

Semantics and Verification of Software

Lecture 23: Wrap-Up

Thomas Noll

Lehrstuhl für Informatik 2
RWTH Aachen University
noll@cs.rwth-aachen.de

<http://www-i2.informatik.rwth-aachen.de/i2/svsw/>

Summer semester 2007

- 1 Repetition: The Interprocedural Fixpoint Solution
- 2 Further Topics in Dataflow Analysis
- 3 Further Topics in Formal Semantics
- 4 Evaluation of the Course

- **Goal:** adapt fixpoint solution to **avoid invalid paths**
- **Approach:** encode call history into data flow properties
(use **stacks** D^+ as dataflow version of runtime stack)
- Non-procedural constructs (**skip**, assignments, tests):
operate only on topmost element
- **call:** computes new topmost entry from current and pushes it
- **return:** removes topmost entry and combines it with underlying entry

The Interprocedural Extension I

Definition (Interprocedural extension (forward analysis))

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. The **interprocedural extension** of S is given by

$$\hat{S} := (Lab, E, F, (\hat{D}, \hat{\sqsubseteq}), \hat{\iota}, \hat{\varphi})$$

where

- $\hat{D} := D^+$
- $d_1 \dots d_n \hat{\sqsubseteq} d'_1 \dots d'_n$ iff $d_i \sqsubseteq d'_i$ for every $1 \leq i \leq n$
- $\hat{\iota} := \iota \in D^+$
- for each $l \in Lab \setminus \{l_c, l_n, l_x, l_r \mid (l_c, l_n, l_x, l_r) \in IF\}$, $\hat{\varphi}_l : D^+ \rightarrow D^+$ is given by $\hat{\varphi}_l(dw) := \varphi_l(d)w$
- for each $(l_c, l_n, l_x, l_r) \in IF$, $\hat{\varphi}_l : D^+ \rightarrow D^+$ is given by
 - $\hat{\varphi}_{l_c}(dw) := \varphi_{l_c}(d)dw$
 - $\hat{\varphi}_{l_n}(w) := w$
 - $\hat{\varphi}_{l_x}(w) := w$
 - $\hat{\varphi}_{l_r}(d'w) := d''w$ where $d'' := \varphi_{l_r}^1(d) \sqcup \varphi_{l_r}^2(d')$

Example (Constant Propagation (cf. Lecture 19))

$\hat{S} := (Lab, E, F, (\hat{D}, \hat{\sqsubseteq}), \hat{\iota}, \hat{\varphi})$ is determined by

- $D := \{\delta \mid \delta : Var_c \rightarrow \mathbb{Z} \cup \{\perp, \top\}\}$
- $\perp \sqsubseteq z \sqsubseteq \top$
- $\iota := \delta_\top \in D$
- for each $l \in Lab \setminus \{l_c, l_n, l_x, l_r \mid (l_c, l_n, l_x, l_r) \in IF\}$,
$$\varphi_l(\delta) := \begin{cases} \delta & \text{if } B^l = \text{skip or } B^l \in BExp \\ \delta[x \mapsto \mathfrak{A}[[a]]\delta] & \text{if } B^l = (x := a) \end{cases}$$
- whenever pc contains $[\text{call } P(a, z)]_{l_r}^{l_c}$ and
 $\text{proc } [P(\text{val } x, \text{res } y)]_{l_n}^{l_x} \text{ is } c \text{ [end]}_{l_x}$,
 - **call**: set input parameter and reset output parameter
 $\varphi_{l_c}(\delta) := \delta[x \mapsto \mathfrak{A}[[a]]\delta, y \mapsto \top]$
 - **return**: propagate output parameter to caller by resetting old z value **and copying y to z**
 $\varphi_{l_r}^1(\delta) := \delta[z \mapsto \perp] \quad \varphi_{l_r}^2(\delta') := \delta'[x \mapsto \perp, y \mapsto \perp, z \mapsto \delta'(y)]$

The Equation System

For an interprocedural dataflow system $\hat{S} := (Lab, E, F, (\hat{D}, \hat{\sqsubseteq}), \hat{t}, \hat{\varphi})$, the intraprocedural equation system

$$AI_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup \{\varphi_{l'}(AI_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

is extended to a system with three kinds of equations (for every $l \in Lab$):

- for actual **dataflow information**: $AI_l \in D$
(extension of intraprocedural AI)
- for **single nodes**: $f_l : D^+ \rightarrow D^+$
(extension of intraprocedural transfer functions)
- for flow graphs of **complete procedures**: $F_l : D^+ \rightarrow D^+$
($F_l(w)$ yields information at l if surrounding procedure is called with information w)

- 1 Repetition: The Interprocedural Fixpoint Solution
- 2 Further Topics in Dataflow Analysis
- 3 Further Topics in Formal Semantics
- 4 Evaluation of the Course

Context–Sensitive Dataflow Analysis

- **Observation:** MVP and fixpoint solution maintain **proper relationship between procedure calls and returns**
- **But:** do not distinguish between different procedure calls

$$AI_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup \{ \varphi_{l_c}(AI_{l_c}) \mid (l_c, l_n, l_x, l_r) \in IF \} & \text{if } l = l_n \\ \bigsqcup \{ f_{l'}(AI_{l'}) \mid (l', l) \in F \} & \text{for some } (l_c, l_n, l_x, l_r) \in IF \\ & \text{otherwise} \end{cases}$$

- information about calling states combined for all call sites
- procedure body only analyzed once using combined information
- resulting information used at all return points

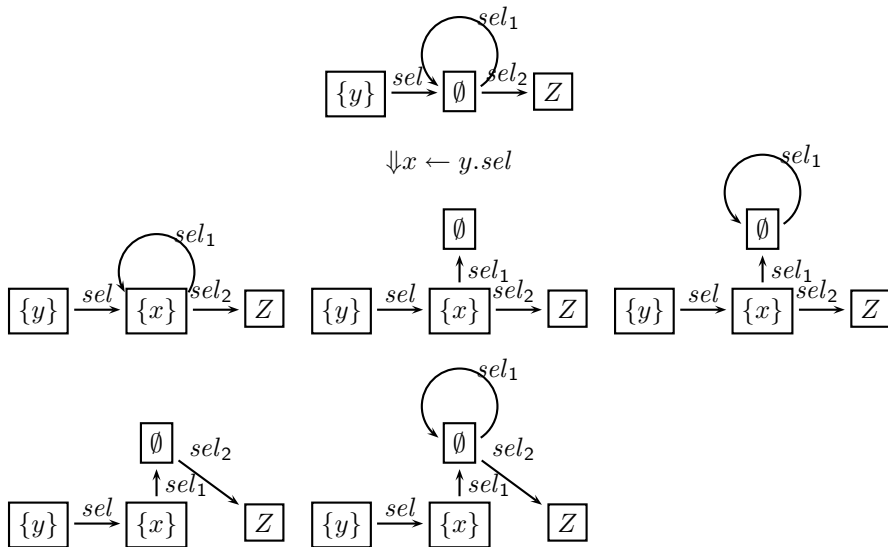
$\implies$ **“context–insensitive”**

- **Alternative:** **context–sensitive** analysis
 - separate information for different call sites
 - implementation by “procedure cloning”
 - more precise
 - more costly

Shape Analysis I

- **So far:** only **static data structures** (variables)
- **Now:** **pointer (variables)** and **dynamic memory allocation** using heaps
- **Goal:** **shape analysis** = approximative analysis of heap data structures
- Interesting information:
 - data types (to avoid type errors, such as dereferencing `nil`)
 - sharing (different pointer variables referencing same address; aliasing)
 - reachability of nodes (garbage collection)
 - disjointness of heap regions (parallelizability)
 - shapes (lists, trees, absence of cycles, ...)
- Representation of (infinitely many) concrete heap states by (finitely many) abstract **shape graphs**
 - abstract nodes A = sets of variables (interpretation: $x \in A$ iff x points to concrete node represented by A)
 - $\emptyset$ represents all concrete nodes that are not directly reachable
 - transfer functions transform (sets of) shape graphs
- see [Nielson/Nielson/Hankin 2005, Sct. 2.6]

Shape Analysis II



- So far: **semantics** and **dataflow analysis** of programs independent
- Of course both are (and should be) related!
- To this aim: introduce **small-step operational semantics** operating on program labels

$$\langle l_0, \sigma_0 \rangle \rightarrow \dots \langle l_n, \sigma_n \rangle \rightarrow \sigma_{n+1}$$

where $l_i \in Lab$ and $\sigma_i : Var \rightarrow \mathbb{Z}$

- **Example:** **correctness of Constant Propagation**

Let $c_0 \in Cmd$ with $l_0 = \text{init}(c_0)$, and let $l \in Lab_{c_0}$, $x \in Var$, and $z \in \mathbb{Z}$ such that $CP_l(x) = z$. Then for every $\sigma_0, \sigma \in \Sigma$ such that $\langle l_0, \sigma_0 \rangle \rightarrow^* \langle l, \sigma \rangle$, $\sigma(x) = z$.

- see [Nielson/Nielson/Hankin 2005, Sct. 2.2]

- 1 Repetition: The Interprocedural Fixpoint Solution
- 2 Further Topics in Dataflow Analysis
- 3 Further Topics in Formal Semantics**
- 4 Evaluation of the Course

Semantics of Functional Languages I

- 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)

- **call-by-value** case:

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

- **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**

- program 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 *Functional Programming* course [Giesl] and [Winskel 1996, Sct. 9]

Semantics of Concurrent Languages

- **Problem:** “classical” view of sequential systems

Program : Input $\rightarrow$ Output

not adequate for concurrent settings

- Missing: aspect of **interaction**
- Typical approach:
 - concurrency modelled by interleaving
 - interaction modelled by (explicit) communication
- Example: Milner’s *Calculus of Communicating Systems* (CCS)
- Syntax: $P ::= 0 \mid \alpha.P \mid P_1 + P_2 \mid P_1 \parallel P_2 \mid \dots$
- (Operational) Semantics: **labelled transition systems** defined by transition rules of the form

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} \quad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \quad \frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \quad \frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\bar{\alpha}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \dots$$

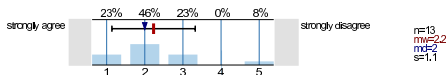
- see course on *Modelling Concurrent and Probabilistic Systems* in WS 2007/08 [Katoen, Noll] and [Winskel 1996, Sct. 14]

- 1 Repetition: The Interprocedural Fixpoint Solution
- 2 Further Topics in Dataflow Analysis
- 3 Further Topics in Formal Semantics
- 4 Evaluation of the Course

Overall Evaluation of Lecture I

Overall evaluation (SS 2007)

1. I learned a lot in this course.



09.07.2007

EvaSys Auswertung

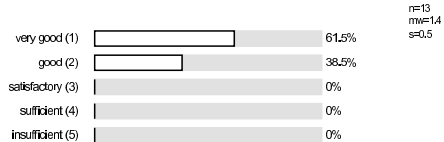
Seite 5

(see <http://www-i2.informatik.rwth-aachen.de/i2/svsw/> for full reports)

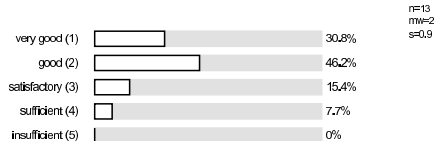
Overall Evaluation of Lecture II

Priv.-Doz. Dr.rer.nat. Thomas Noll, Semantik und Verifikation von Software (1028771)

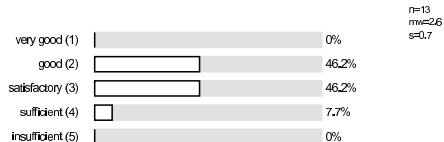
2. I would grade the instructor with:



3. I would grade the course with:



4. I would grade my own effort with:

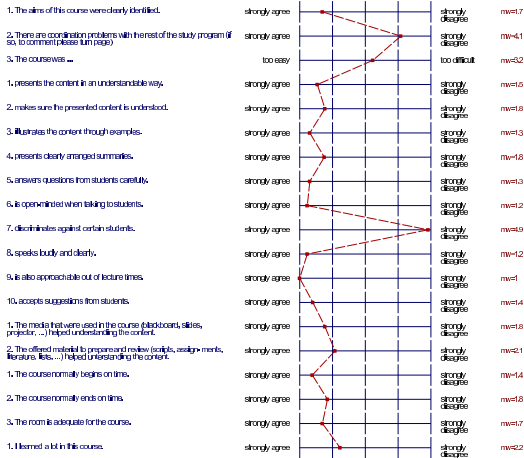


Evaluation Profile of Lecture

Priv.-Doz. Dr. rer. nat. Thomas Noll, Semantik und Verifikation von Software (1028771)

Profillinie

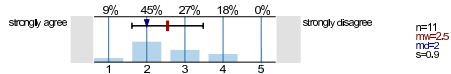
Teilbereich: Informatik
Name des/der Lehrenden: Priv.-Doz. Dr. rer. nat. Thomas Noll
Titel der Lehrveranstaltung: Semantik und Verifikation von Software (1028771)
(Name der Umfrage)



Overall Evaluation of Exercise Class I

Overall evaluation (SS 2007)

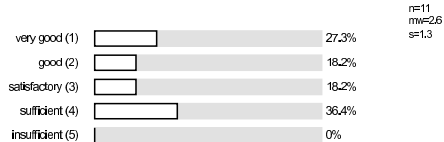
1. I learned a lot in this course.



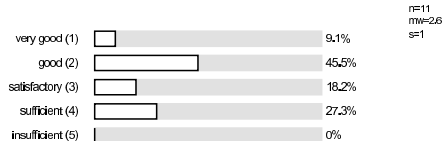
Overall Evaluation of Exercise Class II

Priv.-Doz. Dr.rer.nat. Thomas Noll, Semantik und Verifikation von Software

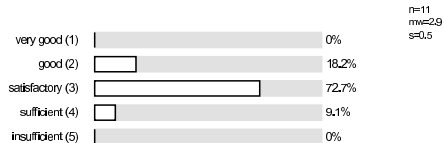
2. I would grade the instructor with:



3. I would grade the course with:



4. I would grade my own effort with:



Evaluation Profile of Exercise Class

Priv.-Doz. Dr. rer. nat. Thomas Noll, Semantik und Verifikation von Software

Profillinie

Teilbereich: Informatik
Name des/der Lehrenden: Priv.-Doz. Dr. rer. nat. Thomas Noll
Titel der Lehrveranstaltung: Semantik und Verifikation von Software
(Name der Umfrage)

1. Lecture and lab are well coordinated with each other.

strongly agree strongly disagree mw=2.3

2. The amount of assignments is appropriate.

strongly agree strongly disagree mw=1.7

3. The assignments are not too difficult.

strongly agree strongly disagree mw=2.3

4. The assignments are application related.

strongly agree strongly disagree mw=2.1

5. The assignments are corrected carefully.

strongly agree strongly disagree mw=3.2

1. presents the content in an understandable way.

strongly agree strongly disagree mw=2.5

2. makes sure the presented content is understood.

strongly agree strongly disagree mw=2.2

3. illustrates the content through examples.

strongly agree strongly disagree mw=2.3

4. presents clearly arranged summaries.

strongly agree strongly disagree mw=3

5. answers questions from students carefully.

strongly agree strongly disagree mw=2.3

6. is open minded when talking to students.

strongly agree strongly disagree mw=1.6

7. discriminates against certain students.

strongly agree strongly disagree mw=4.9

8. speaks loudly and clearly.

strongly agree strongly disagree mw=2.1

9. is also approachable out of lecture times.

strongly agree strongly disagree mw=1

10. accepts suggestions from students.

strongly agree strongly disagree mw=1.4

1. The course normally begins on time.

strongly agree strongly disagree mw=1.4

2. The course normally ends on time.

strongly agree strongly disagree mw=1.7

3. The course has an acceptable group size.

strongly agree strongly disagree mw=1.2

4. The room is adequate for the course.

strongly agree strongly disagree mw=1.6

1. I learned a lot in this course.

strongly agree strongly disagree mw=2.5

Lecture:

- ① Slides more “self-contained”
- ② Repetition of proof principles
 - good for beginners
 - boring for experienced listeners
- ③ Double covering of denotational semantics by *Functional Programming* course

Points 2 and 3 perhaps call for a separate, introductory course on “Universal Algebra”

Exercise class:

- ① Questions and example solutions more “reliable”
- ② Comments and suggestions in correction of exercises
- ③ More explanations to solutions?