



Einführung in die Theoretische Informatik

David Drexel

Alexander Maringele

Julian Fodor

David Obwaller

Alexander Lochmann

Jonas Schöpf

Georg Moser

cbr.uibk.ac.at

Zusammenfassung der letzten LVA

Definition

- ein **Verifikator** einer Sprache $L \subseteq \Sigma^*$, ist ein Algorithmus V sodass
$$L = \{x \in \Sigma^* \mid \exists c, \text{ sodass } V \text{ akzeptiert Eingabe } (x, c)\}$$
- ein **polytime Verifikator** ist ein Verifikator mit (ungefährer) Laufzeit n^k wobei $\ell(x) = n$

Definition

NP ist die Klasse der Sprachen, die einen polytime Verifikator haben

Example

- Es gilt $\text{SAT} \in \text{NP}$; außerdem ist SAT NP-hart.
- Also ist SAT NP-vollständig.



Übersicht

Einführung in die Logik

Syntax & Semantik der Aussagenlogik, Formales Beweisen, KNF und DNF

Einführung in die Algebra

algebraische Strukturen, Beispiele von Algebren, Zusammenhang Boolesche Algebra und Aussagenlogik, Universelle Algebra, Satz von Birkhoff

Einführung in die Theorie der Formalen Sprachen

Grammatiken und Formale Sprachen, Chomsky-Hierarchie, Reguläre Sprachen, Kontextfreie Sprachen, Anwendungen von formalen Sprachen

Einführung in die Berechenbarkeitstheorie und Komplexitätstheorie

Algorithmisch unlösbare Probleme, Turing Maschinen, Registermaschinen, $P \neq NP$

Einführung in die Programmverifikation

Prinzipien der Analyse von Programmen, Verifikation nach Hoare

Einführung in die Logik

Syntax & Semantik der Aussagenlogik, Formales Beweisen, KNF und DNF

Einführung in die Algebra

algebraische Strukturen, Beispiele von Algebren, Zusammenhang Boolesche Algebra und Aussagenlogik, Universelle Algebra, Satz von Birkhoff

Einführung in die Theorie der Formalen Sprachen

Grammatiken und Formale Sprachen, Chomsky-Hierarchie, Reguläre Sprachen, Kontextfreie Sprachen, Anwendungen von formalen Sprachen

Einführung in die Berechenbarkeitstheorie und Komplexitätstheorie

Algorithmisch unlösbare Probleme, Turing Maschinen, Registermaschinen, $P \neq NP$

Einführung in die Programmverifikation

Prinzipien der Analyse von Programmen, Verifikation nach Hoare

Wozu Programmverifikation



- Ariane-5
- Fehler in der Datenkonvertierung
- USD 370 Millionen



- Intel Pentium FDIV-Bug
- Falsche Berechnungen
- USD 475 Millionen



- Gepäckverteilung (Denver)
- Desaster
- USD 560 Millionen



- Blue Screen of Death
- ziemlich lästig

Begutachtung

- Der Code wird von ähnlich qualifizierten Programmierern kontrolliert
- subtile Fehler werden leicht übersehen

Begutachtung

- Der Code wird von ähnlich qualifizierten Programmierern kontrolliert
- subtile Fehler werden leicht übersehen

Testen

- dynamische Technik, bei der das Programm ausgeführt wird
- Wie wird die richtige Testumgebung geschaffen?

Begutachtung

- Der Code wird von ähnlich qualifizierten Programmierern kontrolliert
- subtile Fehler werden leicht übersehen

Testen

- dynamische Technik, bei der das Programm ausgeführt wird
- Wie wird die richtige Testumgebung geschaffen?

Formal Methoden

- erlauben die frühe Integration der Verifikation in die Softwareentwicklung
- sind effizienter als andere Methoden (höhere Erkennensrate von Fehlern)
- sind (im Besonderen wenn automatisierbar) schneller anwendbar

Verifikation nach Hoare

Prädikatenlogik (informell)

- Die Prädikatenlogik ist eine Logik, deren Ausdruckskraft über die der Aussagenlogik weit hinausgeht
- Die wichtigste Erweiterung sind **Prädikatensymbole**
- **Prädikatensymbole** erlauben es uns, über Elemente einer Menge Aussagen zu treffen

Verifikation nach Hoare

Prädikatenlogik (informell)

- Die Prädikatenlogik ist eine Logik, deren Ausdruckskraft über die der Aussagenlogik weit hinausgeht
- Die wichtigste Erweiterung sind **Prädikatensymbole**
- **Prädikatensymbole** erlauben es uns, über Elemente einer Menge Aussagen zu treffen

Sprache einer Prädikatenlogik

Eine Prädikatenlogik durch eine **Sprache** beschrieben, diese Sprache enthält:

1 Funktionssymbole und Prädikatensymbole; Variablen

2 $\underbrace{=}_{\text{Gleichheit}}, \underbrace{\neg, \wedge, \vee, \rightarrow}_{\text{Junktoren}}, \underbrace{\forall, \exists}_{\text{Quantoren}}$

Beispiel

- Sei 7 eine Konstante und `ist_prim` ein Prädikatensymbol
- Wir schreiben `ist_prim(7)`, um auszudrücken, dass 7 eine Primzahl

Beispiel

- Sei 7 eine Konstante und ist_prim ein Prädikatensymbol
- Wir schreiben $\text{ist_prim}(7)$, um auszudrücken, dass 7 eine Primzahl

Definition

Ein Ausdruck der mit Hilfe von Variablen und Funktionssymbolen gebildet wird, heißt **Term**

Beispiel

- Sei 7 eine Konstante und ist_prim ein Prädikatensymbol
- Wir schreiben $\text{ist_prim}(7)$, um auszudrücken, dass 7 eine Primzahl

Definition

Ein Ausdruck der mit Hilfe von Variablen und Funktionssymbolen gebildet wird, heißt **Term**

Definition

- 1 Sei P ein Prädikatensymbol
- 2 Seien t_1, \dots, t_n Terme

Beispiel

- Sei 7 eine Konstante und ist_prim ein Prädikatensymbol
- Wir schreiben $\text{ist_prim}(7)$, um auszudrücken, dass 7 eine Primzahl

Definition

Ein Ausdruck der mit Hilfe von Variablen und Funktionssymbolen gebildet wird, heißt **Term**

Definition

- 1 Sei P ein Prädikatensymbol
- 2 Seien t_1, \dots, t_n Terme

Dann nennen wir die Ausdrücke $P(t_1, \dots, t_n)$ und $t_1 = t_2$ **Atome** oder **atomare Formel**

Definition (Zusicherungen)

Wir definieren **Zusicherungen** induktiv:

- 1 Atome sind Zusicherungen
- 2 Wenn A und B Zusicherungen sind, dann sind auch die folgenden Ausdrücke, Zusicherungen:

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

Definition (Zusicherungen)

Wir definieren **Zusicherungen** induktiv:

- 1 Atome sind Zusicherungen
- 2 Wenn A und B Zusicherungen sind, dann sind auch die folgenden Ausdrücke, Zusicherungen:

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

Konvention

Zusicherungen werden **Formeln** genannt

Definition (Zusicherungen)

Wir definieren **Zusicherungen** induktiv:

- 1 Atome sind Zusicherungen
- 2 Wenn A und B Zusicherungen sind, dann sind auch die folgenden Ausdrücke, Zusicherungen:

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

Konvention

Zusicherungen werden **Formeln** genannt

Definition

Interpretationen \mathcal{I} werden verwendet, um den Ausdrücken der Prädikatenlogik eine **Bedeutung** zu geben

Beispiel

- Wir betrachten die Konstante 7 und das Prädikat `ist_prim`
- Interpretation \mathcal{I} legt fest, dass 7 als die Zahl sieben zu verstehen ist
- \mathcal{I} legt fest, dass das Atom `ist_prim(n)` genau dann wahr ist, wenn n eine Primzahl

Beispiel

- Wir betrachten die Konstante 7 und das Prädikat `ist_prim`
- Interpretation \mathcal{I} legt fest, dass 7 als die Zahl sieben zu verstehen ist
- \mathcal{I} legt fest, dass das Atom `ist_prim(n)` genau dann wahr ist, wenn n eine Primzahl

Beobachtung

- 1 Mit Hilfe von Interpretationen wird der Wahrheitsgehalt von Atomen bestimmt
- 2 Ist die Wahrheit von Atomen in \mathcal{I} definiert, wird die Wahrheit einer beliebigen Formel durch die Bedeutung der Junktoren bestimmt

Beispiel

- Wir betrachten die Konstante 7 und das Prädikat `ist_prim`
- Interpretation \mathcal{I} legt fest, dass 7 als die Zahl sieben zu verstehen ist
- \mathcal{I} legt fest, dass das Atom `ist_prim(n)` genau dann wahr ist, wenn n eine Primzahl

Beobachtung

- 1 Mit Hilfe von Interpretationen wird der Wahrheitsgehalt von Atomen bestimmt
- 2 Ist die Wahrheit von Atomen in \mathcal{I} definiert, wird die Wahrheit einer beliebigen Formel durch die Bedeutung der Junktoren bestimmt

Beispiel

Die Formel `ist_prim(x) \wedge $x = 7$` bedeutet, dass x die Primzahl 7 ist

Definition

Sei \mathcal{I} eine Interpretation und F eine Formel, wir schreiben $\mathcal{I} \models F$, wenn die Formel F in der Interpretation \mathcal{I} wahr ist

Definition

Sei \mathcal{I} eine Interpretation und F eine Formel, wir schreiben $\mathcal{I} \models F$, wenn die Formel F in der Interpretation \mathcal{I} wahr ist

Beispiel

Wenn x die Primzahl 7 ist, dann gilt $\mathcal{I} \models \text{ist_prim}(x) \wedge x = 7$

Definition

Sei \mathcal{I} eine Interpretation und F eine Formel, wir schreiben $\mathcal{I} \models F$, wenn die Formel F in der Interpretation \mathcal{I} wahr ist

Beispiel

Wenn x die Primzahl 7 ist, dann gilt $\mathcal{I} \models \text{ist_prim}(x) \wedge x = 7$

Definition

Die **Konsequenzrelation** $A \models B$ gilt, gdw. für alle Interpretationen \mathcal{I} :

$$\mathcal{I} \models A \text{ impliziert } \mathcal{I} \models B$$

Definition

Sei \mathcal{I} eine Interpretation und F eine Formel, wir schreiben $\mathcal{I} \models F$, wenn die Formel F in der Interpretation \mathcal{I} wahr ist

Beispiel

Wenn x die Primzahl 7 ist, dann gilt $\mathcal{I} \models \text{ist_prim}(x) \wedge x = 7$

Definition

Die **Konsequenzrelation** $A \models B$ gilt, gdw. für alle Interpretationen \mathcal{I} :

$$\mathcal{I} \models A \text{ impliziert } \mathcal{I} \models B$$

Beispiel

Seien $x_1 > 4$ und $x_1 + 1 > 5$ Atome, es gilt:

$$x_1 > 4 \models x_1 + 1 > 5$$

Hoare-Tripel

Definition

- Sei P ein while-Programm (ein Programm einer Registermaschine)
- Seien Q und R Zusicherungen
- Ein **Hoare-Tripel** ist wie folgt definiert:

$$\{Q\} P \{R\}$$

- Q wird **Vorbedingung**
- R wird **Nachbedingung** genannt

Hoare-Tripel

Definition

- Sei P ein while-Programm (ein Programm einer Registermaschine)
- Seien Q und R Zusicherungen
- Ein **Hoare-Tripel** ist wie folgt definiert:

$$\{Q\} P \{R\}$$

- Q wird **Vorbedingung**
- R wird **Nachbedingung** genannt

Beispiel

Seien $x_1 > 4$, $x_1 > 5$ Zusicherungen und $x_1 := x_1 + 1$ ein Programm, dann ist $\{x_1 > 4\} x_1 := x_1 + 1 \{x_1 > 5\}$ ein Hoare-Tripel

Definition

- Ein Hoare-Tripel $\{Q\} P \{R\}$ ist **wahr**, wenn
 - 1 Q **vor** der Ausführung von P gilt
 - 2 R **nach** der Ausführung von P gilt
 - 3 unter der Voraussetzung, dass P **terminiert**

Definition

- Ein Hoare-Tripel $\{Q\} P \{R\}$ ist **wahr**, wenn
 - 1 Q **vor** der Ausführung von P gilt
 - 2 R **nach** der Ausführung von P gilt
 - 3 unter der Voraussetzung, dass P **terminiert**
- Wenn $\{Q\} P \{R\}$ wahr, dann ist P korrekt in Bezug auf Q und R
- Dann sagen wir auch P ist **partiell korrekt**

Definition

- Ein Hoare-Tripel $\{Q\} P \{R\}$ ist **wahr**, wenn
 - 1 Q **vor** der Ausführung von P gilt
 - 2 R **nach** der Ausführung von P gilt
 - 3 unter der Voraussetzung, dass P **terminiert**
- Wenn $\{Q\} P \{R\}$ wahr, dann ist P korrekt in Bezug auf Q und R
- Dann sagen wir auch P ist **partiell korrekt**
- Das Programm P ist **total korrekt**, wenn es partiell korrekt ist und terminiert

Definition

- Ein Hoare-Tripel $\{Q\} P \{R\}$ ist **wahr**, wenn
 - 1 Q **vor** der Ausführung von P gilt
 - 2 R **nach** der Ausführung von P gilt
 - 3 unter der Voraussetzung, dass P **terminiert**
- Wenn $\{Q\} P \{R\}$ wahr, dann ist P korrekt in Bezug auf Q und R
- Dann sagen wir auch P ist **partiell korrekt**
- Das Programm P ist **total korrekt**, wenn es partiell korrekt ist und terminiert

Beispiel

Die folgenden Hoare-Tripel

$$\{x_1 > 4\} x_1 := x_1 + 1 \{x_1 > 5\} \quad \{x_2 = 0\} x_2 := x_2 - 1 \{x_2 = 0\}$$

sind wahr und die jeweiligen Programme total korrekt

Hoare-Kalkül

Definition

Die Regeln des **Hoare-Kalkül** sind wie folgt definiert:

Definition

Die Regeln des **Hoare-Kalkül** sind wie folgt definiert:

$$[z] \quad \overline{\{Q\{x \mapsto t\}\} x := t \{Q\}}$$

Hoare-Kalkül

Definition

Die Regeln des **Hoare-Kalkül** sind wie folgt definiert:

$$[z] \quad \overline{\{Q\{x \mapsto t\}\} x := t \{Q\}} \quad [a] \quad \frac{\{Q'\} P \{R'\}}{\{Q\} P \{R\}} \quad Q \models Q', R' \models R$$

Definition

Die Regeln des **Hoare-Kalkül** sind wie folgt definiert:

$$\begin{array}{l} [z] \quad \frac{}{\{Q\{x \mapsto t\}\} x := t \{Q\}} \quad [a] \quad \frac{\{Q'\} P \{R'\}}{\{Q\} P \{R\}} \quad Q \models Q', R' \models R \\ [s] \quad \frac{\{Q\} P_1 \{R\} \quad \{R\} P_2 \{S\}}{\{Q\} P_1; P_2 \{S\}} \end{array}$$

Definition

Die Regeln des **Hoare-Kalkül** sind wie folgt definiert:

$$\begin{array}{ll} [z] & \overline{\{Q\{x \mapsto t\}\} x := t \{Q\}} \\ [a] & \frac{\{Q'\} P \{R'\}}{\{Q\} P \{R\}} \quad Q \models Q', R' \models R \\ [s] & \frac{\{Q\} P_1 \{R\} \quad \{R\} P_2 \{S\}}{\{Q\} P_1; P_2 \{S\}} \\ [w] & \frac{\{I \wedge B\} P \{I\}}{\{I\} \text{ while } B \text{ do } P \text{ end } \{I \wedge \neg B\}} \end{array}$$

Hoare-Kalkül

Definition

Die Regeln des **Hoare-Kalkül** sind wie folgt definiert:

$$\begin{array}{l} [z] \quad \overline{\{Q\{x \mapsto t\}\} x := t \{Q\}} \quad [a] \quad \frac{\{Q'\} P \{R'\}}{\{Q\} P \{R\}} \quad Q \models Q', R' \models R \\ [s] \quad \frac{\{Q\} P_1 \{R\} \quad \{R\} P_2 \{S\}}{\{Q\} P_1; P_2 \{S\}} \quad [w] \quad \frac{\{I \wedge B\} P \{I\}}{\{I\} \text{ while } B \text{ do } P \text{ end } \{I \wedge \neg B\}} \end{array}$$

Ist ein Hoare-Tripel in diesem Kalkül ableitbar, dann ist es wahr

Hoare-Kalkül

Definition

Die Regeln des **Hoare-Kalkül** sind wie folgt definiert:

$$\begin{array}{l} [z] \quad \frac{}{\{Q\{x \mapsto t\}\} x := t \{Q\}} \quad [a] \quad \frac{\{Q'\} P \{R'\}}{\{Q\} P \{R\}} \quad Q \models Q', R' \models R \\ [s] \quad \frac{\{Q\} P_1 \{R\} \quad \{R\} P_2 \{S\}}{\{Q\} P_1; P_2 \{S\}} \quad [w] \quad \frac{\{I \wedge B\} P \{I\}}{\{I\} \text{ while } B \text{ do } P \text{ end } \{I \wedge \neg B\}} \end{array}$$

Ist ein Hoare-Tripel in diesem Kalkül ableitbar, dann ist es wahr

Beispiel

$$\frac{\{x_1 + 1 > 5\} x_1 := x_1 + 1 \{x_1 > 5\}}{\{x_1 > 4\} x_1 := x_1 + 1 \{x_1 > 5\}} \quad [z]$$

$[a], x_1 > 4 \models x_1 + 1 > 5$

Wir betrachten das folgende einfache while-Programm P :

```
while  $x_i \neq 0$  do  
   $x_i := x_i - 1$   
end
```

Wir betrachten das folgende einfache while-Programm P :

```
while  $x_i \neq 0$  do  
   $x_i := x_i - 1$   
end
```

und zeigen $\{x_i \geq 0\} P \{x_i = 0\}$

Beispiel

Wir betrachten das folgende einfache while-Programm P :

```
while  $x_i \neq 0$  do  
   $x_i := x_i - 1$   
end
```

und zeigen $\{x_i \geq 0\} P \{x_i = 0\}$

$$\frac{\frac{\frac{\{x_i - 1 \geq 0\} x_i := x_i - 1 \{x_i \geq 0\}}{\{x_i \geq 0 \wedge x_i \neq 0\} x_i := x_i - 1 \{x_i \geq 0\}}}{\{x_i \geq 0\} P \{x_i \geq 0 \wedge x_i = 0\}}}{\{x_i \geq 0\} P \{x_i = 0\}} \begin{matrix} [z] \\ [a] \\ [w] \\ [a] \end{matrix}$$

Beispiel

Wir betrachten das folgende einfache while-Programm P :

```
while  $x_i \neq 0$  do  
   $x_i := x_i - 1$   
end
```

und zeigen $\{x_i \geq 0\} P \{x_i = 0\}$

$$\frac{\frac{\frac{\{x_i - 1 \geq 0\} x_i := x_i - 1 \{x_i \geq 0\}}{\{x_i \geq 0 \wedge x_i \neq 0\} x_i := x_i - 1 \{x_i \geq 0\}}}{\{x_i \geq 0\} P \{x_i \geq 0 \wedge x_i = 0\}}}{\{x_i \geq 0\} P \{x_i = 0\}}$$

[z]
[a]
[w]
[a]

wir verwenden:

- 1 $x_i \geq 0 \wedge x_i = 0 \models x_i = 0$
- 2 die Schleifeninvariante $x_i \geq 0$
- 3 $x_i \geq 0 \wedge x_i \neq 0 \models x_i - 1 \geq 0$

Beispiel

Wir betrachten das folgende einfache while-Programm P :

```
while  $x_i \neq 0$  do
   $x_i := x_i - 1$ 
end
```

und zeigen $\{x_i \geq 0\} P \{x_i = 0\}$

$$\frac{\frac{\frac{\{x_i - 1 \geq 0\} x_i := x_i - 1 \{x_i \geq 0\}}{\{x_i \geq 0 \wedge x_i \neq 0\} x_i := x_i - 1 \{x_i \geq 0\}} [z]}{\{x_i \geq 0\} P \{x_i \geq 0 \wedge x_i = 0\}} [a]}{\{x_i \geq 0\} P \{x_i = 0\}} [w]$$

wir verwenden:

1 $x_i \geq 0 \wedge x_i = 0 \models x_i = 0$

2 die Schleifeninvariante $x_i \geq 0$

3 $x_i \geq 0 \wedge x_i \neq 0 \models x_i - 1 \geq 0$

Beispiel

Wir betrachten das folgende einfache while-Programm P :

```
while  $x_i \neq 0$  do  
   $x_i := x_i - 1$   
end
```

und zeigen $\{x_i \geq 0\} P \{x_i = 0\}$

$$\frac{\frac{\frac{\{x_i - 1 \geq 0\} x_i := x_i - 1 \{x_i \geq 0\}}{\{x_i \geq 0 \wedge x_i \neq 0\} x_i := x_i - 1 \{x_i \geq 0\}} [z]}{\{x_i \geq 0\} P \{x_i \geq 0 \wedge x_i = 0\}} [a]}{\{x_i \geq 0\} P \{x_i = 0\}} [w]$$

wir verwenden:

- 1 $x_i \geq 0 \wedge x_i = 0 \models x_i = 0$
- 2 die Schleifeninvariante $x_i \geq 0$
- 3 $x_i \geq 0 \wedge x_i \neq 0 \models x_i - 1 \geq 0$

Beispiel

Wir betrachten das folgende einfache while-Programm P :

```
while  $x_i \neq 0$  do  
   $x_i := x_i - 1$   
end
```

und zeigen $\{x_i \geq 0\} P \{x_i = 0\}$

$$\frac{\frac{\frac{\{x_i - 1 \geq 0\} x_i := x_i - 1 \{x_i \geq 0\}}{\{x_i \geq 0 \wedge x_i \neq 0\} x_i := x_i - 1 \{x_i \geq 0\}} [z]}{\{x_i \geq 0\} P \{x_i \geq 0 \wedge x_i = 0\}} [a]}{\{x_i \geq 0\} P \{x_i = 0\}} [w]$$

wir verwenden:

- 1 $x_i \geq 0 \wedge x_i = 0 \models x_i = 0$
- 2 die Schleifeninvariante $x_i \geq 0$
- 3 $x_i \geq 0 \wedge x_i \neq 0 \models x_i - 1 \geq 0$



Automatisierung

Frage

Kann Verifikation nach Hoare vollständig automatisiert werden?

Frage

Kann Verifikation nach Hoare vollständig automatisiert werden?

Antwort

Nein

- ein total korrektes Programm muss als **terminierend** nachgewiesen werden
- Termination ist ein unentscheidbares Problem

Frage

Kann Verifikation nach Hoare vollständig automatisiert werden?

Antwort

Nein

- ein total korrektes Programm muss als **terminierend** nachgewiesen werden
- Termination ist ein unentscheidbares Problem

Frage

Was tun?

Frage

Kann Verifikation nach Hoare vollständig automatisiert werden?

Antwort

Nein

- ein total korrektes Programm muss als **terminierend** nachgewiesen werden
- Termination ist ein unentscheidbares Problem

Frage

Was tun?

Antwort

- Entweder wird Termination vorausgesetzt und muss dann per Hand bewiesen werden
- Oder der Verifikator für Termination ist partiell

Bemerkung

Termination von imperativen Programmen kann etwa von folgenden Tools gezeigt werden

AProVE, COSTA, Julia, SACO, SPEED, Terminator, T₁T₂, ...

Bemerkung

Termination von imperativen Programmen kann etwa von folgenden Tools gezeigt werden

AProVE, COSTA, Julia, SACO, SPEED, Terminator, T₁T₂, ...

Bemerkung

Neben der Termination können auch andere (unentscheidbare) Eigenschaften von Programmen automatisch verifiziert werden, etwa der **Speicherbedarf**, die **Laufzeitkomplexität**, etc.

AProVE, COSTA, LOOPUS, RaML, SPEED, T_CT, ...

Bemerkung

Termination von imperativen Programmen kann etwa von folgenden Tools gezeigt werden

AProVE, COSTA, Julia, SACO, SPEED, Terminator, T₁T₂, ...

Bemerkung

Neben der Termination können auch andere (unentscheidbare) Eigenschaften von Programmen automatisch verifiziert werden, etwa der **Speicherbedarf**, die **Laufzeitkomplexität**, etc.

AProVE, COSTA, LOOPUS, RaML, SPEED, T_CT, ...

NB: diese Tools **müssen** unvollständig sein



Prüfungsorganisation und -vorbereitung

Organisation der Klausur

1te Klausur

- Bitte registrieren Sie sich online für die Klausur
- Die Klausur findet im **Großen Hörsaal** (A-K) und im **HS B** (L-Z) statt
- Prüfungsstoff ist alles **außer heute**
- Die Prüfung is **closed-book**: keine Unterlagen, keine Taschenrechner, etc.
- Alte Klausuren (plus Musterlösungen) sind online

Organisation der Klausur

1te Klausur

- Bitte registrieren Sie sich online für die Klausur
- Die Klausur findet im **Großen Hörsaal** (A-K) und im **HS B** (L-Z) statt
- Prüfungsstoff ist alles **außer heute**
- Die Prüfung ist **closed-book**: keine Unterlagen, keine Taschenrechner, etc.
- Alte Klausuren (plus Musterlösungen) sind online

2te und 3te Klausur

- Die 2te Klausur bzw. 3te Klausur findet Anfang Sommersemester bzw. im September statt; genaue Information: OLAT
- Bitte melden Sie sich dann **rechtzeitig** online für die Klausur an
- Prüfungsstoff ist alles

Organisation der SL Klausur

SL Klausur am Freitag, den 31. Jänner

- SL Klausur findet am **31.1.** zum Zeitpunkt der SL statt
- Die anschließende Vorlesungsklausur aus FP beginnt erst um 13:30
- Prüfungstoff ist **alles**

Organisation der SL Klausur

SL Klausur am Freitag, den 31. Jänner

- SL Klausur findet am **31.1.** zum Zeitpunkt der SL statt
- Die anschließende Vorlesungsklausur aus FP beginnt erst um 13:30
- Prüfungsstoff ist **alles**

Keine weitere SL Klausur

- Bei Krankheit bitte ärztliche Bestätigung vorlegen, dann werden die entsprechenden Fälle individuell gelöst
- Die LVA wird auch im Sommersemester angeboten werden (da Teil der STEOP)

Prüfungsvorbereitung

Feedback

Organisation

Gibt es einen besonderen Grund dafür, dass die Lösungen der Altklausuren nie jene der 1. bis 6. Aufgabe beinhalten?

Antwort

Richtige Antworten für die Multiple-Choice Fragen auf den Angaben (letztes Blatt)

Feedback

Organisation

Gibt es einen besonderen Grund dafür, dass die Lösungen der Altklausuren nie jene der 1. bis 6. Aufgabe beinhalten?

Antwort

Richtige Antworten für die Multiple-Choice Fragen auf den Angaben (letztes Blatt)

Formales Beweisen

Kann das formale Beweisen nur Tautologien ableiten?

Antwort

Ja.