



Semantics and Verification of Software

Winter Semester 2017/18

Lecture 20: Wrap-Up

Thomas Noll

Software Modeling and Verification Group

RWTH Aachen University

<http://moves.rwth-aachen.de/teaching/ws-1718/sv-sw/>

Outlook: Semantics of Functional Programming Languages

Operational Semantics of Functional Programming Languages

- Program = list of **function definitions**
- Simplest setting: **first-order** function definitions of the form

$$f(x_1, \dots, x_n) = t$$

- function name f
- formal parameters x_1, \dots, x_n
- term t over (base and defined) function calls and x_1, \dots, x_n
- **Operational semantics** (only function calls; for terms t_j , numbers z_j and variables x_k)
 - **call-by-value** case:

$$\frac{t_1 \rightarrow z_1 \quad \dots \quad t_n \rightarrow z_n \quad t[x_1 \mapsto z_1, \dots, x_n \mapsto z_n] \rightarrow z}{f(t_1, \dots, t_n) \rightarrow z}$$

- **call-by-name** case:

$$\frac{t[x_1 \mapsto t_1, \dots, x_n \mapsto t_n] \rightarrow z}{f(t_1, \dots, t_n) \rightarrow z}$$

Denotational Semantics of Functional Programming Languages

- **Denotational semantics**
 - program = **equation system** (for functions)
 - induces call-by-value and call-by-name **functional**
 - **monotonic and continuous** w.r.t. graph inclusion
 - semantics := **least fixpoint** (Tarski/Knaster Theorem)
 - **coincides** with operational semantics
- **Extensions:** higher-order types, data types, ...
- see [Winskel 1996, Sct. 9] and *Functional Programming* course [Giesl]

Outlook: Semantics of Logic Programming Languages

Syntax of Logic Programming Languages

- **Program** = list of predicate definitions
- **Predicate definition** = sequence of **clauses** of the form $q_0: \neg q_1, \dots, q_n$ with atoms q_i
- **Atom** = predicate call $p(t_1, \dots, t_k)$ with predicate p and terms t_j over variables, constants and function symbols

Example 20.1

```
father(tom, sally).  
father(tom, erica).  
father(mike, tom).  
mother(anna, sally).  
sibling(X, Y) :- parent(Z, X), parent(Z, Y).  
parent(X, Y) :- mother(X, Y).  
parent(X, Y) :- father(X, Y).
```

Outlook: Semantics of Logic Programming Languages

Operational Semantics of Logic Programming Languages

- Defined by (SLD) resolution
- Starts with single goal, called query
- Try to find refutation proof of negated query
(\Rightarrow instantiated query is logical consequence of program)
- Involves backtracking if several clause heads match

Example 20.2

```
father(tom,sally).  
father(tom,erica).  
father(mike,tom).  
mother(anna,sally).  
sibling(X,Y) :- parent(Z,X), parent(Z,Y).  
parent(X,Y) :- mother(X,Y).  
parent(X,Y) :- father(X,Y).
```

Refutation proof:

```
sibling(sally,erica).  
← parent(Z,sally), parent(Z,erica).  
← mother(Z,sally), parent(Z,erica).  
← parent(anna,erica).  
← mother(anna,erica). ⚡  
← father(anna,erica). ⚡  
← father(Z,sally), parent(Z,erica).  
← parent(tom,erica).  
← mother(tom,erica). ⚡  
← father(tom,erica). ⚡  
← □
```

Outlook: Semantics of Logic Programming Languages

Denotational Semantics of Logic Programming Languages

- meaning of program = {fully instantiated valid atoms}
- fixpoint iteration:
 - start with empty set
 - 1st step: all instantiations of facts (i.e., clauses with empty RHS)
 - $i + 1$ st step: all instantiations of facts that can be derived from known facts
- **monotonic and continuous** w.r.t. set inclusion
- semantics := **least fixpoint** (Tarski/Knaster Theorem)
- **coincides** with operational semantics

Example 20.3

```
father(tom, sally).
father(tom, erica).
father(mike, tom).
mother(anna, sally).
sibling(X, Y) :- parent(Z, X), parent(Z, Y).
parent(X, Y) :- mother(X, Y).
parent(X, Y) :- father(X, Y).
```

Fixpoint iteration:

$$A_0 = \emptyset$$

$$A_1 = \{f(t, s), f(t, e), f(m, t), m(a, s)\}$$

$$A_2 = A_1 \cup \{p(t, s), p(t, e), p(m, t), p(a, s)\}$$

$$A_3 = A_2 \cup \{s(s, e), s(e, s)\}$$

$$A_4 = A_3$$

Further Topics in Logic Programming Languages

- (Prolog) extensions: arithmetic, lists, cut, I/O, ...
- see *Logic Programming* course [Giesl]

Summary

Summary I

Semantics of programming languages

- Models the **computational meaning** of each program
- Provides abstract entities that represent just the **relevant features** of all possible executions
 - relationship between input and output
 - whether execution terminates or not
- **Ignores details** that have no relevance to the correctness of implementations
- **Equivalences** identify programs with identical semantics

(Structural) operational semantics

- Uses syntax-guided rules to specify **transition relations** (for evaluation/execution)
- Represents **scoping** of identifiers by splitting of data state into environment and store part
- Handles interaction between concurrent processes by **labels** on transitions; requires **small-step semantics**

Summary

Summary II

Denotational semantics

- Denotations are defined **inductively** on program structure
- Recursion (loops, procedures) handled as **least fixpoints** of continuous functions on CCPOs

Axiomatic semantics

- A **Hoare Logic** gives rules for the relation between assertions about values of variables before and after execution of each construct
- Employs **logical variables** for remembering (initial) values of program variables
- **Partial** vs. **total** correctness
- Recursion handled by **invariants**

Oral Exams

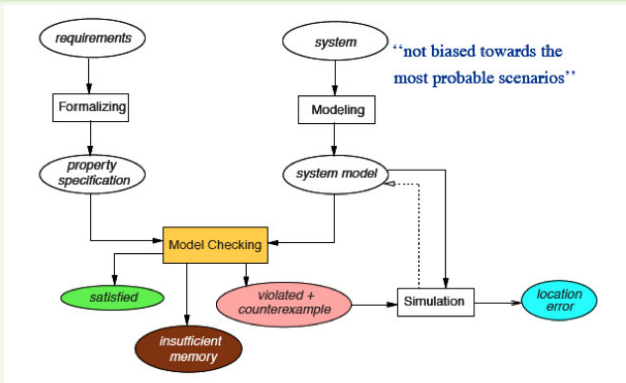
- **Oral exams:**
 - Thu Feb 8 morning
 - Tue Feb 20 morning
 - Thu Mar 8 morning
 - Thu Mar 15 morning
 - Thu Mar 29 morning

Registration via foodle poll at

<https://terminplaner2.dfn.de/foodle/Oral-Exam-in-Semantics-Verification-5a5db>

Master-Level Teaching in Summer 2018

Course *Introduction to Model Checking* [Katoen]



Course *Static Program Analysis* [Noll]

1. Dataflow analysis
2. Abstract interpretation
3. Interprocedural analysis
4. Analysis of heap data structures
5. Applications in optimising compilers and SW verification

Seminar *Formal Verification Meets Machine Learning* [Katoen, Noll, NN]

- Safety-related issues for machine learning
- Explainability in AI and in model checking
- Scalability and applicability of formal verification

Profilinie


Teilbereich: Informatik
 Name der/des Lehrenden: apl. Prof. Dr.rer.nat. Thomas Noll
 Titel der Lehrveranstaltung: Semantik und Verifikation von Software (17ws-53304)
 (Name der Umfrage)


Verwendete Werte in der Profillinie: Mittelwert


Allgemein


1.6 Die Veranstaltung interessiert mich. trifft zu  trifft nicht zu n=8 mw=1,6 md=2,0 s=0,5


Konzept der Vorlesung

2.1 Die Lernziele der Vorlesung sind definiert. trifft zu  trifft nicht zu n=7 mw=1,6 md=1,0 s=0,8


2.2 Die Vorlesung hat eine klar erkennbare Struktur. trifft zu  trifft nicht zu n=7 mw=1,3 md=1,0 s=0,5


2.3 Die zur Verfügung gestellten Materialien sind hilfreich. trifft zu  trifft nicht zu n=5 mw=1,8 md=2,0 s=0,8


2.4 Die ausgewählten Beispiele sind hilfreich. trifft zu  trifft nicht zu n=7 mw=1,6 md=1,0 s=0,8


2.5 Es werden Zusammenfassungen an sinnvollen Stellen gemacht. trifft zu  trifft nicht zu n=6 mw=1,5 md=1,5 s=0,5


Vermittlung und Verhalten


3.1 ... erklärt den Stoff verständlich. trifft zu  trifft nicht zu n=7 mw=1,6 md=2,0 s=0,5


3.2 ... geht auf Verständnisfragen ein. trifft zu  trifft nicht zu n=7 mw=1,1 md=1,0 s=0,4


3.3 ... berücksichtigt unterschiedliche Kenntnisstände der Studierenden. trifft zu  trifft nicht zu n=7 mw=2,0 md=2,0 s=1,2

3.4 ... schafft es, mich für den Vorlesungsstoff zu begeistern. trifft zu  trifft nicht zu n=8 mw=2,3 md=2,0 s=0,9

3.5 ... spricht angemessen laut und deutlich. trifft zu  trifft nicht zu n=7 mw=1,0 md=1,0 s=0,0

3.6 ... ist gut vorbereitet. trifft zu  trifft nicht zu n=7 mw=1,0 md=1,0 s=0,0

3.7 ... ist außerhalb der Vorlesung ansprechbar. (*) trifft zu  trifft nicht zu n=6 mw=1,7 md=2,0 s=0,5

3.8 ... setzt Medien ein, die zum Verständnis beitragen. trifft zu  trifft nicht zu n=6 mw=1,7 md=2,0 s=0,5

Rahmenbedingungen

4.1 Der zeitliche Rahmen der Vorlesung wird eingehalten. trifft zu  trifft nicht zu n=7 mw=1,3 md=1,0 s=0,5

(*) Hinweis: Wenn die Anzahl der Antworten auf eine Frage zu gering ist, wird für die Frage keine Auswertung angezeigt.

Auswertungsteil der offenen Fragen

Besondere Anregungen / Kritik / Wünsche:

5.1 Was hat Ihnen an der Vorlesung besonders **gut** gefallen?

Schematische Struktur
- Grundlagen werden
ausführlich erläutert
- Beweisprinzipien sehr
schon

Beweisbeispiele an der Tafel

The professor takes us time
to explain everything and
has the time to answer
questions

5.2 Was hat Ihnen an der Vorlesung **nicht** gefallen?

Symboltabelle / Symbole
manchmal inkonsistent
wsp. sp - Definitionen vor
der Benutzung / zum
Verständnis sinnvoller
Bei zu vielen Beweisen unklar,
ob diese relevant ^{in VL} sind.
umte mit end updates?!
chain & Charakterisierung
fix -
w.r.t. chain anfügen
Bilder ?

†
Maybe speed up tedious proofs using
a proof assistant like Coq or Isabelle

Sometimes lecture notes
instead of slides to understand
the lectures better would
be nice.