

Semantics and Verification of Software

Lecture 23: Dataflow Analysis VI (MOP vs. Fixpoint Solution & Wrap-Up)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

`noll@cs.rwth-aachen.de`

`http://www-i2.informatik.rwth-aachen.de/i2/svsw10/`

Summer Semester 2010

- 1 Repetition: The MOP Solution
- 2 Repetition: Constant Propagation
- 3 MOP vs. Fixpoint Solution
- 4 Further Topics in Formal Semantics
- 5 Upcoming Courses
- 6 Evaluation of the Course

The MOP Solution I

- Other **solution method** for dataflow systems
- MOP = **Meet Over all Paths**
- Analysis information for block B^l = **least upper bound over all paths leading to l**

Definition (Paths)

Let $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. For every $l \in L$, the set of **paths up to l** is given by

$$Path(l) := \{[l_1, \dots, l_{k-1}] \mid k \geq 1, l_1 \in E, \\ (l_i, l_{i+1}) \in F \text{ for every } 1 \leq i \leq k, l_k = l\}.$$

For a path $p = [l_1, \dots, l_{k-1}] \in Path(l)$, we define the **transfer function** $\varphi_p : D \rightarrow D$ by

$$\varphi_p := \varphi_{l_{k-1}} \circ \dots \circ \varphi_{l_1} \circ \text{id}_D$$

(so that $\varphi_{[]} = \text{id}_D$).

Definition (MOP solution)

Let $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system where $L = \{l_1, \dots, l_n\}$. The **MOP solution** for S is determined by

$$\text{mop}(S) := (\text{mop}(l_1), \dots, \text{mop}(l_n)) \in D^n$$

where, for every $l \in L$,

$$\text{mop}(l) := \bigsqcup \{\varphi_p(\iota) \mid p \in \text{Path}(l)\}.$$

Remark:

- $\text{Path}(l)$ is generally infinite

⇒ not clear how to compute $\text{mop}(l)$

- In fact: MOP solution generally undecidable

Undecidability of the MOP Solution

Theorem (Undecidability of MOP solution)

The MOP solution for Constant Propagation is undecidable.

Proof.

Based on undecidability of **Modified Post Correspondence Problem**:

Let Γ be some alphabet, $n \in \mathbb{N}$, and $u_1, \dots, u_n, v_1, \dots, v_n \in \Gamma^+$.

Does there exist $i_1, \dots, i_m \in \{1, \dots, n\}$ with $m \geq 1$ and $i_1 = 1$ such that $u_{i_1} u_{i_2} \dots u_{i_m} = v_{i_1} v_{i_2} \dots v_{i_m}$?

(on the board)



- 1 Repetition: The MOP Solution
- 2 Repetition: Constant Propagation
- 3 MOP vs. Fixpoint Solution
- 4 Further Topics in Formal Semantics
- 5 Upcoming Courses
- 6 Evaluation of the Course

Formalizing Constant Propagation Analysis I

The **dataflow system** $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ is given by

- set of labels $L := L_c$,
- extremal labels $E := \{\text{init}(c)\}$ (forward problem),
- flow relation $F := \text{flow}(c)$ (forward problem),
- complete lattice $(D, \sqsubseteq)$ where
 - $D := \{\delta \mid \delta : \text{Var}_c \rightarrow \mathbb{Z} \cup \{\perp, \top\}\}$
 - $\delta(x) = z \in \mathbb{Z}$: x has **constant value** z
 - $\delta(x) = \perp$: x **undefined**
 - $\delta(x) = \top$: x **overdefined** (i.e., different possible values)
 - $\sqsubseteq \subseteq D \times D$ defined by pointwise extension of $\perp \sqsubseteq z \sqsubseteq \top$ (for every $z \in \mathbb{Z}$)

Example

$$\begin{aligned}\text{Var}_c &= \{w, x, y, z\}, \\ \delta_1 &= (\underbrace{\perp}_w, \underbrace{1}_x, \underbrace{2}_y, \underbrace{\top}_z), \quad \delta_2 = (\underbrace{3}_w, \underbrace{1}_x, \underbrace{4}_y, \underbrace{\top}_z) \\ \implies \delta_1 \sqcup \delta_2 &= (\underbrace{3}_w, \underbrace{1}_x, \underbrace{\top}_y, \underbrace{\top}_z)\end{aligned}$$

Dataflow system $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ (continued):

- extremal value $\iota := \delta_{\top} \in D$ where $\delta_{\top}(x) := \top$ for every $x \in \text{Var}_c$,
- transfer functions $\{\varphi_l \mid l \in L\}$ defined by

$$\varphi_l(\delta) := \begin{cases} \delta & \text{if } B^l = \text{skip or } B^l \in BExp \\ \delta[x \mapsto \mathfrak{A}[[a]]\delta] & \text{if } B^l = (x := a) \end{cases}$$

where

$$\begin{aligned} \mathfrak{A}[[x]]\delta &:= \delta(x) \\ \mathfrak{A}[[z]]\delta &:= z \end{aligned} \quad \mathfrak{A}[[a_1 \text{ op } a_2]]\delta := \begin{cases} z_1 \text{ op } z_2 & \text{if } z_1, z_2 \in \mathbb{Z} \\ \perp & \text{if } z_1 = \perp \text{ or } z_2 = \perp \\ \top & \text{otherwise} \end{cases}$$

for $z_1 := \mathfrak{A}[[a_1]]\delta$ and $z_2 := \mathfrak{A}[[a_2]]\delta$

- 1 Repetition: The MOP Solution
- 2 Repetition: Constant Propagation
- 3 MOP vs. Fixpoint Solution
- 4 Further Topics in Formal Semantics
- 5 Upcoming Courses
- 6 Evaluation of the Course

Theorem 23.1 (MOP vs. Fixpoint Solution)

Let $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. Then

$$\text{mop}(S) \sqsubseteq \text{fix}(\Phi_S)$$

Reminder: by Definition 21.2,

$$\Phi_S : D^n \rightarrow D^n : (d_{l_1}, \dots, d_{l_n}) \mapsto (d'_{l_1}, \dots, d'_{l_n})$$

where $L = \{l_1, \dots, l_n\}$ and, for each $1 \leq i \leq n$,

$$d'_{l_i} := \begin{cases} \iota & \text{if } l_i \in E \\ \bigsqcup \{ \varphi_{l'}(d_{l'}) \mid (l', l_i) \in F \} & \text{otherwise} \end{cases}$$

MOP vs. Fixpoint Solution I

Theorem 23.1 (MOP vs. Fixpoint Solution)

Let $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. Then

$$\text{mop}(S) \sqsubseteq \text{fix}(\Phi_S)$$

Reminder: by Definition 21.2,

$$\Phi_S : D^n \rightarrow D^n : (d_{l_1}, \dots, d_{l_n}) \mapsto (d'_{l_1}, \dots, d'_{l_n})$$

where $L = \{l_1, \dots, l_n\}$ and, for each $1 \leq i \leq n$,

$$d'_{l_i} := \begin{cases} \iota & \text{if } l_i \in E \\ \bigsqcup \{ \varphi_{l'}(d_{l'}) \mid (l', l_i) \in F \} & \text{otherwise} \end{cases}$$

Proof.

on the board



MOP vs. Fixpoint Solution I

Theorem 23.1 (MOP vs. Fixpoint Solution)

Let $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. Then

$$\text{mop}(S) \sqsubseteq \text{fix}(\Phi_S)$$

Reminder: by Definition 21.2,

$$\Phi_S : D^n \rightarrow D^n : (d_{l_1}, \dots, d_{l_n}) \mapsto (d'_{l_1}, \dots, d'_{l_n})$$

where $L = \{l_1, \dots, l_n\}$ and, for each $1 \leq i \leq n$,

$$d'_{l_i} := \begin{cases} \iota & \text{if } l_i \in E \\ \bigsqcup \{ \varphi_{l'}(d_{l'}) \mid (l', l_i) \in F \} & \text{otherwise} \end{cases}$$

Proof.

on the board □

The next example shows that both solutions can indeed be different.

Example 23.2 (Constant Propagation)

```
c := if [z > 0]1 then
    [x := 2;]2
    [y := 3;]3
else
    [x := 3;]4
    [y := 2;]5
    [z := x+y;]6
    [...]7
```

Example 23.2 (Constant Propagation)

```
c := if [z > 0]1 then
    [x := 2;]2
    [y := 3;]3
else
    [x := 3;]4
    [y := 2;]5
    [z := x+y;]6
    [...]7
```

Transfer functions

(for $\delta = (\delta(x), \delta(y), \delta(z)) \in D$):

$$\varphi_1((a, b, c)) = (a, b, c)$$

$$\varphi_2((a, b, c)) = (2, b, c)$$

$$\varphi_3((a, b, c)) = (a, 3, c)$$

$$\varphi_4((a, b, c)) = (3, b, c)$$

$$\varphi_5((a, b, c)) = (a, 2, c)$$

$$\varphi_6((a, b, c)) = (a, b, a + b)$$

Example 23.2 (Constant Propagation)

```
c := if [z > 0]1 then
    [x := 2;]2
    [y := 3;]3
else
    [x := 3;]4
    [y := 2;]5
    [z := x+y;]6
    [...]7
```

Transfer functions

(for $\delta = (\delta(x), \delta(y), \delta(z)) \in D$):

$$\varphi_1((a, b, c)) = (a, b, c)$$

$$\varphi_2((a, b, c)) = (2, b, c)$$

$$\varphi_3((a, b, c)) = (a, 3, c)$$

$$\varphi_4((a, b, c)) = (3, b, c)$$

$$\varphi_5((a, b, c)) = (a, 2, c)$$

$$\varphi_6((a, b, c)) = (a, b, a + b)$$

① Fixpoint solution:

$$CP_1 = \iota = (\top, \top, \top)$$

$$CP_2 = \varphi_1(CP_1) = (\top, \top, \top)$$

$$CP_3 = \varphi_2(CP_2) = (2, \top, \top)$$

$$CP_4 = \varphi_1(CP_1) = (\top, \top, \top)$$

$$CP_5 = \varphi_4(CP_4) = (3, \top, \top)$$

$$CP_6 = \varphi_3(CP_3) \sqcup \varphi_5(CP_5) \\ = (2, 3, \top) \sqcup (3, 2, \top) = (\top, \top, \top)$$

$$CP_7 = \varphi_6(CP_6) = (\top, \top, \top)$$

Example 23.2 (Constant Propagation)

```
c := if [z > 0]1 then
    [x := 2;]2
    [y := 3;]3
else
    [x := 3;]4
    [y := 2;]5
    [z := x+y;]6
    [...]7
```

Transfer functions

(for $\delta = (\delta(x), \delta(y), \delta(z)) \in D$):

$$\varphi_1((a, b, c)) = (a, b, c)$$

$$\varphi_2((a, b, c)) = (2, b, c)$$

$$\varphi_3((a, b, c)) = (a, 3, c)$$

$$\varphi_4((a, b, c)) = (3, b, c)$$

$$\varphi_5((a, b, c)) = (a, 2, c)$$

$$\varphi_6((a, b, c)) = (a, b, a + b)$$

① Fixpoint solution:

$$CP_1 = \iota = (\top, \top, \top)$$

$$CP_2 = \varphi_1(CP_1) = (\top, \top, \top)$$

$$CP_3 = \varphi_2(CP_2) = (2, \top, \top)$$

$$CP_4 = \varphi_1(CP_1) = (\top, \top, \top)$$

$$CP_5 = \varphi_4(CP_4) = (3, \top, \top)$$

$$CP_6 = \varphi_3(CP_3) \sqcup \varphi_5(CP_5) \\ = (2, 3, \top) \sqcup (3, 2, \top) = (\top, \top, \top)$$

$$CP_7 = \varphi_6(CP_6) = (\top, \top, \top)$$

② MOP solution:

$$\text{mop}(7) = \varphi_{[1,2,3,6]}(\top, \top, \top) \sqcup \\ \varphi_{[1,4,5,6]}(\top, \top, \top) \\ = (2, 3, 5) \sqcup (3, 2, 5) \\ = (\top, \top, 5)$$

A sufficient criterion for the coincidence of MOP and Fixpoint Solution is the distributivity of the transfer functions.

Definition 23.3 (Distributivity)

- Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be complete lattices, and let $F : D \rightarrow D'$. F is called **distributive (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$)** if, for every $d_1, d_2 \in D$,

$$F(d_1 \sqcup_D d_2) = F(d_1) \sqcup_{D'} F(d_2).$$

A sufficient criterion for the coincidence of MOP and Fixpoint Solution is the distributivity of the transfer functions.

Definition 23.3 (Distributivity)

- Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be complete lattices, and let $F : D \rightarrow D'$. F is called **distributive (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$)** if, for every $d_1, d_2 \in D$,

$$F(d_1 \sqcup_D d_2) = F(d_1) \sqcup_{D'} F(d_2).$$

- A dataflow system $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ is called **distributive** if every $\varphi_l : D \rightarrow D$ ($l \in L$) is so.

Example 23.4

- ① The Available Expressions dataflow system is distributive:

$$\begin{aligned}\varphi_l(A_1 \sqcup A_2) &= ((A_1 \cap A_2) \setminus \text{kill}_{\text{AE}}(B^l)) \cup \text{gen}_{\text{AE}}(B^l) \\ &= ((A_1 \setminus \text{kill}_{\text{AE}}(B^l)) \cup \text{gen}_{\text{AE}}(B^l)) \cap \\ &\quad ((A_2 \setminus \text{kill}_{\text{AE}}(B^l)) \cup \text{gen}_{\text{AE}}(B^l)) \\ &= \varphi_l(A_1) \sqcup \varphi_l(A_2)\end{aligned}$$

Example 23.4

- ① The Available Expressions dataflow system is distributive:

$$\begin{aligned}\varphi_l(A_1 \sqcup A_2) &= ((A_1 \cap A_2) \setminus \text{kill}_{\text{AE}}(B^l)) \cup \text{gen}_{\text{AE}}(B^l) \\ &= ((A_1 \setminus \text{kill}_{\text{AE}}(B^l)) \cup \text{gen}_{\text{AE}}(B^l)) \cap \\ &\quad ((A_2 \setminus \text{kill}_{\text{AE}}(B^l)) \cup \text{gen}_{\text{AE}}(B^l)) \\ &= \varphi_l(A_1) \sqcup \varphi_l(A_2)\end{aligned}$$

- ② The Live Variables dataflow system is distributive (similar)

Example 23.4

- ❶ The Available Expressions dataflow system is distributive:

$$\begin{aligned}\varphi_l(A_1 \sqcup A_2) &= ((A_1 \cap A_2) \setminus \text{kill}_{\text{AE}}(B^l)) \cup \text{gen}_{\text{AE}}(B^l) \\ &= ((A_1 \setminus \text{kill}_{\text{AE}}(B^l)) \cup \text{gen}_{\text{AE}}(B^l)) \cap \\ &\quad ((A_2 \setminus \text{kill}_{\text{AE}}(B^l)) \cup \text{gen}_{\text{AE}}(B^l)) \\ &= \varphi_l(A_1) \sqcup \varphi_l(A_2)\end{aligned}$$

- ❷ The Live Variables dataflow system is distributive (similar)
- ❸ The Constant Propagation dataflow system is not distributive:

$$\begin{aligned}(\top, \top, \top) &= \varphi_{z:=x+y}((2, 3, \top) \sqcup (3, 2, \top)) \\ &\neq \varphi_{z:=x+y}((2, 3, \top)) \sqcup \varphi_{z:=x+y}((3, 2, \top)) \\ &= (\top, \top, 5)\end{aligned}$$

Theorem 23.5 (MOP vs. Fixpoint Solution)

Let $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a distributive dataflow system. Then

$$\text{mop}(S) = \text{fix}(\Phi_S)$$

Theorem 23.5 (MOP vs. Fixpoint Solution)

Let $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a distributive dataflow system. Then

$$\text{mop}(S) = \text{fix}(\Phi_S)$$

Proof.

- by showing that $\Phi_S(\text{mop}(S)) = \text{mop}(S)$...
(see [Nielson/Nielson/Hankin 2005, p. 81])
- ... and using $\text{mop}(S) \sqsubseteq \text{fix}(\Phi_S)$ (Theorem 23.1)



- 1 Repetition: The MOP Solution
- 2 Repetition: Constant Propagation
- 3 MOP vs. Fixpoint Solution
- 4 Further Topics in Formal Semantics
- 5 Upcoming Courses
- 6 Evaluation of the Course

Semantics of Functional Languages I

- Program = list of **function definitions**

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 $x_1, \dots, x_n$

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:

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

- **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

- 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

- 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, ...

- 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]

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

Program : Input $\rightarrow$ Output

not adequate for concurrent settings

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

Program : Input $\rightarrow$ Output

not adequate for concurrent settings

- Missing: aspect of **interaction**

- **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**

- **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)*

- **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: $P ::= 0 \mid \alpha.P \mid P_1 + P_2 \mid P_1 \parallel P_2 \mid \dots$

- **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

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

- **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

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

- see course on **Modelling Concurrent and Probabilistic Systems** in Summer 2009 [Katoen, Noll] and [Winskel 1996, Sct. 14]

- 1 Repetition: The MOP Solution
- 2 Repetition: Constant Propagation
- 3 MOP vs. Fixpoint Solution
- 4 Further Topics in Formal Semantics
- 5 Upcoming Courses
- 6 Evaluation of the Course

- Course **Advanced Model Checking** [Katoen]
- Practical Course **Model Checking** [Katoen, Sher, Yue]
- Course **Compiler Construction** [Noll] (“Hiwi” jobs available!)

- 1 Repetition: The MOP Solution
- 2 Repetition: Constant Propagation
- 3 MOP vs. Fixpoint Solution
- 4 Further Topics in Formal Semantics
- 5 Upcoming Courses
- 6 Evaluation of the Course

Profilinie

Teilbereich:	Informatik
Name der/des	Priv.-Doz. Dr.rer.nat. Thomas Noll
Titel der Lehrveranstaltung: (Name der Umfrage)	Semantik und Verifikation von Software (10ss-29423) (Vorlesung)

Konzept der Vorlesung

Mir ist klar, wozu die Vorlesung gut ist.



Die Vorlesung hat eine klar erkennbare Struktur.



Die Vorlesung kann mit den zur Verfügung gestellten Materialien (Skript, Lehrbuch, Handouts ...) gut nachbereitet werden.



Ich habe das nötige Vorwissen für diese Vorlesung.



Die ausgewählten Beispiele helfen mir, die Inhalte der Vorlesung zu verstehen.



Es werden Zusammenfassungen an sinnvollen Stellen gemacht.



Der Schwierigkeitsgrad ist ...



Vermittlung und Verhalten

... kann den Stoff verständlich erklären.



... geht sorgfältig auf Verständnisfragen ein.



... berücksichtigt unterschiedliche Kenntnisstände der Studierenden.



... schafft es, mich für den Vorlesungsstoff zu begeistern.



... spricht angemessen laut und deutlich.



... ist offen für Verbesserungsvorschläge.



... lässt sich außerhalb der Vorlesung
gut ansprechen, z.B. in Sprechstunden oder
per Email.

trifft völlig
zu

trifft nicht zu
prg
max=1

Der Einsatz von Hilfsmitteln wie Wandtafel, Overhead,
Beamer und Demonstrationen ist gut.

trifft völlig
zu

trifft nicht zu
prg
max=1.2

Schrift und Zeichnungen in der Vorlesung sind gut lesbar.

trifft völlig
zu

trifft nicht zu
prg
max=1.2

Tafelanschrieb / Folien sind übersichtlich.

trifft völlig
zu

trifft nicht zu
prg
max=1.7

Das Tempo ist ...

zu hoch

zu niedrig
prg
max=2.2

Profillinie

Teilbereich:

- Name decides

↓ Titel der

Lehrveranstaltung:
(Name der Umfrage)

Informatik

Priv.-Doz. Dr.rer.nat. Thomas Noll

Semantik und Verifikation von Software (10ss-29422) (Übung)

Konzept der Übung

Vorlesung und Übung sind **inhaltlich** gut aufeinander abgestimmt.

Vorlesung und Übung sind **zeitlich** gut aufeinander abgestimmt.

Mir ist klar, wozu die Übung gut ist.

Der Ablauf der Übung ist gut strukturiert.

trifft völlig zu					trifft gar nicht zu	n=7 max=1,0
------------------	--	--	--	--	---------------------	----------------

Die ausgewählten Übungsaufgaben helfen mir, die Inhalte der Vorlesung zu verstehen.

Die Übungsaufgaben sind verständlich gestellt.

The diagram shows a horizontal scale with five points. The leftmost point is labeled 'trifft völlig zu' and the rightmost point is labeled 'trifft gar nicht zu'. A red dot is placed on the second point from the left. To the right of the scale, the text 'n=6' and 'max=2.5' is displayed.

Die Übungsaufgaben haben einen angemessenen Umfang.

Die vorgestellten Lösungswege sind nachvollziehbar.

trifft völlig zu | trifft gar nicht zu

Falls Sie Ihre Lösung abgeben konnten: Wurde diese angemessen korrigiert?

trifft völlig zu					trifft gar nicht zu
------------------	--	--	--	--	---------------------

$n=5$
 $\sum x_i = 1$

Die Übungsaufgaben sind ...

Vermittlung und Verhalten

... kann den Stoff verständlich erklären.

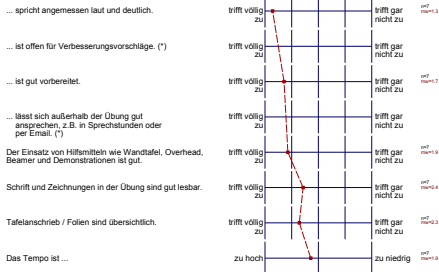
... geht sorgfältig auf Verständnisfragen ein.

trifft völlig zu					trifft gar nicht zu
------------------	--	--	--	--	---------------------

$n=7$
 $\bar{x}=1.6$

... berücksichtigt unterschiedliche
Kenntnisstände der Studierenden. (*)

trifft völlig zu					trifft gar nicht zu
------------------	--	--	--	--	---------------------



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