universität
innsbruck

Einführung in die Theoretische Informatik

Martin Avanzini Christian Dalvit Jamie Hochrainer
Georg Moser Johannes Niederhauser Jonas Schöpf

`https://tcs-informatik.uibk.ac.at`

universität innsbruck

Zusammenfassung

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 $|x| = n$

Definition

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

Example

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

Hoare-Kalkül

Definition

Die Regeln des Hoare-Kalkül sind wie folgt definiert:

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

$$[s] \quad \frac{\{Q\}\ P_1\ \{R\} \quad \{R\}\ P_2\ \{S\}}{\{Q\}\ P_1; P_2\ \{S\}} \qquad [w] \quad \frac{\{I \wedge B\}\ P\ \{I\}}{\{I\}\ \text{while}\ B\ \text{do}\ P\ \text{end}\ \{I \wedge \neg B\}}$$

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

Beispiel

$$\cfrac{\cfrac{}{\{x_1 + 1 > 5\}\ x_1 := x_1 + 1\ \{x_1 > 5\}}\ [z]}{\{x_1 > 4\}\ x_1 := x_1 + 1\ \{x_1 > 5\}}\ [a], x_1 > 4 \models x_1 + 1 > 5$$

universität innsbruck

Komplexitätstheorie

Beispiel

Wir betrachten das Rucksackproblem

- gegeben, n verschiedene Gegenstände mit einem bestimmten Gewicht g_i und Wert w_i $(1 \leqslant i \leqslant n)$
- gesucht, eine optimale Auswahl der Gegenstände, sodass Wert maximiert und Gewicht minimiert

Beispiel (Fortsetzung)

Das Rucksackproblem als Entscheidungsproblem mit Maximalgewicht G und Minimalwert W

$$\text{KNAPSACK} := \{((g_i)_{1 \leqslant i \leqslant n}, (w_i)_{1 \leqslant i \leqslant n}, G, W) \mid \text{es existiert } I \text{ mit } \sum_{j \in I} g_j \leqslant G, \sum_{j \in I} w_j \geqslant W\}$$

Hier gilt $I \subseteq \{i \mid 1 \leqslant i \leqslant n\}$. Wir zeigen, dass KNAPSACK $\in$ NP

Beispiel (Fortsetzung)

- zunächst betrachten wir lösbare und unlösbare Instanzen
- sei etwa $n = 3$ mit $g_1 = g_2 = g_3 = 1$, $w_1 = w_2 = w_3 = 1$ und Minimalwert $W = 3$, aber Maximalgewicht $G < 3$; dann müssen alle drei Gegenstände in I gewählt werden, damit $\sum_{j \in I} w_j \geqslant 3$
- anderseits gilt aber dann $\neg(\sum_{j \in I} g_j < 3)$; diese Instanz ist also nicht lösbar
- eine lösbare Instanz für $n = 3$, wäre etwa $g_1 = 3$, $g_2 = 2$, $q_3 = 1$ $w_1 = w_2 = w_3 = 1$ mit $G = 2$ und $W = 1$
- nun überlegen wir welches Zertifikat wir wählen sollten, damit die Aufgabe leicht (= in polynomieller Zeit) lösbar wird
- gegeben Indexmenge I, ist es offensichtlich einfach zu prüfen, dass gilt:
  1. $\sum_{j \in I} g_j \leqslant G$
  2. $\sum_{j \in I} w_j \geqslant W$
- also wählen wir I als Zertifikat, der polytime Verifier nimmt dann die gegebene Instanz $((g_i)_{1 \leqslant i \leqslant n}, (w_i)_{1 \leqslant i \leqslant n}, G, W)$ und prüft die Bedingungen

Formale Sprachen und Komplexitätstheorie

- die Chomskyhierarchie beschreibt den Zusammenhang zwischen Klassen von formalen Sprachen (regulär, kontextfrei, . . .)
- ähnlich bezeichnen Komplexitäsklassen, Klassen von formalen Sprachen (P, NP, . . .)

Satz

es gelten die

- *Die Klasse der regulär Sprachen ist in* P *enthalten*
- *Die Klasse der kontextfreien Sprachen ist auch in* P *enthalten*
- *Die Klasse der kontextsensitiven Sprachen ist in* EXPTIME *enthalten, wobei*

$$\mathsf{EXPTIME} := \bigcup_{k \geqslant 1} \mathsf{DTIME}(2^{k \cdot n})$$

Komplexitätstheorie, graphisch

"I can't find an efficient algorithm, but neither can all these famous people."

universität
innsbruck

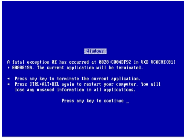
Programmverifikation

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

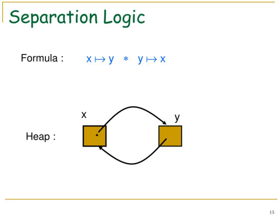
Testen

- 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

Beispiele formalen Softwareverifikation



Beispiel (Hoare Kalkül)

Gegeben seien P, Q und R:

$$(P)\ x_1 := x_1 - 1;\ x_1 := x_1 - 1;\ x_1 := x_1 - 1$$

$$(Q)\ x_1 = 10$$

$$(R)\ \mathit{prim}(x_1)$$

- das Prädikatensymbol $\mathit{prim}(x)$ ist wahr, wenn x eine Primzahl ist
- wir zeigen totale Korrektheit von P

Beispiel (Fortsetzung)

Zunächst betrachten wir die korrekte Ableitung Π_1 für die erste Zuweisung

$$\frac{}{\{x_1 - 1 = 9\}\ x_1 := x_1 - 1\ \{x_1 = 9\}}\ [z]$$

in ähnlicher Weise können wir Ableitungen für die zweite und dritte Zuweisung, bezeichnet mit Π_2, Π_3 definieren

Beispiel (Fortsetzung)

$$\dfrac{\dfrac{\overline{\{x_1 - 1 = 9\}\ x_1 := x_1 - 1\ \{x_1 = 9\}}\ [z]}{\{x_1 = 10\}\ x_1 := x_1 - 1\ \{x_1 = 9\}}\ [a] \qquad \Psi}{\{x_1 = 10\}\ x_1 := x_1 - 1; x_1 := x_1 - 1; x_1 := x_1 - 1\ \{\mathrm{prim}(x_1)\}}\ [s]$$

wobei Ψ wie folgt definiert ist

$$\dfrac{\dfrac{\overline{\{x_1 - 1 = 8\}\ x_1 := x_1 - 1\ \{x_1 = 8\}}\ [z]}{\{x_1 = 9\}\ x_1 := x_1 - 1\ \{x_1 = 8\}}\ [a] \qquad \dfrac{\overline{\{\mathrm{prim}(x_1 - 1)\}\ x_1 := x_1 - 1\ \{\mathrm{prim}(x_1)\}}\ [z]}{\{x_1 = 8\}\ x_1 := x_1 - 1\ \{\mathrm{prim}(x_1)\}}\ [a]}{\{x_1 = 9\}\ x_1 := x_1 - 1; x_1 := x_1 - 1\ \{\mathrm{prim}(x_1)\}}\ [s]$$

wir verwenden die folgenden Konsequenzrelation bei den Abschwächungen

- $(x_1 = 10) \models (x_1 - 1 = 9)$
- $(x_1 = 9) \models (x_1 - 1 = 8)$
- $(x_1 = 8) \models (\mathrm{prim}(x_1 - 1))$

universität
innsbruck

Prüfungsorganisation und -vorbereitung

Organisation der Vorlesungklausur, 11. Feber

1te Klausur

- Bitte registrieren Sie sich heute (!) online für die Klausur
- Die Klausur findet in Präsenz von 10:15-11:45 im Großen Hörsaal und HSB 3 statt
- Aufteilung auf die Hörsäle anhand des Familiennamens:
  - Studierende deren Familiennamen mit A-K angefängt: Großer Hörsaal
  - Studierende deren Familiennamen L-Z: HSB 3
- Studierendenausweis wird vor der Klausur kontrolliert, ohne Anmeldung keine Klausur
- Prüfungsstoff ist alles
- Die Prüfung is open-book: Unterlagen sind erlaubt
- Alte Klausuren (inkl. Musterlösungen) sind online

2te und 3te Klausur

- Die 2te Klausur findet am 11. März, die 3te Klausur im Juni/Juli statt
- Bitte melden Sie sich dann rechtzeitig online für die Klausur an
- Es besteht die Möglichkeit die SL nochmals im Sommersemester zu besuchen, um sich für die 3te Klausur vorzubereiten

10 Multiple-Choice Fragen

- jeweils 5-10 Antwortmöglichkeiten
- die Anzahl der richtigen Antworten ist nicht angegeben
- Korrekte Antworten werden positiv, falsche Antwort negativ bewertet
- Die Aufgaben sind randomisiert
- Während der Klausur können Sie sich bei Unklarheiten an die Aufsichtspersonen richten, Fragen zum Stoff werden natürlich nicht beantwortet

Organisation der SL Klausur

SL Klausur am Freitag, den 4. Feber

- SL Klausur findet am 4.2. entweder in Präsenz in den jeweilen Gruppen oder online (14:15–15:00) statt
- In jedem Fall dauert die Klausur 45’ und ist open-book, dh. es dürfen Unterlagen verwendet werden; es ist keine weitere Anmeldung erforderlich
- Es werden drei Multiple-Choice Fragen zu lösen sein mit größerem Umfang als in der Vorlesungsklausur

Zweite SL Klausur

- Die zweite SL Klausur findet am 4. März statt; der genaue Termin wird noch bekanntgegeben
- Sie können die SL auch im Rahmen der LVA im Sommersemester nochmals besuchen (Teil der STEOP)

universität innsbruck

Prüfungsvorbereitung