical foundations

Formal logic:

- Syntax: a formal language (formula expressing facts)
- Semantics: to define the meaning of the language, that is which facts are valid)
- Deductive system: made of axioms and inference rules to formaly derive theorems, that is facts that are provable

Last time

Propositional classical logic

- Syntax
- Semantics

  Models, Validity, and Satisfiability; Entailment and Equivalence
- Checking Unsatisfiability

  Truth tables

  "Rewriting" using equivalences

  Proof systems: clausal/non-clausal

  - non-clausal: Hilbert calculus

    sequent calculus
  - clausal: Resolution; DPLL (translation to CNF needed)
  - Binary Decision Diagrams

Limitations of Propositional Logic

- Fixed, finite number of objects
  Cannot express: let G be group with arbitrary number of elements
- No functions or relations with arguments
  Can express: finite function/relation table p_{ij}
  Cannot express: properties of function/relation on all arguments, e.g., $+$ is associative
- Static interpretation
  Programs change value of their variables, e.g., via assignment, call, etc.
  Propositional formulas look at one single interpretation at a time

Beyond the Limitations of Propositional Logic

- First order logic

  (+ functions)

- Temporal logic

  (+ computations)

- Dynamic logic

  (+ computations + functions)

Last time

First-order logic

- Syntax (signature, terms, formulae, substitutions)
- Semantics (Σ-algebras (structures); assignments; value of a term; truth value of a formula)
- Models, Validity, and Satisfiability
- Entailment and Equivalence

Models, Validity, and Satisfiability

F is valid in $\mathcal{A}$ under assignment β:

$$\mathcal{A}, \beta \models F \quad :\Leftrightarrow \quad \mathcal{A}(\beta)(F) = 1$$

F is valid in $\mathcal{A}$ ($\mathcal{A}$ is a model of F):

$$\mathcal{A} \models F \quad :\Leftrightarrow \quad \mathcal{A}, \beta \models F, \text{ for all } \beta \in X \to U_{\mathcal{A}}$$

F is valid (or is a tautology):

$$\models F \quad :\Leftrightarrow \quad \mathcal{A} \models F, \text{ for all } \mathcal{A} \in \Sigma\text{-alg}$$

F is called satisfiable iff there exist $\mathcal{A}$ and β such that $\mathcal{A}, \beta \models F$.
Otherwise F is called unsatisfiable.

Entailment and Equivalence

F entails (implies) G (or G is a consequence of F), written $F \models G$

$:\Leftrightarrow$ for all $\mathcal{A} \in \Sigma$-alg and $\beta \in X \to U_{\mathcal{A}}$,
whenever $\mathcal{A}, \beta \models F$ then $\mathcal{A}, \beta \models G$.

F and G are called equivalent

$:\Leftrightarrow$ for all $\mathcal{A} \in \Sigma$-alg und $\beta \in X \to U_{\mathcal{A}}$ we have
$\mathcal{A}, \beta \models F \quad \Leftrightarrow \quad \mathcal{A}, \beta \models G$.

Entailment and Equivalence

Proposition 2.6:

F entails G iff $(F \to G)$ is valid

Proposition 2.7:

F and G are equivalent iff $(F \leftrightarrow G)$ is valid.

Extension to sets of formulas N in the "natural way", e.g., $N \models F$

$:\Leftrightarrow$ for all $\mathcal{A} \in \Sigma$-alg and $\beta \in X \to U_{\mathcal{A}}$:
if $\mathcal{A}, \beta \models G$, for all $G \in N$, then $\mathcal{A}, \beta \models F$.

Validity vs. Unsatisfiability

Validity and unsatisfiability are just two sides of the same medal as explained by the following proposition.

Proposition 2.8:

$$F \text{ valid} \quad \Leftrightarrow \quad \neg F \text{ unsatisfiable}$$

Hence in order to design a theorem prover (validity checker) it is sufficient to design a checker for unsatisfiability.

Q: In a similar way, entailment $N \models F$ can be reduced to unsatisfiability. How?

Theory of a Structure

Let $\mathcal{A} \in \Sigma$-alg. The (first-order) theory of $\mathcal{A}$ is defined as

$$Th(\mathcal{A}) = \{G \in \mathsf{F}_\Sigma(X) \mid \mathcal{A} \models G\}$$

Problem of axiomatizability:

For which structures $\mathcal{A}$ can one axiomatize $Th(\mathcal{A})$, that is, can one write down a formula F (or a recursively enumerable set F of formulas) such that

$$Th(\mathcal{A}) = \{G \mid F \models G\}?$$

Analogously for sets of structures.

Two Interesting Theories

Let $\Sigma_{Pres} = (\{0/0, s/1, +/2\}, \emptyset)$ and $\mathbb{Z}_+ = (\mathbb{Z}, 0, s, +)$ its standard interpretation on the integers.
$Th(\mathbb{Z}_+)$ is called Presburger arithmetic (M. Presburger, 1929).
(There is no essential difference when one, instead of $\mathbb{Z}$, considers the natural numbers $\mathbb{N}$ as standard interpretation.)

Presburger arithmetic is decidable in 3EXPTIME (D. Oppen, JCSS, 16(3):323–332, 1978), and in 2EXPSPACE, using automata-theoretic methods (and there is a constant $c \geq 0$ such that $Th(\mathbb{Z}_+) \notin \text{NTIME}(2^{2^{cn}})$).

Two Interesting Theories

However, $\mathbb{N}_* = (\mathbb{N}, 0, s, +, *)$, the standard interpretation of $\Sigma_{PA} = (\{0/0, s/1, +/2, */2\}, \emptyset)$, has as theory the so-called Peano arithmetic which is undecidable, not even recursively enumerable.

Note: The choice of signature can make a big difference with regard to the computational complexity of theories.

Logical theories

Syntactic view

first-order theory: given by a set $\mathcal{F}$ of (closed) first-order Σ-formulae.

the models of $\mathcal{F}$: $\quad \mathsf{Mod}(\mathcal{F}) = \{\mathcal{A} \in \Sigma\text{-alg} \mid \mathcal{A} \models G, \text{ for all } G \text{ in } \mathcal{F}\}$

Semantic view

given a class $\mathcal{M}$ of Σ-algebras

the first-order theory of $\mathcal{M}$: $\mathsf{Th}(\mathcal{M}) = \{G \in F_\Sigma(X) \text{ closed} \mid \mathcal{M} \models G\}$

Theories

$\mathcal{F}$ set of (closed) first-order formulae

$$\mathrm{Mod}(\mathcal{F}) = \{A \in \Sigma\text{-alg} \mid \mathcal{A} \models G, \text{ for all } G \text{ in } \mathcal{F}\}$$

$\mathcal{M}$ class of Σ-algebras

$$\mathrm{Th}(\mathcal{M}) = \{G \in F_\Sigma(X) \text{ closed} \mid \mathcal{M} \models G\}$$

$\mathrm{Th}(\mathrm{Mod}(\mathcal{F}))$ the set of formulae true in all models of $\mathcal{F}$

represents exactly the set of consequences of $\mathcal{F}$

Theories

$\mathcal{F}$ set of (closed) first-order formulae

$$\mathrm{Mod}(\mathcal{F}) = \{A \in \Sigma\text{-alg} \mid \mathcal{A} \models G, \text{ for all } G \text{ in } \mathcal{F}\}$$

$\mathcal{M}$ class of Σ-algebras

$$\mathrm{Th}(\mathcal{M}) = \{G \in F_\Sigma(X) \text{ closed} \mid \mathcal{M} \models G\}$$

$\mathrm{Th}(\mathrm{Mod}(\mathcal{F}))$ the set of formulae true in all models of $\mathcal{F}$
represents exactly the set of consequences of $\mathcal{F}$

Note: $\mathcal{F} \subseteq \mathrm{Th}(\mathrm{Mod}(\mathcal{F}))$ (typically strict)

$\mathcal{M} \subseteq \mathrm{Mod}(\mathrm{Th}(\mathcal{M}))$ (typically strict)

Examples

1. Groups

Let $\Sigma = (\{e/0, */2, i/1\}, \emptyset)$

Let $\mathcal{F}$ consist of all (universally quantified) group axioms:

$$\begin{array}{lll} \forall x, y, z & x * (y * z) \approx (x * y) * z & \\ \forall x & x * i(x) \approx e & \wedge \quad i(x) * x \approx e \\ \forall x & x * e \approx x & \wedge \quad e * x \approx x \end{array}$$

Every group $\mathcal{G} = (G, e_G, *_G, i_G)$ is a model of $\mathcal{F}$

$\mathsf{Mod}(\mathcal{F})$ is the class of all groups

$\mathcal{F} \subset \mathsf{Th}(\mathsf{Mod}(\mathcal{F}))$

Examples

2. Linear (positive)integer arithmetic

Let $\Sigma = (\{0/0, s/1, +/2\}, \{\leq /2\})$

Let $\mathbb{Z}_+ = (\mathbb{Z}, 0, s, +, \leq)$ the standard interpretation of integers.

$\{\mathbb{Z}_+\} \subset \mathsf{Mod}(\mathsf{Th}(\mathbb{Z}_+))$

3. Uninterpreted function symbols

Let $\Sigma = (\Omega, \Pi)$ be arbitrary

Let $\mathcal{M} = \Sigma$-alg be the class of all Σ-structures

The theory of uninterpreted function symbols is Th(Σ-alg) the family of all first-order formulae which are true in all Σ-algebras.

Examples

4. Lists

Let $\Sigma = (\{\mathsf{car}/1, \mathsf{cdr}/1, \mathsf{cons}/2\}, \emptyset)$

Let $\mathcal{F}$ be the following set of list axioms:

$$\begin{aligned} \mathsf{car}(\mathsf{cons}(x, y)) &\approx x \\ \mathsf{cdr}(\mathsf{cons}(x, y)) &\approx y \\ \mathsf{cons}(\mathsf{car}(x), \mathsf{cdr}(x)) &\approx x \end{aligned}$$

$\mathsf{Mod}(\mathcal{F})$ class of all models of $\mathcal{F}$

$\mathsf{Th}_{\mathsf{Lists}} = \mathsf{Th}(\mathsf{Mod}(\mathcal{F}))$ theory of lists (axiomatized by $\mathcal{F}$)

"Most general" models

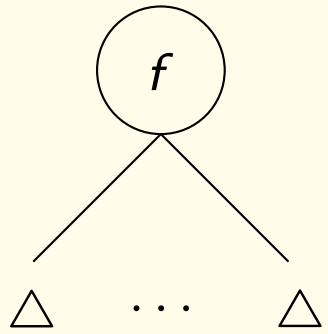
We assume that $\Pi = \emptyset$.

Term algebras

A term algebra (over Σ) is a Σ-algebra $\mathcal{A}$ such that

- $U_{\mathcal{A}} = \mathsf{T}_\Sigma$ (= the set of ground terms over Σ)
- $f_{\mathcal{A}} : (s_1, \ldots, s_n) \mapsto f(s_1, \ldots, s_n),\ \ f/n \in \Omega$

$$f_{\mathcal{A}}(\triangle, \ldots, \triangle) =$$



Term algebras

In other words, *values are fixed* to be ground terms and *functions are fixed* to be the term constructors.

Free algebras

Let $\mathcal{K}$ be the class of Σ-algebras which satisfy a set of axioms which are either equalities

$$\forall x : t(x) \approx s(x)$$

or implications:

$$\forall x : t_1(x) \approx s_1(x) \wedge \cdots \wedge t_n(x) \approx s_n(x) \rightarrow t(x) \approx s(x)$$

We can construct the "most general" model in $\mathcal{K}$:

- Step 1: Construct the term algebra $T_\Sigma(X)$ (resp. T_Σ)

Free algebras

Let $\mathcal{K}$ be the class of Σ-algebras which satisfy a set $\mathcal{F}$ of axioms which are either equalities

$$\forall x : t(x) \approx s(x)$$

or implications:

$$\forall x : t_1(x) \approx s_1(x) \wedge \cdots \wedge t_n(x) \approx s_n(x) \rightarrow t(x) \approx s(x)$$

We can construct the "most general" model in $\mathcal{K}$:

- Step 2: Identify all terms t, t' such that $\mathcal{K} \models t \approx t'$
  (all terms which become equal as a consequence of the axioms).

  $t \sim t'$ iff $\mathcal{K} \models t \approx t'$

  iff $\mathcal{F} \models t \approx t'$ because we assumed that $\mathcal{K} = \mathrm{Mod}(\mathcal{F})$

  - $\sim$ congruence relation: it clearly is an equivalence relation. Also:

  If $f/n \in \Omega$ and $t_1 \sim t_1', \ldots, t_n \sim t_n'$ then $f(t_1, \ldots, t_n) \sim f(t_1', \ldots, f_n')$.

Free algebras

Let $\mathcal{K}$ be the class of Σ-algebras which satisfy a set $\mathcal{F}$ of axioms which are either equalities

$$\forall x : t(x) \approx s(x)$$

or implications:

$$\forall x : t_1(x) \approx s_1(x) \wedge \cdots \wedge t_n(x) \approx s_n(x) \rightarrow t(x) \approx s(x)$$

We can construct the "most general" model in $\mathcal{K}$:

- Step 3: Construct the Σ-algebra of equivalence classes:

  $T_\Sigma(X)/\sim$ resp. $T_\Sigma/\sim$.

  Universe: $U = \{[t] \mid t \in T_\Sigma(X)\}$ where $[t] = \{t' \in T_\Sigma(X) \mid t \sim t'\}$
  (similar construction for T_Σ)

  Operations: $f([t_1], \ldots, [t_n]) := [f(t_1, \ldots, t_n)]$

  Since $\sim$ is a congruence, definition is independent on the choice of the representatives for the equivalence classes.

Free algebras

Let $\mathcal{K}$ be the class of Σ-algebras which satisfy a set $\mathcal{F}$ of axioms which are either equalities

$$\forall x : t(x) \approx s(x)$$

or implications:

$$\forall x : t_1(x) \approx s_1(x) \wedge \cdots \wedge t_n(x) \approx s_n(x) \rightarrow t(x) \approx s(x)$$

We can construct the "most general" model in $\mathcal{K}$:

- $T_\Sigma(X)/\sim$ is the free algebra in $\mathcal{K}$ freely generated by X.
  $T_\Sigma/\sim$ is the free algebra (with no generators) in $\mathcal{K}$.

Universal property of the free algebras

For every $\mathcal{A} \in \mathcal{K}$ and every $\beta : X \to \mathcal{A}$ there exists a unique extension β' of β which is an algebra homomorphism:

$$\beta' : T_\Sigma(X)/\sim \to \mathcal{A}$$

For every $\mathcal{A} \in \mathcal{K}$ there exists a unique algebra homomorphism:

$$\beta' : T_\Sigma \to \mathcal{A}$$

Examples

Example 1: $T_\Sigma(X)$ is the free algebra freely generated by X for the class of all algebras of type Σ.

Example 2: Let X be a set of symbols and X^* be the class of all finite strings of elements in X, including the empty string.

We construct the monoid $(X^*, \cdot, 1)$ by defining $\cdot$ to be concatenation, and 1 is the empty string.

$(X^*, \cdot, 1)$ is the free monoid freely generated by X.

Formal specification

- Specification for program/system
- Specification for properties of program/system

Verification tasks:

Check that the specification of the program/system has the required properties.

Formal specification

- **Specification languages for describing programs/processes/systems**

  Model based specification

  transition systems, abstract state machines, specifications based on set theory

  Axiom-based specification

  algebraic specification

  Declarative specifications

  logic based languages (Prolog)

  functional languages, λ-calculus (Scheme, Haskell, OCaml, ...)

  rewriting systems (very close to algebraic specification): ELAN, SPIKE, ...

- **Specification languages for properties of programs/processes/systems**

Formal specification

- **Specification languages for describing programs/processes/systems**

  Model based specification

  transition systems, abstract state machines, specifications based on set theory

  Axiom-based specification

  algebraic specification

  Declarative specifications

  logic based languages (Prolog)

  functional languages, λ-calculus (Scheme, Haskell, OCaml)

  rewriting systems (very close to algebraic specification): ELAN, SPIKE

- **Specification languages for properties of programs/processes/systems**

  Temporal logic

Algebraic specification

- appropriate for specifying the interface of a module or class
- enables verification of implementation w.r.t. specification
- for every ADT operation: argument and result types (sorts)
- semantic equations over operations (axioms) e.g. for every combination of "defined function" (e.g. top, pop) and constructor with the corresponding sort (e.g. push, empty)
- problem: consistency?, completeness?

Example: Algebraic specification

```
fmod NATSTACK is
  sorts Stack .
  protecting NAT .
  op empty : -> Stack .
  op push : Nat Stack -> Stack .
  op pop : Stack -> Stack .
  op top : Stack -> Nat .
  op length : Stack -> Nat .
  var S S2 : Stack .
  var X Y : Element .
  eq pop(push(X,S)) = S .
  eq top(push(X,S)) = X .
  eq length(empty) = 0 .
  eq length(push(X,S)) =
                1 + length(S) .
endfm
```

Example: Algebraic specification

reduce pop(push(X,S)) == S .

reduce top(pop(push(X,push(Y,S)))) == Y .

reduce S == push(X,S2) implies push(top(S),pop(S)) == S .

reduce S == push(X,S2) implies length(pop(S)) + 1 == length(S) .

- the equations can be used as term rewriting rules
- this allows proving properties of the specification

Syntax of Algebraic Specifications

Signatures: as in FOL (S, Ω, Π)

Example:

$$
\begin{aligned}
STACK = (\quad & \{Stack, Nat\}, \\
& \{\text{empty} : \epsilon \longrightarrow Stack, \\
& \ \text{push} : Nat \times Stack \longrightarrow Stack, \\
& \ \text{pop} : Stack \longrightarrow Stack, \\
& \ \text{top} : Stack \longrightarrow Nat, \\
& \ \text{length} : Stack \longrightarrow Nat, \\
& \ 0 : \epsilon \longrightarrow Nat, 1 : \epsilon \longrightarrow Nat \\
& \}
\end{aligned}
$$

Semantics of Algebraic Specifications

Σ-algebras

Observations

- different Σ-algebras are not necessarily "equivalent"
- we seek the most "abstract" Σ-algebra,
  since it anticipates as little implementation decisions as possible

Semantics of Algebraic Specifications

Σ-algebras

Observations

- different Σ-algebras are not necessarily "equivalent"
- we seek the most "abstract" Σ-algebra,
  since it anticipates as little implementation decisions as possible

No equations: Term algebras

Axioms Ax – Equations/Horn clauses: Free algebras

$T_\Sigma/\sim$, where

$t \sim t'$ iff
$Ax \models t \approx t'$ iff
For every $\mathcal{A} \in \mathrm{Mod}(Ax)$, $\mathcal{A} \models t \approx t'$

Algebraic Specification

"A gentle introduction to CASL"

M. Bidoit and P. Mosses

`http://www.lsv.ens-cachan.fr/~bidoit/GENTLE.pdf`

In the lecture I sshowed the following pages:

pages 15-26

pages 27-45

pages 66-72