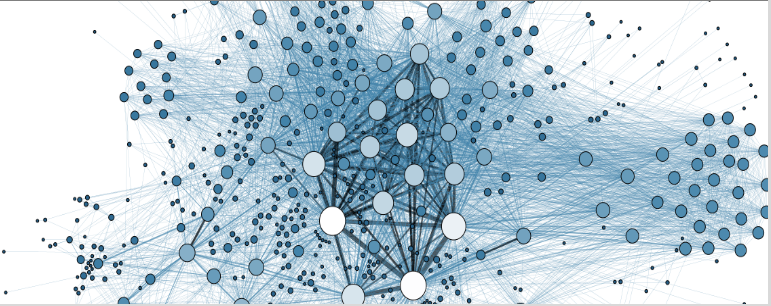
Karlsruher Institut für Technologie

Grundbegriffe der Informatik
Tutorium 33

Lukas Bach, lukas.bach@student.kit.edu | gbi.lukasbach.com

TUTORIUM IM WINTERSEMESTER 2016/2017



Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 28.10.2016

Lukas Bach, lu-
kas.bach@student.kit.edu

Organisatorisches

Lukas Bach, lukas.bach@student.kit.edu

Termine

- Vorlesung und Übung
  - Mittwoch 9:45 - 11:15 Vorlesung
  - Freitag 9:45 - 11:15 abwechselnd Vorlesung und Übung
- Tutorium
  - Donnerstags, 14:00 - 15:30
  - 50.34 Informatikbau, -107
- Übungsblätter
  - Alle zwei Wochen
  - Ausgabe Mittwochs, Abgabe Donnerstags bis 16:00 zwei Wochen drauf

Lukas Bach, lukas.bach@student.kit.edu

Übungsschein

- min. 50% aller Punkte auf Übungsblättern richtig
- Rückgabe im Tutorium
- Bestehen ist *keine* Voraussetzung für die Klausur, *aber* fürs Modul!
- Gemeinsames Abgeben, Abschreiben verboten
- Übungsblätter und später auch Musterlösungen im ILIAS

Lukas Bach, lukas.bach@student.kit.edu

Tutorium

- Alle Tutorienfolien auf:

  `http://gbi.lukasbach.com`

- Bei Fragen: lukas.bach@student.kit.edu
- Keine Anwesenheitspflicht
- Möglichkeit andere Tutorien zu besuchen

Lukas Bach, lukas.bach@student.kit.edu

Signale und Nachrichten

Lukas Bach, lukas.bach@student.kit.edu

Signale und Nachrichten

- Objekt: 101
  - Eins null eins oder 101 als Zahl oder 5 in binär oder zwei merkwürdige Striche mit einem Kreis dazwischen?
  - Vom Kontext abhängig.
  - Zunächst einfach ein konkretes Objekt.

Signale und Nachrichten

Lukas Bach, lukas.bach@student.kit.edu

- Signal
  - Physikalische Veränderung
  - Lässt sich verschieden interpretieren.
  - Beispiele:
    - Notfallalarm in Serverraum

      
    - Für Besucher nur schönes Leuchten
    - Für Security die Information, zu kommen
    - Für Techniker die Information, Ausrüstung zu holen
- Nachricht: Objekt wie oben, das von Signal unabhängig ist
  - Roter Notfallalarm ist ein anderes Signal als ein blauer Notfallalarm, aber vielleicht dieselbe Nachricht.

Lukas Bach, lukas.bach@student.kit.edu

Signale und Nachrichten

- Der interessante Teil: Informationen
- Bedeutung einer Nachricht
- Der vorher fehlende Kontext.
- Im obigen Beispiel:
  - Rote Alarmleuchte ist ein Signal (blaue Signalleuchte in Raum nebendran vielleicht auch)
  - “Alarm”: Nachricht
  - Information: Security soll herkommen, Techniker sollen das Werkzeug bereit halten, Besucher sollten Platz machen.

Lukas Bach, lu-
kas.bach@student.kit.edu

Mengen

Lukas Bach, lukas.bach@student.kit.edu

Mengen

- Erster wirklich wichtiger Teil.

Lukas Bach, lukas.bach@student.kit.edu

Mengen

Zeichnung

Lukas Bach, lukas.bach@student.kit.edu

Mengen

Definition: Mengen

"Unter einer Menge verstehen wir jede Zusammenfassung von bestimmten wohlunterschiedenen Objekten unserer Anschauung oder unseres Denkens (welche die Elemente dieser Menge genannt werden) zu einem Ganzen."

- Beispiel: $\{a, b, c, d\} =: A$ $\{a, c, 4\} =: B, \{10, 11\} =: C$
- Das Objekt c ist in A enthalten: $c \in A$, $c \in B$, $c \notin C$
- Reihenfolge gleich: $\{a, b\} = \{b, a\}$
- Elemente doppelt? $\{a, a, b, a\} = \{a, b\}$

Lukas Bach, lukas.bach@student.kit.edu

Mehr über Mengen

- Kardinalität oder Größe: Die Anzahl der Elemente der Menge
  - $A := \{a, b, c\}$. $|A| = 3$
  - $B := \{c, d\}$. $|B| = 2$
  - Was ist $|\{1, 2, 3, 2\}|$? 3!
  - Was ist $|\{\}|$? 0

Leere Menge

Die Menge, die nichts enthält, nennen wir die leere Menge, und schreiben sie als $\{\}$ oder $\emptyset$.

Was ist $|\{\{\}\}|$? 1! $\{\emptyset\}$ enthält eine leere Menge, die selbst ein Element ist.

Lukas Bach, lukas.bach@student.kit.edu

Mehr über Mengen

Zeichnung

Lukas Bach, lukas.bach@student.kit.edu

Mehr über Mengen

Karlsruher Institut für Technologie

Seien $A := \{a, b, c\}$, $B := \{b, c\}$, $C := \{c, b\}$, $D := \{b, c, d\}$.

- Teilmenge: $A \subseteq B$, also A ist Teilmenge von B genau dann, wenn alle Elemente aus A auch in B sind.
- Echte Teilmenge: $A \subset B$ genau dann, wenn $A \subseteq B$ *und* $A \neq B$.
  - Beispiele: $B \subseteq A$, sogar $B \subset A$.
    $C \subseteq B$ und $B \subseteq C$, aber $C \not\subset B$ und $B \not\subset C$.
- Schnittmenge: $A \cap B = \{b, c\}$.
  $A \cap B$ enthält *genau* die Elemente, die in A *und* in B sind.
- Vereinigungsmenge: $A \cup D = \{a, b, c, d\}$.
  $A \cup B$ enthält *genau* die Elemente, die in A *oder* in B sind.
- Mengendifferenz: $A \setminus B = \{a\}$, also alle Elemente in A, die nicht in B sind.
- Komplementärmenge: $\bar{A}$ enthält alle Elemente des *Universums*, die nicht in A sind. Angenommen, Universum = Lateinisches Alphabet:
  $\bar{A} = \{d, e, f, g, \dots, y, z\}$

Lukas Bach, lukas.bach@student.kit.edu

Potenzmenge

Karlsruher Institut für Technologie

Potenzmenge

Die Potenzmenge 2^M einer Menge M enthält genau alle Mengen, die Teilmenge von M sind.

Was bedeutet das allgemein?

- $M \in 2^M$
- $\emptyset \in 2^M$
- Konkretes Beispiel: Was ist 2^M mit $M = \{0, 1\}$?
    - Natürlich $\emptyset \in 2^M$ und $\{0, 1\} \in 2^M$.
    - $\{0\} \in 2^M$ und $\{1\} \in 2^M$.
    - Weitere? Nein, diese vier Mengen sind alle möglichen Teilmengen.
    - $\Rightarrow 2^M = \{\{\}, \{0\}, \{1\}, \{0, 1\}\}$.

Lukas Bach, lukas.bach@student.kit.edu

Potenzmenge

$M = \{0, 1\}$, $2^M = \{\{\}, \{0\}, \{1\}, \{0, 1\}\}$.
Was ist 2^{2^M}?

- Also $2^{\{\{\},\{0\},\{1\},\{0,1\}\}}$.
- Natürlich $\emptyset \in 2^M$ und $2^M = \{\{\}, \{0\}, \{1\}, \{0, 1\}\} \in 2^{2^M}$.

$2^{2^M} = \{$
$\quad \{\},$
$\quad \{\{\}\}, \{\{0\}\}, \{\{1\}\}, \{\{0, 1\}\},$
$\quad \{\{\}, \{0\}\}, \{\{\}, \{1\}\}, \{\{\}, \{0, 1\}\}, \{\{0\}, \{1\}\},$
$\qquad \{\{0\}, \{0, 1\}\}, \{\{1\}, \{0, 1\},$
$\quad \{\{\}, \{0\}, \{1\}\}, \{\{\}, \{0\}, \{0, 1\}\}, \{\{\}, \{1\}, \{0, 1\}\}, \{\{0\}, \{1\}, \{0, 1\}\},$
$\quad \{\{\}, \{0\}, \{1\}, \{0, 1\}\}$
$\}$

Lukas Bach, lu-
kas.bach@student.kit.edu

Alphabete

Lukas Bach, lukas.bach@student.kit.edu

Alphabete

Karlsruher Institut für Technologie

Alphabet

Ein Alphabet ist eine *endliche*, *nichtleere* Menge von Zeichen.

Was davon sind Alphabete? $\{d, 34, \pi, \%\}$, $\{a, b, c, \dots, y, z\}$, $\emptyset$, $\mathbb{N}$.

- $\{d, 34, \pi, \%\}$ und $\{a, b, c, \dots, y, z\}$ sind Alphabete.
- $\emptyset$ ist leer und damit kein Alphabet.
- $\mathbb{N} = \{1, 2, 3, \dots\}$ enthält alle natürlichen Zahlen und ist damit nicht endlich, also kein Alphabet.
- $\{0, 1\}$ ist das Alphabet, das alle Binärzahlen enthält.
- $\{\cdot, +, -, /\} =: R$ ist ein Alphabet von Rechenzeichen. $R \cup \{0, 1, \dots, 9\}$ ist ein Alphabet, das ein Taschenrechner als Eingabealphabet benutzen könnte.

Lukas Bach, lukas.bach@student.kit.edu

Relationen und Abbildungen

Lukas Bach, lukas.bach@student.kit.edu

Paare und Tupel

Karlsruher Institut für Technologie

Paar

Ein Paar ist eine geordnete Menge der Kardinalität 2.

Schreibweise mit runden Klammern ().

- Beispiel: $(a, 4) \neq (4, a)$
- Beispiel für eine Menge aus Tupeln: $\{(``AgeOfEmpires'', ``Strategie''), (``Battlefield'', ``Shooter''), (``SeriousSam'', ```Shooter'')\}$

Lukas Bach, lukas.bach@student.kit.edu

Tupel

Karlsruher Institut für Technologie

Tupel

Ein Tupel ist eine geordnete Menge. Konkret ist ein n-Tupel ein Tupel der Kardinalität n.

Also wie ein Paar, nur mit beliebiger Kardinalität. Ein Paar ist spezifisch ein 2-Tupel.
Beispiel: $(4tb, 512gb, 128gb, 4mb) \neq (512gb, 4mb, 4tb, 128gb)$.

Lukas Bach, lukas.bach@student.kit.edu

Kartesisches Produkt

Karlsruher Institut für Technologie

Zwei Mengen: $A := \{a, b, c\}$ und $B := \{1, 2, 3\}$.
Wir wollen alle Tupel mit erstem Element aus A und zweiten Element aus B.

$$\{ \quad (a, 1), (a, 2), (a, 3), \qquad (b, 1), (b, 2), (b, 3), \qquad (c, 1), (c, 2), (c, 3) \quad \}$$
$$= A \times B$$

Kreuzprodukt von zwei Mengen

Zu zwei Mengen A und B ist das Kreuzprodukt definiert als Menge aller Paare (a, b) mit $a \in A$ und $b \in B$.

Lukas Bach, lukas.bach@student.kit.edu

Kreuzprodukt

Karlsruher Institut für Technologie

Kreuzprodukt von zwei Mengen

Zu zwei Mengen A und B ist das Kreuzprodukt $A \times B$ definiert als Menge aller Paare (a, b) mit $a \in A$ und $b \in B$.

Kreuzprodukt von n Mengen

Zu n Mengen $M_1, M_2, \ldots, M_n$ ist das Kreuzprodukt $M_1 \times M_2 \times \cdots \times M_n$ definiert als Menge aller n-Tupel $(e_1, e_2, \ldots, e_n)$ mit $e_1 \in M_1$, $e_2 \in M_2, \ldots,$ $e_n \in M_n$.

Mengenpotenz

$$\underbrace{A \times A \times \cdots \times A}_{n \times mal} = A^n.$$

Lukas Bach, lukas.bach@student.kit.edu

Kreuzprodukt: Beispiele

Karlsruher Institut für Technologie

Kreuzprodukt von zwei Mengen

Zu zwei Mengen A und B ist das Kreuzprodukt $A \times B$ definiert als Menge aller Paare (a, b) mit $a \in A$ und $b \in B$.

$A := \{a, b\}, B := \{1, 2\}$. $A \times B = \{(a, 1), (a, 2), (b, 1), (b, 2)\}$.

Lukas Bach, lukas.bach@student.kit.edu

Kreuzprodukt: Beispiele

Karlsruher Institut für Technologie

Kreuzprodukt von n Mengen

Zu n Mengen $M_1, M_2, \ldots, M_n$ ist das Kreuzprodukt $M_1 \times M_2 \times \cdots \times M_n$ definiert als Menge aller n-Tupel $(e_1, e_2, \ldots, e_n)$ mit $e_1 \in M_1$, $e_2 \in M_2, \ldots,$ $e_n \in M_n$.

$A := \{a, b\}$, $B := \{1, 2\}$, $C := \{\omega\}$. $A \times B \times C$
$= \{(a, 1, \omega), (a, 2, \omega), (b, 1, \omega), (b, 2, \omega)\}$.

Lukas Bach, lukas.bach@student.kit.edu

Kreuzprodukt: Beispiele

Karlsruher Institut für Technologie

Mengenpotenz

$$\underbrace{A \times A \times \cdots \times A}_{nmal} = A^n.$$

- $A := \{a, b\}$. $A^2 = \{(a, b), (b, a), (a, a), (b, b)\}$
  $A^3 = \{(a, a, a), (a, a, b), (a, b, b), \dots\}$.
- A beliebige Menge. $A^0? = \emptyset$
- Achtung! $2^M \neq M^2$. Potenzmengen nicht mit Mengenpotenz verwechseln!

Lukas Bach, lukas.bach@student.kit.edu

Relation

Karlsruher Institut für Technologie

Binäre Relation

Eine binäre Relation auf zwei Mengen A und B ist eine Menge $R \subseteq A \times B$.

- Für die Mengen $M_{Spiele} = \{``Battlefield'', ``AgeOfEmpires'', ``SeriousSam''\}$, $M_{Genre} = \{``Shooter'', ``Strategie''\}$ sind folgendes mögliche Relationen:
  - $\{(``AgeOfEmpires'', ``Strategie''), (``Battlefield'', ``Shooter''), (``SeriousSam'', ```Shooter''\}$
  - $\{(``AgeOfEmpires'', ``Strategie''), (``AgeOfEmpires'', ``Shooter'')$
  - $\emptyset$
- "Kleinergleichrelation" auf $M = \{1, 2, 3\}$: $R_{\leq} = \{(1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3)\} \in M \times M$

Lukas Bach, lukas.bach@student.kit.edu

Relation

Karlsruher Institut für Technologie

Binäre Relation

Eine binäre Relation auf zwei Mengen A und B ist eine Menge $R \subseteq A \times B$.

Ternäre Relation

Eine ternäre Relation auf drei Mengen A, B und C ist eine Menge $R \subseteq A \times B \times C$.

n-äre Relation

Eine n-äre Relation auf n Mengen M_1, M_2 ... M_n ist eine Menge $R \subseteq M_1 \times M_2 \times \cdots \times M_n$.

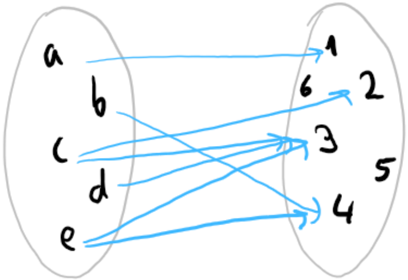
Lukas Bach, lukas.bach@student.kit.edu

Linkstotalität

Linkstotale Relation

Eine Relation $R \subseteq A \times B$ heißt linkstotal, wenn für jedes $a \in A$ ein $b \in B$ existiert mit $(a, b) \in R$.

Die linke Seite der Relation ist also “total” aufgefüllt.



Lukas Bach, lukas.bach@student.kit.edu

Rechtstotalität

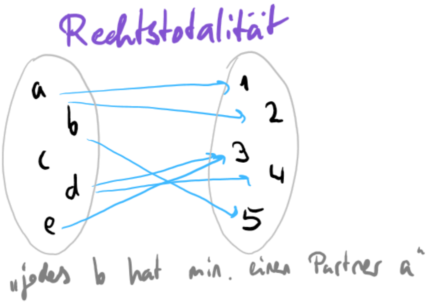
Karlsruher Institut für Technologie

Rechtstotale Relation

Eine Relation $R \subseteq A \times B$ heißt rechtstotal, wenn für jedes $b \in B$ ein $a \in A$ existiert mit $(a, b) \in R$.

Die rechte Seite der Relation ist also “total” aufgefüllt.

Wenn die Relation zusätzlich eine Abbildung ist, heißt diese dann surjektiv.



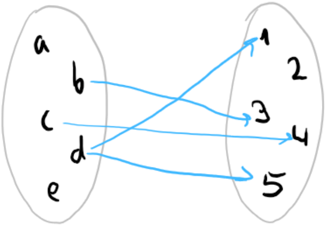
Lukas Bach, lukas.bach@student.kit.edu

Linkseindeutigkeit

Linkseindeutige Relation

Eine Relation $R \subseteq A \times B$ heißt linkseindeutig, wenn für zwei beliebige Elemente $(a, \alpha) \in R, (b, \beta) \in R$ aus der Relation R gilt: wenn $a \neq b$, dann gilt auch $\alpha \neq \beta$.

Also: Keine zwei Elemente der linken Seite der Relation haben dasselbe rechte Element.
Angenommen, $a \neq b$ und $\alpha = \beta$. $\Rightarrow$ offenbar nicht linkseindeutig.
Wenn die Relation zusätzlich eine Abbildung ist, heißt diese dann injektiv.



Lukas Bach, lukas.bach@student.kit.edu

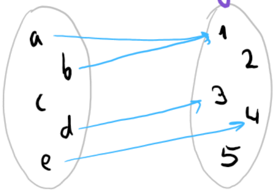
Rechtseindeutig

Rechtseindeutige Relation

Eine Relation $R \subseteq A \times B$ heißt rechtseindeutig, wenn für zwei beliebige Elemente $(a, \alpha) \in R, (b, \beta) \in R$ aus der Relation R gilt: wenn $\alpha \neq \beta$, dann gilt auch $a \neq b$.

Also: Keine zwei Elemente der rechten Seite der Relation haben dasselbe linke Element.

Rechtseindeutigkeit



„jedes a hat max einen Partner b"

Lukas Bach, lukas.bach@student.kit.edu

Abbildung

Abbildung

Eine Relation R heißt eine Abbildung, wenn sie linkstotal *und* rechtseindeutig sind.

- Injektive Funktion: linkstotal, rechtseindeutig, linkseindeutig
- Surjektive Funktion: linkstotal, rechtseindeutig, rechtstotal

Bijektivität

Eine Relation heißt bijektiv, wenn sie injektiv und surjektiv ist.

Damit ist sie linkstotal und rechtseindeutig (weil es eine Abbildung ist) und linkseindeutig (injektiv) und rechtstotal (surjektiv).
Tolle Eigenschaft: Für jedes Element $(a, b) \in R$ der bijektiven Relation R ist *jedem a genau ein b* zugeordnet.

Lukas Bach, lukas.bach@student.kit.edu

Abbildungen Schreibweise

Seien $A = B = \mathbb{R}$, $f \subseteq A \times B$. Wir suchen Relation, die für jedes $a \in A$ ein Element $(a, b) \in f$ enthält mit $b = a^2$.
$f = \{(0, 0), (0.1, 0.01), (2, 4), \dots\}$
Unendlich viele Elemente, und unmöglich alle zu nennen.
(Mathematischere) Schreibweise für Abbildungen:
$f : A \to B, a \mapsto a^2$, also Quadratfunktion.
Ist diese Funktion injektiv oder surjektiv?

- Nicht injektiv, da z.B. $f(1) = f(-1)$, also $(1, 1) \in f$ und $(-1, 1) \in f$.
- Nicht surjektiv, da z.B. -1 nie als Funktionswert angenommen wird, daher $(a, -1) \notin f$ für beliebige $a \in A$.

Lukas Bach, lu-
kas.bach@student.kit.edu

Tutorium vom 4.11.2016

Lukas Bach, lu-
kas.bach@student.kit.edu

Wiederholung

Lukas Bach, lukas.bach@student.kit.edu

Wiederholung

$A := \{a, b, c\}, B := \{b, c, d\}, C := \{a, d\}$

- $A \cap B = \{b, c\}$
- $A \cup B = \{a, b, c, d\}$
- $A \backslash B = \{a\}$
- $C^2 = C \times C = \{(a, a), (a, d), (d, a), (d, d)\}$
- $2^C = \{\emptyset, \{a, d\}, \{a\}, \{d\}\}$
- Unterschied zwischen $\{a, b\}$ und (a, b)?
- Definition von...
  - Alphabet?
  - Abbildung?

Lukas Bach, lukas.bach@student.kit.edu

Wörter

Lukas Bach, lukas.bach@student.kit.edu

Wörter

Karlsruher Institut für Technologie

Konkatenation

Durch Konkatenation werden einzelne Buchstaben aus einem Alphabet miteinander verbunden.

- Symbol: $\cdot$, also zwei Buchstaben a und b miteinander konkateniert: $a \cdot b$.
- Nicht kommutativ: $a \cdot b \neq b \cdot a$
- Aber assoziativ: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- Kurzschreibweise: Ohne Punkte, also $a \cdot b = ab$

Lukas Bach, lukas.bach@student.kit.edu

Wörter

Wörter: Intuitivere Definition

Ein Wort w entsteht durch die Konkatenation durch Buchstaben aus einem Alphabet.

Also Abfolge von Zeichen.
Sei $A := \{a, b, c\}$.

- Mögliche Worte: $w_1 := a \cdot b$, $w_2 = b \cdot c \cdot c$, $w_3 = a \cdot c \cdot c \cdot b \cdot a$.
- Keine möglichen Worte: d.
- Konkatenation nicht kommutativ: Wort *abc* ist ungleich dem Wort *bca*.

Lukas Bach, lukas.bach@student.kit.edu

Wörter

Karlsruher Institut für Technologie

Wörter: Abstraktere Definition

Ein Wort w über dem Alphabet A ist definiert als surjektive Abbildung $w : \mathbb{Z}_n \to A$. Dabei heißt n die Länge $|w|$ des Wortes.

- $\mathbb{Z}_n = \{i \in \mathbb{N} : 0 \leq i < n\}$
  $\mathbb{Z}_3 = \{0, 1, 2\}, \mathbb{Z}_2 = \{0, 1\}, \mathbb{Z}_0 = \emptyset$.
- Länge oder Kardinalität eines Wortes: $|w|$. $|abcde| = 5$.
- Wort $w = abdec$ als Relation aufgeschrieben:
  $w = \{(0, a), (1, b), (2, d), (3, e), (4, c)\}$. Also
  $w(0) = a, w(1) = b, w(2) = d, \dots$
  Damit sieht man auch: $|w| = |\{(0, a), (1, b), (2, d), (3, e), (4, c)\}| = 5$.

Lukas Bach, lukas.bach@student.kit.edu

Wörter

Karlsruher Institut für Technologie

- Wort der Kardinalität 0?

Das leere Wort

Das leere Wort ε ist definiert ein Wort mit Kardinalität 0, also mit 0 Zeichen.

- Leere Wort wird interpretiert als “nicht sichtbar” und kann überall platziert werden: $aabc = a\varepsilon abc = \varepsilon\varepsilon a\varepsilon bc\varepsilon$.
- $|\{\varepsilon\}| = 1$, die Menge ist nicht leer! Das leere Wort ist nicht *nichts*! (Vergleiche leere Menge)
- $|\varepsilon| = 0$.

Lukas Bach, lukas.bach@student.kit.edu

Mehr über Wörter

Karlsruher Institut für Technologie

A^n

Zu einem Alphabet A ist A^n definiert als die Menge aller Wörter der Länge n über dem Alphabet A.

- Nicht mit Mengenpotenz verwechseln!
- $A := \{a, b, c\}$, $A^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$.
  $A^1 = A$, $A^0 = \{\varepsilon\}$.

Die Menge aller Wörter *beliebiger* Länge:

- $A^* := \bigcup_{i \in \mathbb{N}_0} A_i$
- $A := \{a, b, c\}$. $aa \in A^*$, $abcabcabc \in A^*$, $aaaa \in A^*$, $\varepsilon \in A^*$.

Lukas Bach, lukas.bach@student.kit.edu

Mehr über Wörter

Konkatenation von Wörtern:

- $lager \cdot regal = lagerregal$
- $lag \cdot erregal = lagerregal$

Konkatenation von Wörtern.

$$w_1 \cdot w_2 : \mathbb{Z}_{m+n} \to A_1 \cup A_2$$
$$i \mapsto \begin{cases} w_1(i) & \text{falls } 0 \leq i < m \\ w_2(i-m) & \text{falls } m \leq i < m+n \end{cases}$$

- Warum $\mathbb{Z}_{m+n}$? Wörter w_1 und w_2 mit $|w_1| = m$ und $|w_2| = n$ werden konkateniert, also neues Wort hat Länge $m + n$.

Lukas Bach, lukas.bach@student.kit.edu

Mehr über Wörter

Konkatenation von Wörtern.

$$w_1 \cdot w_2 : \mathbb{Z}_{m+n} \to A_1 \cup A_2$$

$$i \mapsto \begin{cases} w_1(i) & \text{falls } 0 \leq i < m \\ w_2(i-m) & \text{falls } m \leq i < m+n \end{cases}$$

$|w_1| = 5 := m$ $\quad$ $|w_2| = 8 := n$

$w_1(1)$ $w_1(2)$ $w_1(3)$ $w_1(4)$ $w_1(5)$ $\quad$ $w_2(1)$ $w_2(2)$ $w_2(3)$ $w_2(4)$ $w_2(5)$ $w_2(6)$ $w_2(7)$ $w_2(8)$

grund · begriffe

Wert von i: 0 1 2 3 4 $\quad$ 5 6 7 8 9 10 11 12

Fall 1: $0 \leq i < m = 5$

Fall 2: $m \leq i < m+n = 13$

Lukas Bach, lukas.bach@student.kit.edu

Mehr über Wörter

Karlsruher Institut für Technologie

- Immernoch: Reihenfolge ist wichtig!
  $OTT \cdot O = OTTO \neq OOTT = O \cdot OTT$
- Auf wieviele Weisen kann man *abc* als Konkatenation nichtleerer Wörter schreiben? abc, $a \cdot bc$, $ab \cdot c$, $a \cdot b \cdot c$.
- Wortkonkatenation mit dem leeren Wort: $w \cdot \varepsilon = w = \varepsilon \cdot w$.

Lukas Bach, lukas.bach@student.kit.edu

Mehr über Wörter

Wort Potenzen

Sich direkt wiederholende Teilworte kann man als Wortpotenz darstellen, daher $w_i^n = w_i \cdot w_i \cdots w_i$ (n $\times$ mal).

- $a^4 = aaaa$, $b^3 = bbb$, $c^0 = \varepsilon$, $d^1 = d$.
- $a^3 c^2 b^6 = aaaccbbbbbb$.
- $b \cdot a \cdot (n \cdot a)^2 = banana$.
- $(a^3 b^2)^2 c (a^2 bcb^3)^3 dd = (aaabb)^2 c (aabcbbb)^3 dd$
  $= aaabb \cdot aaabb \cdot \quad c \quad \cdot aabcbbb \cdot aabcbbb \cdot aabcbbb \cdot \quad dd.$

Lukas Bach, lukas.bach@student.kit.edu

Übung zu Wörter

Sei A ein Alphabet.

Übung zu Wörter

1. Finde Abbildung $f : A^* \to A^*$, sodass für alle $w \in A^*$ gilt:
   $2 \cdot |w| = |f(w)|$.
2. Finde Abbildung $g : A^* \to A^*$, sodass für alle $w \in A^*$ gilt:
   $|w| + 1 = |g(w)|$.
3. Finde Abbildung $h : A^* \to A^*$, sodass für alle $w \in A^*$ gilt:
   $\lfloor \frac{|w|}{2} \rfloor = |h(w)|$. (Zusatz)
4. Sind f, g, h injektiv und/oder surjektiv?

1. $f : A^* \to A^*, w \mapsto w \cdot w$.
2. $g : A^* \to A^*, w \mapsto w \cdot x, x \in A$.
3. $h : A^* \to A^*, w \mapsto \hat{w}$ mit $\hat{w}_i = \begin{cases} w_i & \text{wenn } i \leq \lfloor \frac{|w|}{2} \rfloor \\ \varepsilon & \text{sonst} \end{cases}$ und $i \in \mathbb{Z}_{|w|}$.

Übung zu Wörter

Lukas Bach, lukas.bach@student.kit.edu

1. $f : A^* \to A^*, w \mapsto w \cdot w$.
   - f ist injektiv, denn jedes w aus der Bildmenge wird von maximal einem Wort abgebildet.
   - f ist nicht surjektiv, denn z.B. bildet nichts auf $x \in A$ ab (oder auf andere Wörter mit ungerader Anzahl an Buchstaben).
2. $g : A^* \to A^*, w \mapsto w \cdot x, x \in A$.
   - g ist injektiv.
   - g ist nicht surjektiv, denn z.B. bildet nichts auf ε ab.
3. $h : A^* \to A^*, w \mapsto \hat{w}$ mit $\hat{w}_i = \begin{cases} w_i & \text{wenn } i \leq \lfloor \frac{|w|}{2} \rfloor \\ \varepsilon & \text{sonst} \end{cases}$ und $i \in \mathbb{Z}_{|w|}$.
   - h ist nicht injektiv, denn z.B. $x = h(xy) = h(xz)$ mit $x, y, z \in A$.
   - h ist surjektiv, denn für jedes $w \in A^*$ existiert ein $\hat{w} \in A^*$ mit $\hat{w} = w \cdot w$ sodass $h(\hat{w}) = w$.

Lukas Bach, lukas.bach@student.kit.edu

Formale Sprachen

Lukas Bach, lukas.bach@student.kit.edu

Formale Sprache

- Was war nochmal A^*? Menge aller Wörter *beliebiger* Länge über Alphabet A.

Formale Sprache

Eine Formale Sprache L über einem Alphabet A ist eine Teilmenge $L \subseteq A^*$.

- Zufälliges Beispiel: $A := \{b, n, a\}$.
  - $L_1 := \{ban, baan, nba, aa\}$ ist eine mögliche formale Sprache über A.
  - $L_2 := \{banana, bananana, banananana, ...\}$ $= \{w : w = bana(na)^k, k \in \mathbb{N}\}$ auch.
  - $L_3 := \{ban, baan, baaan, ...\}$ auch. Andere Schreibweise? $L_3 = \{w : w = ba^k n, k \in \mathbb{N}\}$
- Formale Sprachen sind also nicht zwangsweise endliche Mengen.
- Praktischeres Beispiel: $A := \{w : w \text{ ist ein ASCII Symbol }\}$.
  - $L_4 := \{class, if, else, while, for, ...\}$ ist eine formale Sprache über A.
  - $L_5 := \{w : w = a \cdot b \text{ mit } a \text{ als Großbuchstabe und } b \text{ als Groß- oder Kleinbuchstabe }\} \setminus L_4$ ist eine formale Sprache von korrekten Klassennamen in Java.

Lukas Bach, lukas.bach@student.kit.edu

Übung zu formalen Sprachen

$A := \{a, b\}$

- Sprache L aller Wörter über A, die nicht das Teilwort ab enthalten?
  - Was passiert wenn ein solches Wort ein a enthält? Dann keine b's mehr!
  - $L = \{w_1 \cdot w_2 : w_1 \in \{b\}^* \text{ und } w_2 \in \{a\}^*\}$
  - Andere Möglichkeit: Suche Wörter mit ab und nehme diese Weg.
  - $L = \{a, b\}^* \backslash \{w_1 \cdot ab \cdot w_2 : w_1, w_2 \in \{a, b\}^*\}$

Lukas Bach, lukas.bach@student.kit.edu

Übung zu formalen Sprachen

Karlsruher Institut für Technologie

Sei $A := \{a, b\}$, $B := \{0, 1\}$.

Aufgabe zu formalen Sprachen

1. Sprache $L_1 \subseteq A^*$ von Wörtern, die mindestens drei b's enthalten.
2. Sprache $L_2 \subseteq A^*$ von Wörtern, die gerade Zahl von a's enthält.
3. Sprache $L_3 \subseteq B^*$ von Wörtern, die, interpretiert als Binärzahl eine gerade Zahl sind.

1. $L_1 = \{w = w_1 b w_2 b w_3 b w_4 : w_1, w_2, w_3, w_4 \in A^*\}$
2. $L_2 = \{w = (w_1 a w_2 a w_3)^* : w_1, w_2, w_3 \in \{b\}^*\}$ (Ist da ε drin?)
3. $L_3 = \{w = w \cdot 0 : w \in B^*\}$

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 11.11.2016

Lukas Bach, lukas.bach@student.kit.edu

Aussagenlogik

Aussagenlogik

Lukas Bach, lukas.bach@student.kit.edu

- Das wars erst mal zu formalen Sprachen.
- Heute ist Freitag.
- Die Menge aller Männer dieser Welt ist disjunkt zur Menge aller Frauen dieser Welt.

Das sind alles Aussagen. Aussagen sind entweder *wahr* oder *falsch*.

Lukas Bach, lukas.bach@student.kit.edu

Aussagenlogik

Wir kapseln Aussagen und verwendet Variablen dafür.
Zum Beispiel:

- $A :=$ “Die Straße ist nass.”
- $B :=$ “Es regnet.”

Aussagen lassen sich verknüpfen:

- Logisches Und: $A \wedge B = A$ und $B =$ Die Straße ist nass und es regnet.
- Logisches Oder: $A \vee B = A$ oder $B =$ Die Straße ist nass oder es regnet. Es kann auch beides wahr sein.
- Negierung: $\neg A =$ nicht $A =$ Die Straße ist nicht nass.
- Implikation: $A \rightarrow B =$ Aus A folgt $B =$ Wenn die Straße nass ist, dann regnet es.
- Äquivalenz: $A \leftrightarrow B = A$ und B sind äquivalent $=$ Die Straße ist *genau dann* nass, *wenn* es regnet.
  - $A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$, also die Straße ist nass wenn es regnet *und* es regnet wenn die Straße nass ist.

Übung zu Aussagenlogik

Lukas Bach, lukas.bach@student.kit.edu

- $A :=$ “Die Straße ist nass.”
- $B :=$ “Es regnet.”
- $C :=$ “π ist gleich 3.”

- Was ist $B \rightarrow C$? “Wenn es regnet, ist π gleich 3.”

x_1	x_2	$\neg x_1$	$x_1 \wedge x_2$	$x_1 \vee x_2$	$x_1 \rightarrow x_2$
f	f	w	f	f	w
f	w	w	f	w	w
w	f	f	f	w	f
w	w	f	w	w	w

Lukas Bach, lukas.bach@student.kit.edu

Syntax der Aussagenlogik

Karlsruher Institut für Technologie

Menge der Aussagevariablen:

$Var_{AL} \subseteq \{P_i : i \in \mathbb{N}_0\}$ oder $\{P, Q, R, S, \dots\}$

Alphabet der Aussagenlogik:

$A_{AL} = \{(,), \neg, \wedge, \vee, \rightarrow, \leftrightarrow\} \cup Var_{AL}$

Lukas Bach, lukas.bach@student.kit.edu

Boolesche Funktionen

Karlsruher Institut für Technologie

Boolesche Funktionen

Eine boolsche Funktion ist eine Abbildung der Form $f : \mathbb{B}^n \to \mathbb{B}$ mit $\mathbb{B} = \{w, f\}$.

Typische Boolsche Funktionen: $b_\neg(x) = \neg x$, $b_\vee(x_1, x_2) = x_1 \vee x_2 \ldots$

Lukas Bach, lukas.bach@student.kit.edu

Interpretationen

Karlsruher Institut für Technologie

Interpretation

Eine Interpretation ist eine Abbildung $I : V \rightarrow \mathbb{B}$, die einer Variablenmenge eine "Interpretation", also wahr oder falsch zuordnet.

Weiter legt man $val_I(F)$ als Auswertung einer aussagenlogischer Formel F fest.

$$\begin{aligned} val_I(X) &= I(X) \\ val_I(\neg G) &= b_\neg(val_I(G)) \\ val_I(G \wedge H) &= b_\wedge(val_I(G), val_I(H)) \\ val_I(G \vee H) &= b_\vee(val_I(G) val_I(H)) \\ val_I(G \rightarrow H) &= b_\rightarrow(val_I(G), val_I(H)) \end{aligned}$$

Lukas Bach, lukas.bach@student.kit.edu

Übung zu Interpretationen

- Wie viele Interpretationen gibt es bei $k = 1, 2, 3$ Variablen?
- Wie viele Interpretationen gibt es bei $k+1$ Variablen im Vergleich zu k Variablen?

Lukas Bach, lukas.bach@student.kit.edu

Übung zur Aussagenlogik

Karlsruher Institut für Technologie

Sei $A := w, B := w, C := f$.

- Ist $(A \wedge B) \vee \neg C$ wahr oder falsch?
  $(A \wedge B) \vee \neg C = (w \wedge w) \vee \neg f = w \vee \neg f = w \vee w = w$, die Aussage ist also wahr.
- Ist $\neg(A \vee A)$ wahr oder falsch? Falsch! Wann ist $\neg(A \vee A)$ im allgemeinen wahr? Genau dann, wenn $\neg A$ wahr ist.

Aussagen Äquivalenz

Erinnerung: $A \leftrightarrow B$ heißt: $A \rightarrow B \wedge B \rightarrow A$.
Wenn zwei Aussagen äquivalent sind, sind ihre Wahrheitswerte immer gleich, wenn die Wahrheitswerte, von denen sie abhängen, gleich sind.
Mann sagt und schreibt dann: A ist *genau dann* wahr, *wenn* B wahr ist.

- $\neg(A \vee A)$ ist genau dann wahr, wenn $\neg A$ wahr ist, also gilt:
  $\neg(A \vee A) \leftrightarrow \neg A$.

Lukas Bach, lukas.bach@student.kit.edu

Mehr zu Äquivalenz

Alternative Definition zu Äquivalenz

Zwei Formeln G und H heißen äquivalent, wenn für jede Interpretation gilt $val_I(G) = val_I(H)$.

Vorher Äquivalenz von Formeln unter gegebener Interpretation, diesmal Äquivalenz von Formeln unter beliebiger Interpretation.

Bemerkung

- Man schreibt $G \equiv H$
- $\mathbb{B}^V \to \mathbb{B} : I \mapsto val_I(G)$

Beispiele

$(\neg(\neg P))$ ist äquivalent zu P

$(\neg(P \wedge Q))$ ist äquivalent zu $((\neg P) \vee (\neg Q))$

Beispiele zu Äquivalenz

Lukas Bach, lukas.bach@student.kit.edu

- Ein Wort w hat die Länge $n \leftrightarrow |w| = n$.
- Die Vereinigung zweier Mengen A und B hat die Kardinalität $|A| + |B|$ $\leftrightarrow A \cap B = \emptyset \leftrightarrow A$ und B sind disjunkt.
- p ist eine rationale Zahl $\leftrightarrow p$ lässt sich darstellen als $p = \frac{a}{b}, a \in \mathbb{Z}, b \in \mathbb{N} \leftrightarrow p \in \mathbb{Q}$.

Wahrheitstabellen

Lukas Bach, lukas.bach@student.kit.edu

- $(((P \rightarrow Q) \vee Q) \rightarrow (P \wedge \neg Q))$

P	Q	$(P \wedge Q)$	$\vee Q$	$\rightarrow$	$(P \wedge \neg Q)$
w	w	w	w	f	f
w	f	f	f	w	w
f	w	f	w	f	f
f	f	f	f	w	f

Übungen zu Aussagenlogik

Lukas Bach, lukas.bach@student.kit.edu

Übungen zu Aussagenlogik

- Schreibe Wahrheitstabellen zu den Formeln um den Wahrheitsgehalt festzustellen.
- $\neg(P \wedge Q) \wedge \neg(Q \wedge P)$
- $(P \wedge Q \wedge R) \leftrightarrow (\neg P \vee Q)$
- $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$
- Welche dieser Aussagen sind wahr?
- $\neg(P \wedge Q) = \neg P \vee \neg Q$
- $P \wedge P = P \vee P$
- $(P \vee Q) \wedge R = (P \wedge R) \vee (Q \wedge R)$

Wahrheitstabellen

Lukas Bach, lukas.bach@student.kit.edu

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
w	w	f	w	w	w	w
w	f	f	f	w	f	f
f	w	w	f	w	w	f
f	f	w	f	f	w	w

Aufgabe

Finde einen logischen Ausdruck in A und B unter Verwendung von $\wedge$, $\vee$ und $\neg$, der die Aussage “Entweder A oder B” repräsentiert

Lukas Bach, lukas.bach@student.kit.edu

Wahrheitstabellen

Karlsruher Institut für Technologie

Aufgabe

Finde einen logischen Ausdruck in A und B unter Verwendung von $\wedge$, $\vee$ und $\neg$, der die Aussage "Entweder A oder B" repräsentiert

Lösung

A	B	$A \wedge \neg B$	$\neg A \wedge B$	$(A \wedge \neg B) \vee (\neg A \wedge B)$
w	w	f	f	f
w	f	w	f	w
f	w	f	w	w
f	f	f	f	f

Lukas Bach, lukas.bach@student.kit.edu

Weitere Begriffe

Tautologie

Die Formel G ist eine Tautologie (oder allgemeingültig), wenn G für alle Interpretationen wahr ist.

Erfüllbarkeit

Eine Formel G ist erfüllbar, wenn sie für mindestens eine Interpretation wahr ist.

Lemma

Wenn $G \equiv H$ ist, dann ist $G \leftrightarrow H$ eine Tautologie.

Lukas Bach, lukas.bach@student.kit.edu

Übung zu Tautologien

Sind das Tautologien?

- $(G \longrightarrow (H \longrightarrow K)) \longrightarrow ((G \longrightarrow H) \longrightarrow (G \longrightarrow K))$ Ja
- $(\neg P \longrightarrow Q) \wedge R \vee P$ Nein
- $G \longrightarrow (H \longrightarrow G)$ Ja
- $(\neg P \longrightarrow \neg Q) \longrightarrow ((\neg P \longrightarrow Q) \longrightarrow P)$ Ja

Lukas Bach, lukas.bach@student.kit.edu

Übung zu Erfüllbarkeit

Karlsruher Institut für Technologie

Sind die folgenden Ausdrücke erfüllbar?

- $\neg(A \vee \neg A)$ nein
- $(P \wedge \neg Q) \vee (\neg P \wedge R)$ Ja

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 17.11.2016

Lukas Bach, lukas.bach@student.kit.edu

Quiz

- Was macht die Funktion val_I?
- Was bedeutet Äquivalenz?
- Was bedeutet Tautologie und Erfüllbarkeit?
- Welche dieser Aussagen sind erfüllbar?
  - $\neg(P \wedge Q) \leftrightarrow \neg P \vee \neg Q$
  - $P \wedge P \leftrightarrow P \vee P$

Lukas Bach, lu-
kas.bach@student.kit.edu

Vollständige Induktion

Lukas Bach, lukas.bach@student.kit.edu

Wahrheitsgehalt von unendlich Aussagen

Karlsruher Institut für Technologie

Beispielsituation: Wir haben unendlich viele Dominosteine. Behauptung: Alle Dominosteine fallen um.

- Wir haben Aussagen: {“1. Stein fällt um”, “2. Stein fällt um”, ...}
- Wie zeigen wir unendlich viele Aussagen?
- Stelle Aussagen in Abhängigkeit einer Laufvariable n dar:
  - $A(n) :=$ “n-ter Stein fällt um” $\forall n \in \mathbb{N}$.
- Aussage $A :=$ “Alle Steine fallen um” $\equiv A(i)$ ist wahr $\forall i \in \mathbb{N}$.

Wir haben immernoch unendlich viele Aussagen...

- Zeige: $A(1)$ ist wahr, sowie $A(i)$ gilt $\rightarrow A(i+1)$ gilt für beliebiges $i \in \mathbb{N}$.
- Also: Der erste Stein fällt, sowie: falls der i-te Stein fällt, so fällt auch der $i+1$-te Stein.
- Nach dem Prinzip der vollständigen Induktion fallen dann alle Steine um.

Lukas Bach, lukas.bach@student.kit.edu

Vollständige Induktion

- Beweisverfahren
- In der Regel zu zeigen: Eine Aussage gilt für alle $n \in \mathbb{N}_+$, manchmal auch für alle $n \in \mathbb{N}_0$
- Man schließt “induktiv” von einem n auf n+1
- Idee: Wenn die Behauptung für ein beliebiges **festes** n gilt, dann gilt sie auch für den Nachfolger n+1 (und somit auch für dessen Nachfolger und schließlich für alle n)

Lukas Bach, lukas.bach@student.kit.edu

Struktur des Beweises

Behauptung: (*kurz* **Beh.:**)

Beweis: (*kurz* **Bew.:**)

- Induktionsanfang: (*kurz* **IA:**)
  - Zeigen, dass Behauptung für Anfangswert gilt (oft $n = 1$)
  - Auch mehrere (z.B. zwei) Anfangswerte möglich
- Induktionsvoraussetzung: (*kurz* **IV:**)
  - Sei $n \in \mathbb{N}_+$ (bzw. $n \in \mathbb{N}_0$) fest aber beliebig und es gelte [Behauptung einsetzen]
- Induktionsschritt: (*kurz* **IS:**)
  - Behauptung für n+1 auf n zurückführen
  - Wenn induktive Definition gegeben: verwenden!
  - Sonst: Versuche Ausdruck, in dem (n+1) vorkommt umzuformen in einen Ausdruck, in dem nur n vorkommt

Vorhin:

$$\underbrace{A(1) \text{ ist wahr}}_{IA}, \text{ sowie } \underbrace{A(i) \text{ gilt}}_{IV} \rightarrow \underbrace{A(i+1) \text{ gilt}}_{IS} \text{ für beliebiges i } \in \mathbb{N}$$

Lukas Bach, lukas.bach@student.kit.edu

Übung zu Vollständiger Induktion

Karlsruher Institut für Technologie

Aufgabe

$$x_0 := 0$$
$$\text{Für alle } n \in \mathbb{N}_0 : x_{n+1} := x_n + 2n + 1$$

Zeige mithilfe vollständiger Induktion, dass für alle $n \in \mathbb{N}_0$

$$x_n = n^2$$

gilt.

Lukas Bach, lukas.bach@student.kit.edu

Übung zu vollständiger Induktion

Karlsruher Institut für Technologie

Übungsaufgaben

Zeige die Wahrheit folgender Aussagen mit vollständiger Induktion:

- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \; \forall n \in \mathbb{N}.$
- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \; \forall n \in \mathbb{N}$

Lukas Bach, lukas.bach@student.kit.edu

Formale Sprache

Lukas Bach, lukas.bach@student.kit.edu

Formale Sprache

- Was war nochmal A^*? Menge aller Wörter *beliebiger* Länge über Alphabet A.
- Was war nochmal eine formale Sprache?

Formale Sprache

Eine Formale Sprache L über einem Alphabet A ist eine Teilmenge $L \subseteq A^*$.

Als Beispiel von vorigen Folien:

- $A := \{b, n, a\}$.
  - $L_1 := \{ban, baan, nba, aa\}$ ist eine mögliche formale Sprache über A.
  - $L_2 := \{banana, bananana, banananana, ...\}$
    $= \{w : w = bana(na)^k, k \in \mathbb{N}\}$ auch.
  - $L_3 := \{ban, baan, baaan, ...\}$ auch. Andere Schreibweise?
    $L_3 = \{w : w = ba^k n, k \in \mathbb{N}\}$

Lukas Bach, lukas.bach@student.kit.edu

Produkt von Sprachen

Produkt von formalen Sprachen

Von zwei formalen Sprachen L_1, L_2 lässt sich das Produkt $L_1 \cdot L_2$ bilden mit $L_1 \cdot L_2 = \{w_1 w_2 : w_1 \in L_1 \text{ und } w_2 \in L_2\}$.

Sei $A := \{a, b\}$, $B := \{\alpha, \beta, \gamma, \varepsilon, \delta\}$.

- Sprache $L_1 \subseteq A^*$, die zuerst drei a's enthält und dann entweder zwei b's oder vier a's? $L_1 = \{aaa\} \cdot \{bb, aaaa\}$.
- Sprache $L_2 \subseteq A^*$, die alle Wörter über A enthält außer ε? $L_2 = A \cdot A^* = A^* \setminus \{\varepsilon\}$.
- Sprache $L_3 \subseteq B^*$, die alle Wörter über B enthält, mit:
  - Zwei beliebigen Zweichen aus B.
  - Dann einem ε oder zwei δ's.
  - Dann vier Zeichen aus A.
- $L_3 = B \cdot B \cdot \{\varepsilon, \delta\delta\} \cdot A \cdot A \cdot A \cdot A$.

Lukas Bach, lukas.bach@student.kit.edu

Produkt von Sprachen

Karlsruher Institut für Technologie

Übung zu Produkt von formalen Sprachen

Sei A ein beliebiges Alphabet und $M := \{L : L \text{ ist formale Sprache über } A\} = 2^A$. Produkt von Sprachen lässt sich auch als Abbildung bzw. Verknüpfung $\cdot : M \times M \to M$ darstellen.
Zeige:

- Die Verknüpfung $\cdot$ ist assoziativ.
- Es gibt (mindestens) ein Element $e \in M$, sodass für alle $x \in M$ gilt: $x \cdot e = e \cdot x = x$. (Neutrales Element)
- Es gibt ein Element $o \in M$, sodass für alle $x \in M$ gilt: $x \cdot o = o = o \cdot x$. (Absorbierendes Element)

Lukas Bach, lukas.bach@student.kit.edu

Produkt von Sprachen

Seien $L_1, L_2, L_3 \in M$.

- Die Verknüpfung $\cdot$ ist assoziativ:
  - $(L_1 \cdot L_2) \cdot L_3 = (\{w_1 \cdot w_2 : w_1 \in L_1, w_2 \in L_2\}) \cdot L_3 = \{w_1 w_2 w_3 : w_1 \in L_1, w_2 \in L_2, w_3 \in L_3\} = L_1 \cdot (\{w_2 w_3 : w_2 \in L_2, w_3 \in L_3\}) = L_1 \cdot (L_2 \cdot L_3)$.
- Es gibt (mindestens) ein Element $e \in M$, sodass für alle $x \in M$ gilt: $x \cdot e = e \cdot x = x$. (neutrales Element)
  - $e := \{\varepsilon\}$.
  - $L_1 \cdot \{\varepsilon\} = L_1 = \{\varepsilon\} \cdot L_1$
- Es gibt ein Element $o \in M$, sodass für alle $x \in M$ gilt: $x \cdot o = o = o \cdot x$. (Absorbierendes Element)
  - $o := \emptyset$
  - $L_1 \cdot \emptyset = \emptyset = \emptyset \cdot L_1$

$(M, \cdot)$ ist damit trotzdem keine Gruppe, denn es existieren keine Invers-Element.

Lukas Bach, lukas.bach@student.kit.edu

Potenz von Sprachen

Potenz von Sprachen

Potenz von formellen Sprachen ist wie folgt definiert:

- $L^0 := \{\varepsilon\}$
- $L^{i+1} := L^i \cdot L$ für $i \in \mathbb{N}_0$.

- $L_1 := \{a\}$.
  - $L_1^0 = \{\varepsilon\}$. $L_1^1 = \{\varepsilon\} \cdot L_1 = L_1$.
  - $L_1^2 = (\{\varepsilon\} \cdot L_1) \cdot L_1 = \{aa\}$.
- $L_2 := \{ab\}^3\{c\}^4$
  - $L_2^0 = \{\varepsilon\}, L_2^1 = ...$
  - $L_2^2 = (\{ab\}^3\{c\}^4)^2 = (\{ab\}^3\{cccc\})^2 = \{abababcccc\}^2 = \{abababccccabababcccc\}$.
- $L_3 := (\{a\} \cup \{b\})^2 = \{aa, ab, ba, bb\}$

Lukas Bach, lukas.bach@student.kit.edu

Konkatenationsabschluss bei formalen Sprachen

Karlsruher Institut für Technologie

Konkatenationsabschluss

Zu einer formalen Sprache L ist der Konkatenationsabschluss L^* definiert als $L^* := \bigcup_{i \in \mathbb{N}_0} L^i$.

ε-freie Konkatenationsabschluss

Zu einer formalen Sprache L ist der ε-freie Konkatenationsabschluss L^+ definiert als $L^+ := \bigcup_{i \in \mathbb{N}_+} L^i$.

- Warum gilt $\varepsilon \notin L^+$ bei formalen Sprache $L \subseteq A^* \backslash \{\varepsilon\}$?
- $L := \{a, b, c\}. L^* = \{\varepsilon, a, aa, ab, ac, aaa, aab, \ldots, b, ba, bb, \ldots\}$
- $L := \{aa, bc\}. L^* = \{\varepsilon, aa, bc, aa \cdot aa, aa \cdot bc, bc \cdot aa, bc \cdot bc, aa \cdot aa \cdot aa, \ldots\}$

Lukas Bach, lukas.bach@student.kit.edu

Übung zu Konkatenationsabschluss

Sei $A := \{a, b\}$, $B := \{A, B, C, D, E, F\}$.

- Sprache $L_1 \subseteq A^*$, die das Teilwort *ab* nicht enthält? $L_1 = \{b\}^*\{a\}^*$.
- Sprache $L_2 \subseteq B^*$, die alle erlaubten Java Variablennamen enthält.
  - $B := \{\_, a, b, ..., z, A, B, ..., Z\}$
  - $C := B \cup \mathbb{Z}_9$
  - $L_2 \subseteq C = (B \cdot C^*)\backslash\{if, class, while, ...\}$

Lukas Bach, lukas.bach@student.kit.edu

Übung zu Konkatenationsabschluss

Sei $L := \{a\}^*\{b\}^*$.

- Was ist alles in L drin?
  - $aaabbabbaaabba$? Nein.
  - $aaabb$, $abbaaabba$? Ja, nein.
  - $aaabb$, abb, $aaabba$? Ja, ja, nein.
  - $aaabb$, abb, $aaabb$, a? Alles drin.
- Was ist alles in L^* drin?
  - $aaabbabbaaabba$? Ja.
  - $aaabb$, $abbaaabba$? Ja.
  - $aaabb$, abb, $aaabba$? Ja.
  - $aaabb$, abb, $aaabb$, a? Ja.
  - Alle Wörter aus $\{a, b\}^*$! $\rightarrow L^* = \{a, b\}^*$.

Lukas Bach, lukas.bach@student.kit.edu

Übung zu Konkatenationsabschluss

Erinnerung

$$L^* := \bigcup_{i\in\mathbb{N}_0} L^i \qquad L^+ := \bigcup_{i\in\mathbb{N}_+} L^i$$

Beweisaufgabe

Beweise: $L^* \cdot L = L^+$.

$\subseteq$:

Voraussetzung: $w \in L^* \cdot L$ mit $w = w'w''$, $w' \in L^*$ und $w'' \in L$.

Dann existiert ein $i \in \mathbb{N}_0$ mit $w' \in L^i$, also $w = w'w'' \in L^i \cdot L = L^{i+1}$.

Weil $i + 1 \in \mathbb{N}_+$, gilt: $L^{i+1} \subseteq L^+$, also $w \in L^+$.

$\supseteq$:

Voraussetzung: $w \in L^* \cdot L$.

Dann existiert ein $i \in \mathbb{N}_+$ mit $w \in L^i$. Da $i \in \mathbb{N}_+$, existiert ein $j \in \mathbb{N}_0$ mit $i = j + 1$, also für ein solches $j \in \mathbb{N}_0$: $w \in L^{j+1} = L^j \cdot L$.

Also $w = w'w''$ mit $w' \in L^j$ und $w'' \in L$.

Wegen $L^j \subseteq L^*$ ist $w = w'w'' \in L^* \cdot L$.

Lukas Bach, lukas.bach@student.kit.edu

Übung zu formalen Sprachen

L_1, L_2 seien formale Sprachen.

- Wie sieht $L_1 \cdot L_2$ aus?
- Wie sieht L_1^3 aus?
- Wie sieht $L_1^2 \cdot L_2 \cdot L_2^0 \cdot L_1^*$ aus?
- Wie sieht $(L_1^*)^0 \cdot L_2^+$ aus?

Lukas Bach, lu-
kas.bach@student.kit.edu

Übersetzung und Kodierung

Herführung zu Zahlendarstellungen

Lukas Bach, lukas.bach@student.kit.edu

Wir betrachten die Alphabete $A_{dez} := \mathbb{Z}_{10}, A_{bin} := \{0, 1\}, A_{oct} := \mathbb{Z}_8$.

- Was können wir daraus machen?
- $A^*_{dez} \supset \{42, 1337, 999\}$.
- $A^*_{bin} \supset \{101010, 10100111001, 1111100111\}$.
- $A^*_{oct} \supset \{52, 2471, 1747\}$.
- Wir suchen eine Möglichkeit, diese Zahlen zu deuten.
- Aber irgendwie so, dass $42_{\in A_{dez}} \overset{Deutung}{=} 101010_{\in A_{bin}} \overset{Deutung}{=} 52_{\in A_{oct}}$.

Lukas Bach, lukas.bach@student.kit.edu

Definition von Zahlendarstellungen

Karlsruher Institut für Technologie

Num_k

Einer Zeichenkette Z_k aus Ziffern wird mit Num_k eine eindeutige Zahl zugeordnet:

$$Num_k(\varepsilon) = 0$$

$Num_k(wx) = k \cdot Num_k(w) + num_k(x)$ mit $w \in Z_k^*$ und $x \in Z_k$.

num_k

Einer einzelnen Ziffer $x \in Z_k$ aus einem Alphabet von Ziffern Z_k wird mit $num_k(x)$ der Wert der Zahl zugewiesen.

- Wichtig: $Num_k(w) \neq num_k(w)$!
- Was ist: $num_{10}(3) = 3$, $num_{10}(7) = 7$, $num_{10}(11) =$ nicht definiert.
- Für Zahlen $\geq k$: Benutze Num_k!

Lukas Bach, lukas.bach@student.kit.edu

Beispiel zu Zahlendarstellungen

Karlsruher Institut für Technologie

$Num_k(\varepsilon) = 0$.
$Num_k(wx) = k \cdot Num_k(w) + num_k(x)$ mit $w \in Z_k^*$ und $x \in Z_k$.

Was ist $Num_{10}(123)$?

- $Num_{10}(123) = 10 \cdot Num_{10}(12) + num_{10}(3) =$
  $10 \cdot (Num_{10}(1) + num_{10}(2)) + num_{10}(3) =$
  $10 \cdot (num_{10}(1) + 10 \cdot num_{10}(2)) + num_{10}(3) = 10 \cdot (1 + 10 \cdot 2) + 3 = 123$.

Yay?
Was ist der dezimale Zahlenwert der Binärzahl 1010? Diesmal Basis $k = 2$.

- $Num_2(1010) = 2 \cdot Num_2(101) + num_2(0) =$
  $2 \cdot (2 \cdot Num_2(10) + num_2(1) + num_2(0) =$
  $2 \cdot (2 \cdot (2 \cdot Num_2(1) + num_2(0)) + num_2(1)) + num_2(0) =$
  $2 \cdot (2 \cdot (2 \cdot num_2(1) + num_2(0)) + num_2(1)) + num_2(0) =$
  $2 \cdot (2 \cdot (2 \cdot 1 + 0) + 1) + 0) = 10$.

Yay!

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben zu Zahlendarstellungen

$Num_k(\varepsilon) = 0.$
$Num_k(wx) = k \cdot Num_k(w) + num_k(x)$ mit $w \in Z_k^*$ und $x \in Z_k$.

Übungen zu Zahlendarstellungen

Berechne den numerischen Wert der folgenden Zahlen anderer Zahlensysteme nach dem vorgestellten Schema:

- $Num_8(345)$.
- $Num_2(11001)$.
- $Num_2(1000)$.
- $Num_4(123)$.
- $Num_{16}(4DF)$. (Zusatz)

Anmerkung: Hexadezimalzahlen sind zur Basis 16 und verwenden als Ziffern (in aufsteigender Reihenfolge: $1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$.

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben zu Zahlendarstellungen

Lösungen:

- $Num_8(345) = 229$.
- $Num_2(11001) = 25$.
- $Num_2(1000) = 8$.
- $Num_4(123) = 27$.
- $Num_{16}(4DF) = 1247$.

Lukas Bach, lukas.bach@student.kit.edu

Einfachere Umrechnung von Zahlendarstellungen

Es gilt:
$2(2(2(2(2 \cdot 1+0)+1)+0)+1)+0 = 2^4 \cdot 1+2^4 \cdot 0+2^3 \cdot 1+2^2 \cdot 0+2^1 \cdot 1+2^0 \cdot 0.$
Daher, einfachere Rechenweise:
$Num_k(w) = k^0 \cdot w(0) + k^1 \cdot w(1) + k^2 \cdot w(2) + ...$
Was sind folgende Zahlen in Dezimal im Kopf gerechnet?

- $Num_2(10101) = 21.$
- $Num_2(11101) = 29.$
- $Num_2(1111111111) = 1023.$

Lukas Bach, lukas.bach@student.kit.edu

Einfachere Umrechnung von Zahlendarstellungen

$Num_k(w) = k^0 \cdot w(0) + k^1 \cdot w(1) + k^2 \cdot w(2) + ...$
Was sind folgende Zahlen in Dezimal im Kopf gerechnet?

- $Num_{16}(A1) = 161.$
- $Num_{16}(BC) = 188.$
- $Num_{16}(14) = 20.$

Lukas Bach, lukas.bach@student.kit.edu

Rechnen mit *div* und *mod*

Karlsruher Institut für Technologie

div Funktion

Die Funktion *div* dividiert ganzzahlig. (Schneidet also den Rest ab).

mod Funktion

Die Modulo Funktion *mod* gibt den Rest einer ganzzahligen Division zurück.

- 22 div 8 $= 2$ $\left(\frac{22}{8} = 2,75\right)$.
- 22 mod 8 $= 6$.

Fülle die Tabelle aus:

x	0	1	2	3	4	5	6	7	8	9	10	11	12
x *div* 4	0	0	0	0	1	1	1	1	2	2	2	2	3
x *mod* 4	0	1	2	3	0	1	2	3	0	1	2	3	0

Lukas Bach, lukas.bach@student.kit.edu

Von Zeichen zu Zahlen zurück zu Zahlen

11101_2 ist also 29_{10}. Was ist 29_{10} in binär?

k-äre Darstellung

Die Repräsentation einer Zahl n zur Basis k lässt sich wie folgt ermitteln:

$$\mathbf{Repr}_k(n) = \begin{cases} \mathbf{repr}_k(n) & \text{falls } n < k \\ \mathbf{Repr}_k(n \operatorname{div} k) \cdot \mathbf{repr}_k(n \bmod k) & \text{falls } n \geq k \end{cases}$$

Achtung! Das $\cdot$ Symbol steht für Konkatenation, nicht für Multiplikation!

Lukas Bach, lukas.bach@student.kit.edu

Beispiel zu $Repr_k$

Karlsruher Institut für Technologie

$$\mathbf{Repr}_k(n) = \begin{cases} \mathbf{repr}_k(n) & \text{falls } n < k \\ \mathbf{Repr}_k(n \operatorname{div} k) \cdot \mathbf{repr}_k(n \bmod k) & \text{falls } n \geq k \end{cases}$$

Zum Beispiel:

$$\begin{aligned}
\mathbf{Repr}_2(29) &= \mathbf{Repr}_2(29 \mathbin{\mathbf{div}} 2) \cdot \mathbf{repr}_2(29 \mathbin{\mathbf{mod}} 2) \\
&= \mathbf{Repr}_2(14) \cdot \mathbf{repr}_2(1) \\
&= \mathbf{Repr}_2(14 \mathbin{\mathbf{div}} 2) \cdot \mathbf{repr}_2(14 \mathbin{\mathbf{mod}} 2) \cdot 1 \\
&= \mathbf{Repr}_2(7) \cdot \mathbf{repr}_2(0) \cdot 1 \\
&= \mathbf{Repr}_2(7 \mathbin{\mathbf{div}} 2) \cdot \mathbf{repr}_2(7 \mathbin{\mathbf{mod}} 2) \cdot 01 \\
&= \mathbf{Repr}_2(3) \cdot \mathbf{repr}(1) \cdot 01 \\
&= \mathbf{Repr}_2(3 \mathbin{\mathbf{div}} 2) \cdot \mathbf{repr}(3 \mathbin{\mathbf{mod}} 2) \cdot 101 \\
&= \mathbf{Repr}_2(1) \cdot \mathbf{repr}(1) \cdot 101 \\
&= 11101
\end{aligned}$$

Lukas Bach, lukas.bach@student.kit.edu

Beispiel zu $Repr_k$

$$\mathbf{Repr}_k(n) = \begin{cases} \mathbf{repr}_k(n) & \text{falls } n < k \\ \mathbf{Repr}_k(n \text{ div } k) \cdot \mathbf{repr}_k(n \text{ mod } k) & \text{falls } n \geq k \end{cases}$$

Beispiel mit Hexadezimalzahlen:

$$\begin{aligned} \mathbf{Repr}_{16}(29) &= \mathbf{Repr}_{16}(29 \textbf{ div } 16) \cdot \mathbf{repr}_{16}(29 \textbf{ mod } 16) \\ &= \mathbf{Repr}_{16}(1) \cdot \mathbf{repr}_{16}(13) \\ &= 1 \cdot D = 1D \end{aligned}$$

Lukas Bach, lukas.bach@student.kit.edu

Übung zu $Repr_k$

Karlsruher Institut für Technologie

$$\mathbf{Repr}_k(n) = \begin{cases} \mathbf{repr}_k(n) & \text{falls } n < k \\ \mathbf{Repr}_k(n \text{ div } k) \cdot \mathbf{repr}_k(n \text{ mod } k) & \text{falls } n \geq k \end{cases}$$

Übung zu $Repr_k$

Berechne die Repräsentationen folgender Zahlen in gegebenen Zahlensystemen:

- $\mathbf{Repr}_2(13)$.
- $\mathbf{Repr}_4(15)$.
- $\mathbf{Repr}_{16}(268)$.

Lösungen:

- $\mathbf{Repr}_2(13) = 1101$.
- $\mathbf{Repr}_4(15) = 33$.
- $\mathbf{Repr}_{16}(268) = 10C$.

Lukas Bach, lukas.bach@student.kit.edu

Feste Länge von Binärzahlen

bin$_\ell$

Die Funktion $\mathbf{bin}_\ell : \mathbb{Z}_{2^\ell} \to \{0, 1\}^\ell$ bringt eine gegebene Binärzahl auf eine feste Länge, indem sie mit Nullen vorne aufgefüllt wird. Formell wird sie definiert als:

$$\mathbf{bin}_\ell(n) = 0^{\ell - |\mathbf{Repr}_2(n)|}\mathbf{Repr}_2(n)$$

Beispiel:

- $\mathbf{bin}_8(3) = 0^{8-|\mathbf{Repr}_2(3)|}\mathbf{Repr}_2(3) = 0^{8-|11|} \cdot 11 = 0^{8-2} \cdot 11 = 0^6 \cdot 11 = 00000011$.
- $\mathbf{bin}_{16}(3) = 0000000000000011$.

Lukas Bach, lukas.bach@student.kit.edu

Zweierkomplement

Was ist mit negative Zahlen?

- Idee: Verwende das erste Bit, um zu speichern, ob die Zahl positiv oder negativ ist.
- Beispiel: $5 = 0101_{zkpl}$, $-5 = 1101_{zkpl}$.

Zweierkomplement Darstellung

Die Zweierkomplementdarstellung einer Zahl x mit der Länge ℓ ist wie folgt definiert:

$$\mathbf{Zkpl}_\ell(x) = \begin{cases} 0\mathbf{bin}_{\ell-1}(x) & \text{falls } x \geq 0 \\ 1\mathbf{bin}_{\ell-1}(2^{\ell-1} + x) & \text{falls } x < 0 \end{cases}$$

- Wieso $\ell - 1$?

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben zu Zweierkomplement-Darstellung

$$\mathbf{Zkpl}_\ell(x) = \begin{cases} 0\mathbf{bin}_{\ell-1}(x) & \text{falls } x \geq 0 \\ 1\mathbf{bin}_{\ell-1}(2^{\ell-1} + x) & \text{falls } x < 0 \end{cases}$$

Was sind folgende Zahlen in Zweierkomplement-Darstellung?

- $\mathbf{Zkpl}_4(3) = 0011$.
- $\mathbf{Zkpl}_4(7) = 0111$.
- $\mathbf{Zkpl}_4(-5) = 1101$.
- $\mathbf{Zkpl}_8(13) = 00001101$.
- $\mathbf{Zkpl}_8(-34) = 10100010$.
- $\mathbf{Zkpl}_8(-9) = 10001001$.

Lukas Bach, lu-
kas.bach@student.kit.edu

Tutorium vom 24.11.2016

Lukas Bach, lukas.bach@student.kit.edu

Übersetzungen

Lukas Bach, lukas.bach@student.kit.edu

Übersetzungen

Karlsruher Institut für Technologie

Definition der Semantikabbildung

Sei *Sem* die Menge der Bedeutungen. Ferner seien A und B Alphabete und $L_A \subseteq A^*$ und $L_B \subseteq B^*$.
Weiter sei $sem_A : L_A \longrightarrow Sem$ und $sem_B : L_B \longrightarrow Sem$
Dann heißt $f : L_A \longrightarrow L_B$ Übersetzung , wenn gilt: für jedes $w \in L_A$ gilt $sem_A(w) = sem_B(f(w))$.

- Bedeutungserhaltende Abbildungen von Wörtern auf Wörter

Beispiel

Betrachte $Trans_{2,16} : \mathbb{Z}_{16}^* \longrightarrow \mathbb{Z}_2^*$ mit $Trans_{2,16}(w) = Repr_2(Num_{16}(w))$

- $Trans_{2,16}(A3) = Repr_2(Num_{16}(A3)) = Repr_2(163) = 10100011$

Lukas Bach, lukas.bach@student.kit.edu

Wozu Übersetzungen

- Lesbarkeit (vergleiche DF_{16} mit 11011111_2)
- Verschlüsselung
- Kompression (Informationen platzsparend aufschreiben)
- Kontextabhängige Semantiken (Deutsch $\longrightarrow$ Englisch)
- Fehlererkennung

Lukas Bach, lukas.bach@student.kit.edu

Codierungen

Karlsruher Institut für Technologie

Definitionen

- Codewort $f(w)$ einer Codierung $f : L_A \to L_B$
- Code: $\{f(w)|w \in L_A\} = f(L_A)$
- Codierung: **Injektive** Übersetzung
  - Ich komme immer eindeutig von einem Codewort f(w) zu w zurück

Bemerkung

- Was ist, wenn L_A unendlich ist (man kann nicht alle Möglichkeiten aufzählen)
- Auswege: Homomorphismen, Block-Codierungen

Lukas Bach, lukas.bach@student.kit.edu

Homomorphismen

Karlsruher Institut für Technologie

Definition von Homomorphismen

Seien A, B Alphabete. Dann ist $h : A^* \longrightarrow B^*$ ein Homomorphismus, falls für alle $w_1, w_2 \in A^*$ gilt:

$$h(w_1 w_2) = h(w_1) h(w_2)$$

- Ein Homomorphismus ist Abbildung, die mit Konkatenation verträglich ist
- Homomorphismus ist ε-frei, wenn für jedes $x \in A : h(x) \neq \varepsilon$
- Homomorphismen lassen das leere Wort unverändert, also $h(\varepsilon) = \varepsilon$

Lukas Bach, lukas.bach@student.kit.edu

Sei h ein Homomorphismus.

Übung zu Homomorphismen

1. $h(a) = 001$ und $h(b) = 1101$. Was ist dann $h(bba)$?

$\rightarrow$ $h(bba) = h(b)h(b)h(a) = 1101 \cdot 1101 \cdot 001 = 11011101001$

2. Sei $h(a) = 01$, $h(b) = 11$ und $h(c) = \varepsilon$. Nun sei $h(w) = 011101$. Was war w?

$\rightarrow$ *aba* oder *cabccac*, ... Allgemein: $w \in \{c\}^* \cdot \{a\} \cdot \{c\}^* \cdot \{b\} \cdot \{c\}^* \cdot \{a\} \cdot \{c\}^*$ ε-Freiheit hat also die Eindeutigkeit zerstört!

3. Kann h aus 2 eine Codierung sein?

$\rightarrow$ Nein, da nicht injektiv!

4. Warum will man ε-freie Homomorphismen?

$\rightarrow$ Information geht sonst verloren!

5. Was heißt hier Ïnformation geht verloren"?

$\rightarrow$ Es gibt $w_1 \neq w_2$ mit $h(w_1) = h(w_2)$

Lukas Bach, lukas.bach@student.kit.edu

- Information kann auch anders "verloren"gehen

$\rightarrow$ z.B. $h(a) = 0, h(b) = 1, h(c) = 10$ – Wie das?

Präfixfreiheit

Gegeben ist ein Homomorphismus $h : A^* \rightarrow B^*$.
Wenn für keine zwei verschiedenen $x_1, x_2 \in A$ gilt, dass $h(x_1)$ Präfix von $h(x_2)$ ist, dann ist h präfixfrei.

Satz

Präfixfreie Codes sind injektiv.

Beispiele

- $h(a) = 01$ und $h(b) = 1101$ ist präfixfrei
- $g(a) = 01$ und $g(b) = 011$ ist nicht präfixfrei

Lukas Bach, lukas.bach@student.kit.edu

Huffman-Codierung

- Komprimiert eine Zeichenkette
- Kodiert häufiger vorkommende Zeichen zu kürzeren Codewörter als Zeichen die seltener vorkommen.
- Vorgehensweise:
  1. Zähle Häufigkeiten aller Zeichen der Zeichenkette
  2. Schreibe alle vorkommenden Zeichen und ihre Häufigkeiten nebeneinander
  3. Wiederhole, bis der Baum fertig ist:
     - Verbinde die zwei Zeichen mit niedrigsten Häufigkeiten zu neuem Knoten über diesen
     - Dieser hat als Zahl die aufsummierte Häufigkeiten
  4. Danach: Alle linken Kanten werden mit 0 kodiert, alle rechten Kanten mit 1

Das Ergebnis ist eine Zeichenkette aus $\{0, 1\}$, die kürzer ist als die ursprüngliche Zeichenkette in binär.

Lukas Bach, lukas.bach@student.kit.edu

Huffman-Codierung

Gegeben

- $w \in A^*$

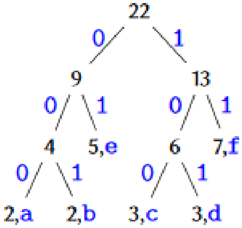
  w = `afebfecaffdeddccefbeff`
- Anzahl der Vorkommen aller Zeichen in w ($N_x(w)$)

Häufigkeiten:

x	a	b	c	d	e	f
$N_x(w)$	2	2	3	3	5	7

Zwei Phasen zur Bestimmung eines Huffman-Codes

1. Konstruieren eines "Baumes"
   - Blätter entsprechen den Zeichen
   - Kanten mit 0 und 1 beschriften



Häufigkeiten:

x	a	b	c	d	e	f
$N_x(w)$	2	2	3	3	5	7

Übung zu Huffman Codierung

Lukas Bach, lukas.bach@student.kit.edu

Übung

Sei $A = \{a, b, c, d, e, f, g, h\}$

- Codiere das Wort `badcfehg` mit Hilfe der Huffman-Codierung

$\rightarrow$ Mögliche Lösung: 001 100 010 011 101 000 111 110

- Wie lauten die Codewörter, wenn für das Wort w gilt: $N_a(w) = 1, N_b(w) = 2, N_c(w) = 2, N_d(w) = 8, N_e(w) = 16, N_f(w) = 32, N_g(w) = 64, N_h(w) = 128$

Mögliche Lösung:

x	a	b	c	d	e	f	g	h
h(x)	0000000	0000001	000001	00001	0001	001	01	1

Lukas Bach, lukas.bach@student.kit.edu

- Wie lang wäre das zweite Wort ($\texttt{abbcccc}\ \texttt{d}^8 \ldots \texttt{g}^{64}\texttt{h}^{128}$) mit dem ersten Code codiert?

$\longrightarrow$ 741 Symbole. Also dreimal so lang wie das Original.

- Wie lang wäre das zweite Wort mit dem zweiten Code codiert?

$\longrightarrow$ 501 Symbole. Also nur zweimal so lang wie das Original.

- Was fällt euch auf?

Lukas Bach, lukas.bach@student.kit.edu

Wahr oder falsch?

Karlsruher Institut für Technologie

Sei $h : A^* \to \mathbb{Z}_2$ eine Huffman-Codierung

- h ist ein ε-freier Homomorphismus **Wahr!**
- Häufigere Symbole werden mit langen Worten codiert, seltene mit kürzeren **Falsch!**
- Die Kompression ist am stärksten, wenn die Häufigkeiten aller Zeichen ungefähr gleich sind. **Falsch!**
- h ist präfixfrei **Wahr!**
- Es kann noch kürzere Codierungen geben **Falsch!**

Lukas Bach, lukas.bach@student.kit.edu

Huffman-Codierung

Eigenschaften

Sei A ein Alphabet und $w \in A$. Dann gilt für die Huffman-Codierung h:

- $h : A^* \to \mathbb{Z}_2$
- h ist ε-freier Homomorphismus
- h ist präfixfreier Homomorphismus
- Häufigere Symbole werden mit kurzen Worten codiert, seltene mit längeren
- Produziert kürzestmögliche Codierungen

Lukas Bach, lukas.bach@student.kit.edu

Block-Codierung mit Huffman

- Wir betrachten nicht mehr einzelne Symbole, sondern Blöcke von fester Länge $b > 1$
- Blätter des Huffman-Baums sind jetzt *Wörter der Länge b*

Beispiel an der Tafel: Codierung von
$aab \cdot deg \cdot deg \cdot aab \cdot ole \cdot aab \cdot deg \cdot aab$.

- Alphabet $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
- Text über A, der nur aus Teilwörtern der Länge 10 zusammengesetzt ist, in denen jeweils immer nur ein Symbol vorkommt
- Angenommen $\mathtt{a}^{10}$, ..., $\mathtt{d}^{10}$ kommen alle gleich häufig vor. Wie lang ist dann die Huffman-Codierung?

$\rightarrow$ Ein Fünftel, weil jeder Zehnerblock durch zwei Bits codiert wird

Lukas Bach, lu-
kas.bach@student.kit.edu

Speicher

Lukas Bach, lukas.bach@student.kit.edu

Speicher

- Ein **Bit** ist Zeichen aus $A = \{0, 1\}$
- Ein **Byte** ist ein Wort aus acht Bits
- Abkürzungen
  - Für Bit: `bit`
  - Für Byte: `B`

Präfixe

Lukas Bach, lukas.bach@student.kit.edu

Dezimal

10^{-3}	10^{-6}	10^{-9}	10^{-12}	10^{-15}	10^{-18}
1000^{-1}	1000^{-2}	1000^{-3}	1000^{-4}	1000^{-5}	1000^{-6}
milli	mikro	nano	pico	femto	atto
m	μ	n	p	f	a
10^{3}	10^{6}	10^{9}	10^{12}	10^{15}	10^{18}
1000^{1}	1000^{2}	1000^{3}	1000^{4}	1000^{5}	1000^{6}
kilo	mega	giga	tera	peta	exa
k	M	G	T	P	E

Binär

2^{10}	2^{20}	2^{30}	2^{40}	2^{50}	2^{60}
1024^{1}	1024^{2}	1024^{3}	1024^{4}	1024^{5}	1024^{6}
kibi	mebi	gibi	tebi	pebi	exbi
Ki	Mi	Gi	Ti	Pi	Ei

Lukas Bach, lukas.bach@student.kit.edu

Gesamtzustand eines Speichers

Zu jedem Zeitpunkt ist

- für jede Adresse festgelegt, welcher Wert dort ist
- beides meist Bitfolgen

Vorstellung: Tabelle mit zwei Spalten

Adresse	**Wert**
Adresse 1	Wert 1
Adresse 2	Wert 2
Adresse 3	Wert 3
...	...
Adresse n	Wert n

Lukas Bach, lukas.bach@student.kit.edu

Zustand eines Speichers – formal

Definition des Speicherzustandes

Sei *Adr* die Menge aller Adressen und *Val* die Menge aller Werte.
Dann ist

$$m : Adr \longrightarrow Val$$

der aktuelle Zustand des Speichers. Dabei ist $m(a)$ der aktuelle Wert an der Adresse a.

Lukas Bach, lukas.bach@student.kit.edu

Lesen und Speichern

Mem

Menge aller möglichen Speicherzustände, also Menge aller Abbildungen von *Adr* nach *Val*

$$Mem := Val^{Adr}$$

Anmerkung: Für zwei Mengen A, B gilt: $A^B := \{f : B \to A\}$.

memread

$$memread : Mem \times Adr \to Val \text{ mit } (m, a) \mapsto m(a)$$

memwrite

$memwrite : Mem \times Adr \times Val \to Mem$ mit $(m, a, v) \mapsto m'$

Für m' wird folgendes gefordert:

$$m(a') := \begin{cases} v & \text{falls } a' = a \\ m(a') & \text{falls } a' \neq a \end{cases}$$

Lukas Bach, lukas.bach@student.kit.edu

Eigenschaften von *memread* und *memwrite*

Karlsruher Institut für Technologie

Eigenschaften (“Invarianten”)

- $memread(memwrite(m, a, v), a) = v$ (Also: An a einen Wert v zu schreiben und danach bei a zu lesen gibt den Wert v zurück $\Rightarrow$ Konsistente Datenhaltung)
- $memread(memwrite(m, a', v'), a) = memread(m, a)$ (Also: Auslesen einer Speicherstelle ist unabhängig davon, was vorher an eine andere Adresse geschrieben wurde $\Rightarrow$ Unabhängige Datenhaltung)

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben

Aktueller Speicherzustand:

Adresse	Wert
00000	01110
00001	00100
00010	00111
00011	00000
...	...

Was ist?

- $memread(memwrite(m, memread(m, 00011), 01010), 00000)$

$\longrightarrow$ 01010

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 1.12.2016

Lukas Bach, lukas.bach@student.kit.edu

Zum Übungsblatt

Lukas Bach, lukas.bach@student.kit.edu

Anmerkungen zum letzten Übungsblatt

- Was ist sind die folgenden Mengen?
  - $\mathbb{N}$ = Menge der natürlichen Zahlen (1, 2, 3, ...)
  - $\mathbb{N}_0$ = $\mathbb{N} \cup \{0\}$
  - $\mathbb{R}$ = Menge der Reellen Zahlen
  - $\mathbb{R}^+$ = Menge der positiven reellen Zahlen
  - $\mathbb{R}_0$ gibt es nicht! 0 ist auch so schon in $\mathbb{R}$
  - $\mathbb{R}_0^+$ genauso nicht!
- Aufgabe: $R : A^* \to A^*$
  - $R(\varepsilon) = \varepsilon$
  - $\forall x \in A : R(x) = x$
  - $\forall w \in A^* \forall x \in A \forall y \in A : R(xwy) = yR(w)x$
  - Zeige: $\forall n \in \mathbb{N}_0 : \forall w \in A^n : |R(w)| = |w|$

Lukas Bach, lu-
kas.bach@student.kit.edu

MIMA

Lukas Bach, lukas.bach@student.kit.edu

Was ist die MIMA?

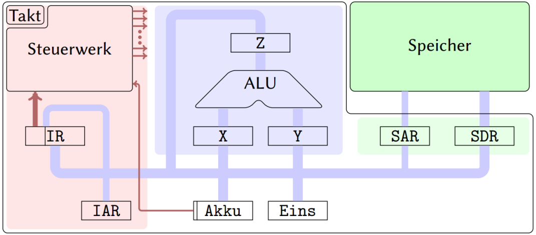
- Theoretischer, idealisierter Prozessor
- Funktioniert wie ein echter Prozessor, ist aber simpler
- Nah an Technischer Informatik

Grundaufbau:

- Adressen als 20*bit* Datenwort
- Speicherworte als 24*bit* Datenwort
- Maschinenbefehle als...
  - 4*bit* Befehl und 20*bit* Adresse
  - oder 8*bit* Befehl und unwichtigem Rest

Lukas Bach, lukas.bach@student.kit.edu

Aufbau der MIMA: Steuerwerk

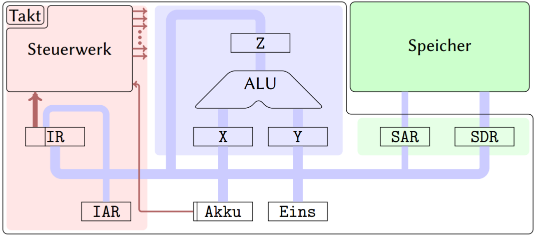


Steuerwerk

- Instruction Register (IR) enthält den nächsten auszuführenden Befehl
- Instruction Adress Register (IAR) enthält die Adresse des nächsten Befehls
- Takt bestimmt die “Tickrate”, also die Geschwindigkeit
- Steuerwerk interpretiert alle Befehle und führt sie aus
- Welche Befehle es gibt: Siehe später

Lukas Bach, lukas.bach@student.kit.edu

Aufbau der MIMA: Akku und Eins

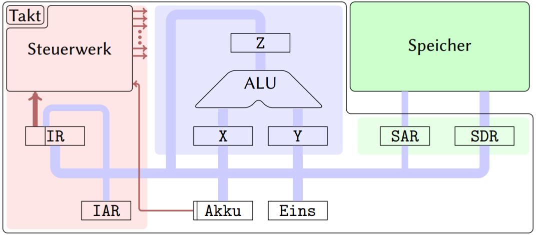


Akku und Eins

- Akku dient als Zwischenspeicher für Datenworte
- Hält maximal ein Wort
- Eins liefert die Konstante 1, hält also Strom
- z.B. erhöhen des IAR

Aufbau der MIMA: ALU

Lukas Bach, lukas.bach@student.kit.edu



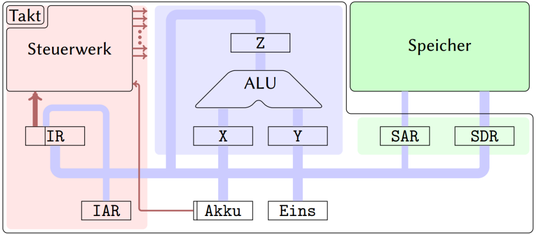
Arithmetic Logic Unit (ALU) / Rechenwerk

- Durchführt arithmetische Operationen
- **mod** , **div** , $+$, $-$, ..., bitweises OR/AND/...

- X und Y sind Eingaberegister
- Z ist Ausgaberegister

Lukas Bach, lukas.bach@student.kit.edu

Aufbau der MIMA: ALU



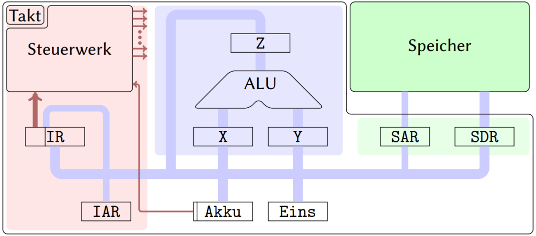
Speicher(werk)

Speicher selbst speichert Befehle und Daten. Speicherwerk besteht aus:

- Speicheradressregister (SAR) ist die Adresse, bei der im Speicher gespeichert/gelesen werden soll
- Speicherdatenregister (SDR) Datum, das bei der Adresse gespeichert werden soll/ gelesen wurde.

Lukas Bach, lukas.bach@student.kit.edu

Aufbau der MIMA: ALU



Busse

- “Kabel” zwischen den Verbindungen
- Ein kompletter Bus überträgt entweder 1, 0, oder nichts
- Kann nur eine einzige Information auf einmal übertragen

Lukas Bach, lukas.bach@student.kit.edu

Konventionen zu MIMA Programmen

Um MIMA Programme und dazugehörige Definitionen verständlicher zu machen, vereinbaren wir folgende Konventionen:

- Befehle (eigentlich Bitfolge) schreiben wir als Befehlname und Adresse
  - $001000000000000000101010 \equiv \mathit{STV}\ 42$
- $X \leftarrow Y \equiv$ "Der Variable X wird der Wert Y zugewiesen"

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

Befehlssyntax	Formel	Bedeutung
LDC const	$Akku \leftarrow const$	Lade eine Konstate *const* in den Akku
LDV adr	$Akku \leftarrow M(adr)$	Lade einen Wert vom Speicher bei Adresse *adr* in den Akku
STV adr	$M(adr) \leftarrow Akku$	Lade Speichere den Wert aus dem Akku im Speicher bei Adresse *adr*
LDIV adr	$Akku \leftarrow M(M(adr))$	Lade einen Wert vom Speicher bei der Adresse, die bei *adr* gespeichert ist, und lade den Wert in den Akku
STIV adr	$M(M(adr)) \leftarrow Akku$	Speichere den Wert im Akku bei der Adresse, die in *adr* gespeichert ist.

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle (2)

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

Befehlssyntax	Formel	Bedeutung
ADD adr	$Akku \leftarrow Akku + M(adr)$	Addiere den Wert bei *adr* zum Akku dazu.
“*OP*” *adr*	$Akku$“OP”$M(adr)$	Wende bitweise Operation auf Akku mit Wert bei *adr* an. $Op \in \{AND, OR, XOR\}$.

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle (3)

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

Befehlssyntax	Bedeutung
NOT	Bitweise Invertierung aller Bits des Akku-Datenwortes
RAR	Rotiere alle Akku-Bits eins nach rechts
EQL adr	Setze Akku auf $11 \cdots 11$, falls Wert bei *adr* gleich Akku-Wert, setze Akku auf $00 \cdots 00$ sonst.
JMP adr	Springe zu Befehlsadresse *adr*
JMN adr	Springe zu Befehlsadresse *adr*, falls Akku negativ (also erstes Bit $= 1$), sonst fahre normal fort.

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle: Sichern und Laden

- Befehle zum laden und Speichern in den Speicher
- LDV um Daten vom Speicher zu laden, STV um Daten in den Speicher zu schreiben
- LDC um eine Konstante zu laden
- Daten werden in einem Zwischenspeicher gelagert, der nur ein Datenwort hält: Akku.

Beispiele:

- *LDV* 9 lädt das Datum, das im Speicher bei Adresse 9 liegt, in den Akku.
- *STV* 9 speichert das Datum, das im Akku liegt, in den Speicher an Adresse 9.
- *LDC* 4 lädt die Zahl 4 in den Akku (also kein Speicherzugriff).

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle: Sichern und Laden

Befehlssyntax	Formel	Bedeutung
LDC const	$Akku \leftarrow const$	Lade eine Konstate *const* in den Akku
LDV adr	$Akku \leftarrow M(adr)$	Lade einen Wert vom Speicher bei Adresse *adr* in den Akku
STV adr	$M(adr) \leftarrow Akku$	Lade Speichere den Wert aus dem Akku im Speicher bei Adresse *adr*

Beispielprogramm mit initialem Speicherabbild

LDC 5
STV a_1
LDC 7
STV a_2
$\vdots$

$\vdots$
LDV a_1
STV a_3
HALT

Adresse	Wert
a_1	0
a_2	0
a_3	0

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle: Indirektes Sichern und Laden

Befehlssyntax	Formel	Bedeutung
LDIV adr	$Akku \leftarrow M(M(adr))$	Lade einen Wert vom Speicher bei der Adresse, die bei *adr* gespeichert ist, und lade den Wert in den Akku
STIV adr	$M(M(adr)) \leftarrow Akku$	Speichere den Wert im Akku bei der Adresse, die in *adr* gespeichert ist.

Beispielprogramm mit initialem Speicherabbild

```
LDIV 4
STV 5
LDIV 5
STIV 4
HALT
```

Adresse	Wert
4	6
5	0
6	7
7	2

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle: Eins plus Eins

- Befehle zu arithmetischen Operationen
- Eine ALU-Operation, angewandt auf dem Wert des Akkus und dem Wert an gegebener Adresse
- Beispiele:
  - *ADD* 4 addiert den Wert im Akku mit dem Wert aus dem Speicher an Adresse 4 und legt das Resultat im Akku ab. Achtung: Addition nicht mit dem Wert 4!
  - *AND* 3 führt bitweise Verundung zwischen dem Wert im Akku und dem Wert aus dem Speicher an Adresse 4 durch und legt das Resultat im Akku ab.

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle: Eins plus Eins

Befehlssyntax	Formel	Bedeutung
ADD adr	$Akku \leftarrow Akku + M(adr)$	Addiere den Wert bei *adr* zum Akku dazu.
“OP” adr	$Akku“OP”M(adr)$	Wende bitweise Operation auf Akku mit Wert bei *adr* an. $Op \in \{AND, OR, XOR\}$.

Beispielprogramm mit initialem Speicherabbild

LDC 5
ADD 3
AND 4
STV 5
LDC 12
XOR 5
HALT

Adresse	Wert
3	3
4	8
5	17

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle: Bits und Bytes

Karlsruher Institut für Technologie

- *NOT* invertiert alle Bits des Datums im Akku. Beispiel *NOT* mit 5 im Akku, angenommen der Akku speichert bis zu 8 bits: $5_{10} = 00000101_2$, nach der Invertierung: 11111010_2.
- *RAR* rotiert alle Bits des Datums im Akku um eine Stelle nach rechts. Beispiel mit 5 im Akku: 00000101_2 wird zu 10000010_2.
- *EQL adr* vergleicht den Wert im Akku mit dem Wert bei *addr*.
  - Setzt Akku $= 11 \cdots 11$ falls Werte gleich sind.
  - Setzt Akku $= 00 \cdots 00$ falls Werte nicht gleich sind.

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle: Bits und Bytes

Befehlssyntax	Bedeutung
NOT	Bitweise Invertierung aller Bits des Akku-Datenwortes
RAR	Rotiere alle Akku-Bits eins nach rechts
EQL adr	Setze Akku auf $11 \cdots 11$, falls Wert bei *adr* gleich Akku-Wert, setze Akku auf $00 \cdots 00$ sonst.

Beispielprogramm mit initialem Speicherabbild

```
LDC 5
NOT
RAR
NOT
RAR
⋮
```

```
⋮
RAR
EQL 15
EQL 0
HALT
```

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle: Springen

- Normalerweise wird die Instruktionsadresse nach jedem Befehl um eins erhöht
- Also Befehle werden von oben nach unten abgearbeitet
- Mit Sprüngen kann man die MIMA zwingen, zu definiertem Befehl zu springen und damit die Vorgehensreihenfolge zu beeinflussen

- *JMP adr* führt als nächsten Befehl den an Adresse *adr* aus.
- *JMN adr* führt als nächsten Befehl den an Adresse *adr* aus, falls der Akku negativ ist.
  - Also wenn das erste Bit im Akku negativ ist.
  - Wenn vorher ein *EQL* erfolgreich verglichen hat, wird also gesprungen.
  - Wenn der Akku positiv ist, werden die Befehle nach *JMN* normal weiter abgearbeitet.

Lukas Bach, lukas.bach@student.kit.edu

MIMA Befehle: Springen

Befehlssyntax	Bedeutung
EQL adr	Setze Akku auf $11\cdots11$, falls Wert bei *adr* gleich Akku-Wert, setze Akku auf $00\cdots00$ sonst.
JMP adr	Springe zu Befehlsadresse *adr*
JMN adr	Springe zu Befehlsadresse *adr*, falls Akku negativ (also erstes Bit $=1$), sonst fahre normal fort.

Beispielprogramm mit initialem Speicherabbild

	LDC 5
a_1 :	JMN a_2
	EQL 1
	JMN a_1
	$\vdots$

	$\vdots$
	NOT
a_2 :	JMP a_3
	NOT
a_3 :	*HALT*

Adresse	Wert
1	5

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben

Karlsruher Institut für Technologie

MIMA-Programm schreiben

Schreibe ein MIMA-Programm:

- Eingabe: Adresse a_1 einer positiven Zahl x.
- Ausgabe: Speichert $3 \cdot x$ in a_1.

Lösung:

LDV a_1
ADD a_1
ADD a_1
STV a_1
HALT

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben

MIMA-Programm schreiben

Schreibe ein MIMA-Programm:

- Eingabe: Adresse a_1 einer positiven Zahl x.
- Ausgabe: Speichert x **mod** 2 in a_1.

Lösung:

LDC 1 // 000000000000000000000001
AND a_1
STV a_1
HALT

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben

MIMA-Programm schreiben

Schreibe ein MIMA-Programm:

- Eingabe: Adresse a_1 einer positiven Zahl x.
- Ausgabe: Speichert x **div** 2 in a_1.

Lösung:

LDC 1
NOT
AND a_1 // Setze "rechtestes" Bit auf 0
RAR
STV a_1
HALT

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 8.12.2016

Lukas Bach, lu-
kas.bach@student.kit.edu

Kontextfreie Grammatiken

Lukas Bach, lukas.bach@student.kit.edu

Kontextfreie Grammatiken

Zur Rekapitulation...

- Was ist ein Alphabet, was eine formale Sprache?
- Was kennen wir für Operationen auf formalen Sprachen?

Betrachte $L := \{a^n b a^n : n \in \mathbb{N}\}$. Wie kann man diese Sprache darstellen?

Lukas Bach, lukas.bach@student.kit.edu

Kontextfreie Grammatiken

Kontextfreie Grammatik

Ein Tupel G = (N, T, S, P) mit

- N Alphabet (Nichtterminalsymbole)
- T Alphabet mit $N \cap T = \emptyset$ (Terminalsymbole)
- $S \in N$ (Startsymbol)
- $P \subseteq N \times (N \cup T)^*$ mit $|P| \in \mathbb{N}_0$

- Was ist $N \times (N \cup T)^*$? Bei $T := \{a, b, c\}, N = \{S, A, B\}$:
  $N \times (N \cup T)^* = \{(S, abSAcB), (A, SSS), (B, BSabc), ...\}$.
- Andere Schreibweise: $P : N \to (N \cup T)^*$.
- Für $(X, w) \in P$ schreibt man $X \to w$
- Statt $\{X \to w_1, X \to w_2\}$ schreibt man auch $\{X \to w_1 | w_2\}$

Lukas Bach, lukas.bach@student.kit.edu

Ableitungsschritt

Erinnerung: N = *Nichtterminalsymbole*, T = *Terminalsymbole*.

Ableitungsschritt

$v \in (N \cup T)^*$ ist in einem Schritt aus $u \in (N \cup T)^*$ ableitbar, wenn

- $u = w_1 X w_2$ und $v = w_1 w_X w_2$ für $w_1, w_2 \in (N \cup T)^*$
- und $X \rightarrow w_X$ in P

Notation

$u \Rightarrow v$

Beispiel

$G := (\{S, B\}, \{a, b\}, S, \{S \rightarrow aBa | aSa, B \rightarrow b\})$

- $S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aaaBaaa \Rightarrow aaabaaa$. Fertig.
- $aaaSaaa \not\Rightarrow aaaabaaaa$! $\Rightarrow$ heißt eine Ableitung!

Lukas Bach, lukas.bach@student.kit.edu

Ableitungsfolge

Ableitungsfolge

Wir definieren $\Rightarrow^i$ für $i \in \mathbb{N}_0$ folgendermaßen:

Für $u, v \in (N \cap T)^*$ gelte:

- $u \Rightarrow^0 v$ genau dann, wenn $u = v$ gilt.
- $u \Rightarrow^{i+1} v$ genau dann, wenn ein $w \in (N \cup T)^*$ existiert, für das $u \Rightarrow w \Rightarrow^i v$ gilt. Für $u \Rightarrow^i v$ sagt man "v ist aus u in i Schritten ableitbar".

Beispiel

$G := (\{S, B\}, \{a, b\}, S, \{S \to aBa | aSa, B \to b\})$

Dann gilt $aaaSaaa \Rightarrow^0 aaaSaaa$

und $aaaSaaa \Rightarrow^2 aaaabaaaa$

Lukas Bach, lukas.bach@student.kit.edu

Ableitbarkeit

Für $u, v \in (N \cup T)^*$ gelte $u \Rightarrow^* v$ genau dann, wenn ein $i \in \mathbb{N}_0$ existiert, mit $u \Rightarrow^i v$. Man sagt dann "v ist aus u ableitbar".

Beispiel

$G := (\{S, B\}, \{a, b\}, S, \{S \rightarrow aBa | aSa, B \rightarrow b\})$
Dann gilt $S \Rightarrow^* aaaSaaa$
und $aSa \Rightarrow^* aaaabaaaa$
aber $aSa \not\Rightarrow abba$.

Lukas Bach, lukas.bach@student.kit.edu

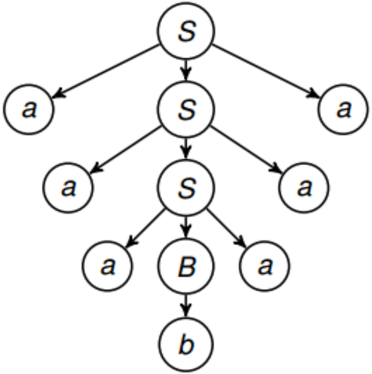
Ableitungsbaum

Beispiel

$G := (\{S, B\}, \{a, b\}, S, \{S \rightarrow aBa | aSa, B \rightarrow b\})$

Dann gilt $S \Rightarrow^* aaabaaa$

- Startsymbol ist Wurzel
- Nichtterminale sind innere Knoten
- Für $X \Rightarrow w$ sind die Zeichen von w die Kinder von X
- Terminale sind die Blätter



Lukas Bach, lukas.bach@student.kit.edu

Übung zu Kontextfreien Grammatiken

Karlsruher Institut für Technologie

Übung

Gegeben ist die Kontextfreie Grammatik (N, T, S, P) mit:

- Nichtterminalsymbolen $N := \{A, B, S\}$.
- Terminalsymbolen $T := \{a, b, c\}$
- Startsymbol S
- Produktionen $P := \{S \rightarrow aaS|bbS|SAS|\varepsilon, A \rightarrow cB, B \rightarrow a, b, c, \varepsilon\}$.

Aufgabe: Welche der folgenden Wörter sind ableitbar? Konstruiere den Ableitungsbaum und zeige, wie sie abgeleitet werden.

- $ccbbcbbbcbbaaaa$?
- $aabbaabbaabb$?
- c?

Lukas Bach, lukas.bach@student.kit.edu

Formale Sprachen erzeugen

Erzeugte Sprache

Sei $G = (N, T, S, P)$ eine kontextfreie Grammatik. Dann nennen wir $L(G) := \{w \in T^* | S \Rightarrow^* w\}$ die von G erzeugte Sprache.

Kontextfreie Sprache

Eine formale Sprache L heißt genau dann kontextfrei, wenn eine kontextfreie Grammatik G existiert, mit $L(G) = L$.

$G := (\{S, B\}, \{a, b\}, S, \{S \rightarrow aBa | aSa, B \rightarrow b\})$

Dann ist $L(G) = \{a^n b a^n | n \in \mathbb{N}_+\}$

Lukas Bach, lukas.bach@student.kit.edu

Karlsruher Institut für Technologie

Verständnisfragen

- $G = (\{X\}, \{a, b\}, X, \{X \to \varepsilon | aX | bX\})$
  - Welche Wörter lassen sich in drei Schritten ableiten?

  $\to$ $\{aa, ab, ba, bb\}$
  - Was ist $L(G)$?

  $\to$ $L(G) = \{a, b\}^*$
- Gibt es auch eine Grammatik G mit $L(G) = \{\}$?

$\to$ $G_1 := (\{X\}, \{a, b\}, X, \{X \to X\})$ oder $G_2 := (\{X\}, \{a, b\}, X, \{\})$

- Wahr oder falsch? Wenn $w_1 \Rightarrow w_2$ gilt, dann gilt auch $w_1 \to w_2$
- Was ist der Unterschied von $\Rightarrow$ und $\Rightarrow^*$?

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben zu kontextfreien Grammatiken

- Sei $L_1 := \{wbaaw'|w, w' \in \{a, b\}^*\}$. Konstruiere eine Grammatik G_1 mit $L(G_1) = L_1$.

$\rightarrow$ $G_1 := (\{X, Y\}, \{a, b\}, X, \{X \rightarrow YbaaY, Y \rightarrow aY|bY|\varepsilon\})$.

- Welche Sprache erzeugt $G_2 = (\{S, X, Y\}, \{a, b\}, S, P_2)$ mit $P_2 = \{S \rightarrow X|Y, X \rightarrow aaXb|aab, Y \rightarrow aYbb|abb\}$?

$\rightarrow$ $L(G_2) = \{a^{2k}b^k|k \in \mathbb{N}_+\} \cup \{a^k b^{2k}|k \in \mathbb{N}_+\}$

Lukas Bach, lukas.bach@student.kit.edu

Beispiel zu kontextfreien Grammatiken

Karlsruher Institut für Technologie

$G = (\{X\}, \{(,)\}, X, \{X \to XX|(X)|\varepsilon\})$

- Welche Wörter sind ableitbar?

$\to$ "wohlgeformte Klammerausdrücke"

- Welche Eigenschaften besitzen diese Wörter?

$\to$ $N_((w) = N_)(w)$ Ist diese Eigenschaft hinreichend?

$\to$ Nein, es muss gelten: Für alle Präfixe v von w gilt $N_((v) \geq N_)(v)$

- Andere Grammatik möglich, die alle wohlgeformten Klammerausdrücke erzeugt?

$\to$ $G = (\{X\}, \{(,)\}, X, \{X \to (X)X|\varepsilon\})$

Lukas Bach, lukas.bach@student.kit.edu

Grenze kontextfreier Grammatiken

Es gibt auch Sprachen, die wir nicht mit einer kontextfreien Grammatik erzeugen können!

Beispiel aus der Vorlesung:
$L_{vv} = \{vcv | v \in \{a, b\}^*\} \subseteq \{a, b, c\}^*$

Lukas Bach, lu-
kas.bach@student.kit.edu

Relationen vol. 2

Lukas Bach, lukas.bach@student.kit.edu

Relationen

Karlsruher Institut für Technologie

Erinnerung Relationen

Es seien A und B Mengen. Eine Teilmenge $R \subseteq A \times B$ heißt Relation.

Lukas Bach, lukas.bach@student.kit.edu

Definition Produkt von Relationen

Es seinen A, B und C Mengen und $R \subseteq A \times B$, $S \subseteq B \times C$ Relationen. Dann ist
$S \circ R := \{(a, c) \in A \times C | \exists b \in B \text{ mit } (a, b) \in R \wedge (b, c) \in S\}$
das Produkt der Relationen R und S.

Bemerkung

$S \circ R$ ist eine Relation auf A und C, bildet also von A nach C ab.

Assoziativität des Produktes

Es seien A, B, C und D Mengen und $R \subseteq A \times B$, $S \subseteq B \times C$ sowie $T \subseteq C \times D$ Relationen. Dann gilt
$(T \circ S) \circ R = T \circ (S \circ R)$.

Lukas Bach, lukas.bach@student.kit.edu

Homogene Relation

Es seien A und B Mengen und $R \subseteq A \times B$ eine Relation. R heißt homogen, wenn $A = B$ und heterogen, wenn $A \neq B$ gilt.

Identität

Sei M eine Menge. $I_M := \{(x, x) | x \in M\}$

Potenz von Relationen

Sei M eine Menge und $R \subseteq M \times M$ eine homogene Relation. Dann definieren wir R^i für $i \in \mathbb{N}_0$ folgendermaßen:

- $R^0 := I_M$
- Für alle $i \in \mathbb{N}_0 : R^{i+1} := R^i \circ R$

Also $R^4 = R \circ R \circ R \circ R$.

Lukas Bach, lukas.bach@student.kit.edu

Reflexitivität

Karlsruher Institut für Technologie

Satz über das neutrale Element

Es seien A und B Mengen und $R \subseteq A \times B$ eine Relation. Dann gilt: $R \circ I_B = R = I_A \circ R$.

Reflexivität

Sei M eine Menge und $R \subseteq M \times M$ eine homogene Relation. Wenn für alle $x \in M : (x, x) \in R$, nennt man R reflexiv.
Also jedes Element der Definitionsmenge der Relation wird auf sich selbst abgebildet (und vielleicht auch auf andere Elemente abgebildet).

Lemma

Sei M eine Menge und $R \subseteq M \times M$ eine homogene Relation. R ist genau dann reflexiv, wenn $I_M \subseteq R$ gilt.

Lukas Bach, lukas.bach@student.kit.edu

Transitivität

Karlsruher Institut für Technologie

Transitivität

Sei M eine Menge und $R \subseteq M \times M$ eine homogene Relation.
R heißt transitiv, wenn:
$\forall x, y, z \in M : (x, y) \in R \wedge (y, z) \in R \rightarrow (x, z) \in R$

Lemma

Sei M eine Menge und $R \subseteq M \times M$ eine homogene Relation. R ist genau dann transitiv, wenn $R \circ R \subseteq R$.

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben

Sei $M := \{1, 2, 3\}$.

- Ist $R := \{(1, 1), (1, 2), (2, 3)\}$ transitiv? Nein!
- Ist R reflexiv? Nein!
- Wie müsste R aussehen, um transitiv zu sein?
- Ist $S := \{(1, 1), (1, 2), (1, 3), (2, 2), (2, 3)\}$ reflexiv? Nein!
- Ist S transitiv? Ja!
- Wie müsste S aussehen, um reflexiv zu sein?

Lukas Bach, lukas.bach@student.kit.edu

Reflexiv-transitive Hülle

Karlsruher Institut für Technologie

Definition

Sei M eine Menge und $R \subseteq M \times M$ eine homogene Relation.
Dann nennt man $R^* := \bigcup_{i \in \mathbb{N}_0} R^i$ die reflexiv-transitive Hülle von R.

Satz

- R^* ist reflexiv
- R^* ist transitiv
- R^* ist die kleinste Relation, die reflexiv und transitiv ist und $R \subseteq R^*$ erfüllt.

Bemerkung

- Sei M eine Menge und $R \subseteq M \times M$ eine homogene, reflexive und transitive Relation. Dann gilt $R^* = R$.

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben

- Sei $M = \{1, 2, 3\}$ und $R := \{(1, 1), (1, 2), (2, 3)\}$ Was ist R^*?

$\rightarrow$ $R^* = \{(1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3)\}$

- Sei M eine Menge und $R \subseteq M \times M$ eine homogene Relation. Was ist $(R^*)^*$?

$\rightarrow$ $(R^*)^* = R^*$

- $M := \{1, 2, 3, 4\}$ und $R := \{(1, 2), (2, 3), (3, 4), (4, 1)\} \subseteq M \times M$. Ist R reflexiv? Ist R transitiv? Nein und nein!

Lukas Bach, lukas.bach@student.kit.edu

Die Relationen R und S über $\mathbb{N}_0$ seien gegeben durch:

- Für alle $a, b \in \mathbb{N}_0 : aRb \Leftrightarrow a|b$ (a ist Teiler von b)
- Für alle $a, b \in \mathbb{N}_0 : aSb \Leftrightarrow ggT(a, b) = 1$

Prüfe auf Reflexivität und Transitivität!

$\rightarrow$ R ist transitiv, aber nicht reflexiv.

$\rightarrow$ S ist reflexiv, aber nicht transitiv. [TODO]

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 15.12.2016

Lukas Bach, lukas.bach@student.kit.edu

Prädikatenlogik

Lukas Bach, lukas.bach@student.kit.edu

Grundlagen zu Prädikatenlogik

Prädikatenlogik (PL) erweitert Aussagenlogik durch Ergänzen von "Prädikaten", einer Art von Funktionen, die Wahrheitswerte zurückgeben.
Alphabet der Prädikatenlogik:

- $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)$, also Alphabet der Aussagenlogik.
- $\forall$ Allquantor ($\forall x$ heißt "für alle x gilt...)
- $\exists$ Existenzquantor ($\exists x$ heißt "es existiert min. ein x... für das gilt...)
- $x, y, z, x_i \in Var_{PL}$ Variablen
- $c, d, c_i \in Const_{PL}$ Konstanten
- $f, g, h, f_i \in Fun_{PL}$ Funktionen
- $R, S, R_i \in Rel_{PL}$ Relationen (funktionieren ähnlich wie Funktionen)
- $\doteq$ Objektgleichheit
- , Komma

Lukas Bach, lukas.bach@student.kit.edu

Gliederung der Prädikatenlogik

Terme

Ein Term ist ein Element aus der Sprache über $A_{Ter} := \{(,), ,\} \cup Var_{PL} \cup Const_{PL} \cup Fun_{PL}$.

Atomare Formeln

Atomare Formeln sind zum Beispiel

- Objektgleichheiten $f_1 \doteq f_2$
- Relation von Termen $R(t_1, t_2, ...)$

Stelligkeit einer Funktion

Die Stelligkeit $ar(f) \in \mathbb{N}_+$ einer Funktion gibt die Anzahl der Parameter von f an. (Analog Stelligkeit von Relationen $ar(R)$)

Verständnis von Termen, Atomaren Formeln, Stelligkeit

Lukas Bach, lukas.bach@student.kit.edu

- Woraus kann ein Term bestehen?

$\longrightarrow$ Aus Klammern (,), Kommas ,, Variablen, Konstanten, Funktionen.

- Was davon sind atomare Formeln: $R(x) \wedge S(f(x, c))$, $R(x, g(c, f(y, x)))$?

$\longrightarrow$ Nein, ja.

- Was sind die Stelligkeiten folgender Funktionen: $f(a, b, c), g(a), h(a, b)$?

$\longrightarrow$ $3, 1, 2$.

Lukas Bach, lukas.bach@student.kit.edu

Grammatik der Prädikatenlogik

Karlsruher Institut für Technologie

Prädikatenlogische Formeln werden durch die Grammatik $G := (N_{Ter}, A_{Ter}, T, P_{Ter})$ erzeugt mit:

- $m + 1$ Nichtterminalsymbolen $N_{Ter} := \{T\} \cup \{L_i | i \in \mathbb{N}_+ \text{ und } i \leq m\}$ ($m =$ Maximale Stelligkeit von Funktionen)
- Terminalsymbolen: Alphabet, aus dem Terme erzeugbar sind
- Produktionen

$$\begin{aligned} L_{i+1} &\to L_i, T && \text{für jedes } i \in \mathbb{N}_+ \text{ mit } i < m \\ L_1 &\to T && \\ T &\to c_i && \text{für jedes } c_i \in Const_{PL} \\ T &\to x_i && \text{für jedes } x_i \in Var_{PL} \\ T &\to f_i(L_{ar(f_i)}) && \text{für jedes } f_i \in Fun_{PL} \end{aligned}$$

Beispiel: Seien Funktionen f, g mit $ar(f) = 2$, $ar(g) = 1$, Konstante c und Variablen x, y gegeben. Was kann man damit machen?

Grammatik der Prädikatenlogik

Lukas Bach, lukas.bach@student.kit.edu

Beispiel: Seien Funktionen f, g mit $ar(f) = 2, ar(g) = 1$, Konstante c und Variablen x, y gegeben. Was kann man damit machen?

Dann:

- $N_{Ter} = \{T, L_1, L_2\}$
- $P_{Ter} = \{L_2 \rightarrow L_1, T$

$$L_1 \rightarrow T$$
$$T \rightarrow c$$
$$T \rightarrow x$$
$$T \rightarrow y$$
$$T \rightarrow g(L_1)$$
$$T \rightarrow f(L_2)\}$$

Aufgabe zu Grammatiken und Prädikatenlogik

Welche dieser Formeln entsprechen dieser Grammatik?

- $f(c, g(x))$
- $f(x, y, c)$
- $g(f(c, c))$
- $g(g(f(g(x), g(f(c, c)))))$
- $g(c, f)c)$

Bilde die Ableitungsbäume zu den korrekten Formeln.

Lukas Bach, lukas.bach@student.kit.edu

Bindungsstärken

Karlsruher Institut für Technologie

Bindungsstärke

Verschiedene Operanden “binden” stärker als andere. Wenn ein Operand stärker als die umliegenden Operanden bindet, tritt derselbe Effekt auf, wie wenn Klammerung geschehen würde.

Bindungsstärken absteigend:

- $\forall / \exists, \neg, \wedge, \vee, \rightarrow / \leftarrow, \leftrightarrow$

Finde äquivalente Formeln, die mit möglichst wenig Klammern auskommen:

- $\exists x \forall y (R(f(x), g(x)) \vee \forall z R(c, x))$

Quantoren

Lukas Bach, lukas.bach@student.kit.edu

- $\forall x p(x)$ heißt: für alle $x \in D$ gilt die Aussage $p(x)$.
- $\exists x p(x)$ heißt: für (mindestens) ein $x \in D$ gilt die Aussage $p(x)$.
- Gilt $\forall x \exists y \quad p(x, y) = \exists y \forall x \quad p(x, y)$?
  - Zum Beispiel $p(x, y)$:="Person x ist mit Person y verheiratet.
  - Also:
    - $\forall x \exists y \quad p(x, y) =$ Für jede Person x gibt es eine Person y, mit der x verheiratet ist.
    - $\exists y \forall x \quad p(x, y) =$ Es gibt eine Person y, sodass für alle Personen x gilt, dass x mit allen Personen y verheiratet ist.
  - Eher nicht. Reihenfolge ist also wichtig!

Lukas Bach, lukas.bach@student.kit.edu

Bindungsbereich von Quantoren

Karlsruher Institut für Technologie

Quantoren binden Variablen nur zu der zugehörigen Teilformel.

- Zum Beispiel: $p(x) \wedge \forall x \exists y (p(x) \wedge q(x, y, z)) \to r(x)$
- Welcher Teil der Formel muss für alle x gelten? Welcher für alle y?
- Variablen, die nicht im Wirkungsbereich eines Quantors liegen, nennt man frei.

Überschattung ist möglich, daher durch Quantoren definierte Variablen beziehen sich immer auf den nächsten Quantor.

- Ist $\forall x (p(x) \wedge \forall x (\neg p(x)))$ erfüllbar?
- Ja: $\forall x (p(x) \wedge \forall \hat{x} (\neg p(\hat{x})))$

Lukas Bach, lukas.bach@student.kit.edu

Bindungsbereich von Quantoren

Substitution ist möglich.Dabei wird eine freie Variable durch einen Term ersetzt, die Substitution wird mit $\beta[a/b]$ bezeichnet, wobei a durch b ersetzt wird.

Führe die folgenden Substitutionen durch:

- $\beta[x/5](p(x) \vee q(x, y)) = p(5) \vee q(5, y)$
- $\beta[x/5](p(x) \vee \forall x(q(x, y)) = p(5) \vee \forall x(q(x, y))$
- $\beta[x/y, y/x, z/f(z)](p(z) \wedge q(x, y)) = p(f(z)) \wedge q(y, x)$

Welche der Variablen sind gebunden (und durch welche Quantoren), welche sind frei?

- $p(x) \rightarrow \forall x \exists y(p(x) \wedge q(y, z) \leftrightarrow \forall z(q(x, z)))$
- $\forall y(p(f(x, y))) \vee \exists z(q(z, f(y, z)))$

Lukas Bach, lukas.bach@student.kit.edu

Semantik von prädikatenlogischen Formeln

Um prädikatenlogische Formeln zu interpretieren, brauchen wir einige neue Mengen:

- Interpretation (D, I), bestehend aus...
  - Universum $D \neq \emptyset$ mit...
    - $I(c_i) \in D$ für $c_i \in Const_{PL}$
    - $I(f_i) : D^{ar(f_i)} \to D$ für $f_i \in Fun_{PL}$
    - $I(R_i) \subseteq D^{ar(R_i)}$ für $R_i \in Rel_{PL}$
    - I weißt also den Komponenten Bedeutungen zu, “definiert diese”
  - Variablenbelegung $\beta : Var_{PL} \to D$, z.B. $\beta(x) := 3$, $\beta(y) := 11$
    - β definiert also Variablenwerte
- Damit können wir prädikatenlogische Formeln definieren!

$val_{D,I,\beta}$

Die Funktion $val_{D,I,\beta} : L_{Ter} \cup L_{For} \to D \cup \mathbb{B}$ weißt einer prädikatenlogischen Formel eine Bedeutung (Wahrheitsgehalt für Formeln und Element des Universums für Terme) zu.

Lukas Bach, lukas.bach@student.kit.edu

Beispiel zur Semantik

Karlsruher Institut für Technologie

Unterschied zwischen $val_{D,I,\beta}$ und I? I ist für Einzelteilen (Konstanten, Funktionen, Relationen) eine Bedeutung zuweist, und $val_{D,I,\beta}$ einer ganzen Formel eine Bedeutung zuweist.

Beispiel:

Sei $D := \mathbb{N}_0, I(c) := 10, ar(f) := 2, ar(p) := 1, ar(q) := 2, \beta(x) := 7$.
Sei $I(f) : \mathbb{N}_0^2 \to \mathbb{N}_0, (x, y) \mapsto x - y$.
Sei $ar(R) := 2, I(R) := \{(x, y) | x \leq y\}$.
Sei $I(p(x)) = w :\Leftrightarrow x \geq 5, I(q(x, y)) = w :\Leftrightarrow x \geq y$.

Lukas Bach, lukas.bach@student.kit.edu

Beispiel zur Semantik

Sei $D := \mathbb{N}_0, I(c) := 10, ar(f) := 2, ar(p) := 1, ar(q) := 2, \beta(x) := 7$.
Sei $I(f) : \mathbb{N}_0^2 \to \mathbb{N}_0, (x, y) \mapsto x - y$.
Sei $ar(R) := 2, I(R) := \{(x, y) | x \leq y\}$.
Sei $I(p(x)) = w :\Leftrightarrow x \geq 5, I(q(x, y)) = w :\Leftrightarrow x \geq y$.

- $T_1 := p(x) \to \exists y(q(y, x) \wedge p(y))$, was ist $val_{D,I,\beta}(T_1)$?
  - Wähle $y = 8 \in \mathbb{N}_0$. Dann: $I(q(8, 7)) = w$, $I(p(8)) = w$, also $val_{D,I,\beta}(\exists y(q(y, x) \wedge p(y))) = w$, und $val_{D,I,\beta}(T_1) = w$.
- $T_2 := p(x) \to \exists y(q(f(c, y), x) \wedge p(y))$, was ist $val_{D,I,\beta}(T_2)$?
  - $val_{D,I,\beta}(p(x)) = w$
  - $val_{D,I,\beta}(q(f(c, y), x)) = val_{D,I,\beta}(q(f(10, y), x)) = val_{D,I,\beta}(q(10 - y, 7)) = w$ für $y \in \{0, 1, 2\}$.
  - $val_{D,I,\beta}(p(y)) = w$ für $y \geq 5$.
  - Also: $val_{D,I,\beta}(T_2) = f$.

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben zu Prädikatenlogik

Aufgaben zu Prädikatenlogik

Gegeben sind folgende Prädikate:

- $\mathit{Vater}(x, y)$:= wahr, gdw. x Vater von y ist, analog $\mathit{Mutter}(x, y)$.
- $\mathit{Männlich}(x, y)$:= wahr, gdw. x männlich ist, analog $\mathit{Weiblich}(x)$.
- $\mathit{Verheiratet}(x, y)$:= wahr, gdw. x und y verheiratet sind.

Drücke die folgenden Aussagen mit prädikatenlogischen Formeln aus:

- Jede männliche Person hat eine Mutter.
  - $\forall x \exists y (\mathit{Männlich}(x) \rightarrow \mathit{Mutter}(y, x))$
  - Kann eine Person jetzt auch mehr als eine Mutter haben? Widerspricht das der ursprünglichen Aussage?
- Jeder Mann hat Kinder (plural!).
  - $\forall x \exists y \exists z (\mathit{Männlich}(x) \rightarrow (\mathit{Vater}(x, y) \wedge \mathit{Vater}(x, z) \wedge \neg(y \doteq z)))$

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben zu Prädikatenlogik

Aufgaben zu Prädikatenlogik

Gegeben sind folgende Prädikate:

- $\mathit{Vater}(x, y) :=$ wahr, gdw. x Vater von y ist, analog $\mathit{Mutter}(x, y)$.
- $\mathit{Männlich}(x, y) :=$ wahr, gdw. x männlich ist, analog $\mathit{Weiblich}(x)$.
- $\mathit{Verheiratet}(x, y) :=$ wahr, gdw. x und y verheiratet sind.

Drücke die folgenden Aussagen mit prädikatenlogischen Formeln aus:

- Jede Frau ist mit höchstens einem Mann verheiratet.
  - $\forall x \forall y \forall z (\mathit{Weiblich}(x) \land ((\mathit{Männlich}(y) \land \mathit{Männlich}(z) \land \neg(y \doteq z) \land \mathit{Verheiratet}(x, y)) \rightarrow \neg \mathit{Verheiratet}(x, z)))$
- Wer männlich ist, ist nicht weiblich und umgekehrt.
  - $\forall x (\mathit{Männlich}(x) \rightarrow \neg \mathit{Weiblich}(x) \land \mathit{Weiblich}(x) \rightarrow \neg \mathit{Männlich}(x))$

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 22.12.2016

Lukas Bach, lukas.bach@student.kit.edu

Algorithmen

Lukas Bach, lukas.bach@student.kit.edu

Algorithmen

- Es existiert eine **endliche** Beschreibung
- Es wird zu einer beliebig großen, aber **endlichen** Eingabe eine **endliche** Ausgabe berechnet
- Es finden **endlich** viele Schritte statt (der Algorithmus terminiert)
- Deterministisch (bei mehrmaliger Ausführung kommt immer das selbe raus)

Lukas Bach, lukas.bach@student.kit.edu

Hier verwendeter Pseudocode

- Zuweisungssymbol $\leftarrow$
- Schlüsselwörter für Verzweigungen **if, then, else, fi**
- Schlüsselwörter für Schleifen **while, do, od, for, to**
- Symbole für Konstanten, Funktionen und Relationen

Lukas Bach, lukas.bach@student.kit.edu

Eine **if**-Verzweigung

if $x < y$ **then**
    $s \leftarrow x$
else
    $s \leftarrow y$
fi

Eine **while**-Schleife

while $x > 0$ **do**
    $x \leftarrow x \operatorname{div} 2$
    $s \leftarrow s + x$
od

Eine **for**-Schleife

for $i \leftarrow 1$ **to** n **do**
    $s \leftarrow s + i$
od

Lukas Bach, lukas.bach@student.kit.edu

Was kann man mit Algorithmen machen?

Karlsruher Institut für Technologie

- Komplexe Algorithmen mit Pseudocode definieren zu Sortierung, Graphen, Datenstrukturen, im Modul Algorithmen I
- Laufzeitanalyse von Algorithmen, später.
- Korrektheitsbeweise, jetzt.

Lukas Bach, lukas.bach@student.kit.edu

Korrektheitsbeweise

Karlsruher Institut für Technologie

Wie findet man heraus, ob ein Algorithmus korrekt funktioniert?

- Durch den Beweis von Zusicherungen, die an bestimmten Stellen des Algorithmus gelten.

Was sind Zusicherungen?

- prädikatenlogische Formeln, die Aussagen über (Zusammenhänge zwischen) Variablen machen

Lukas Bach, lukas.bach@student.kit.edu

Das Hoare-Tripel

Definition

$\{P\}S\{Q\}$ heißt Hoare-Tripel. Dabei gilt:

- S ist ein Programmstück im Pseudocode
- P und Q sind Zusicherungen

- P nennt man Vorbedingung, Q Nachbedingung
- Prädikatenlogische Formeln
- Beispiel (Vorausblick): $\{x \doteq 1\}x \leftarrow x + 1\{x \doteq 2\}$
- Meistens in jeder Zeile nur eine Zeile Code oder ein Zusicherungsblock

Lukas Bach, lukas.bach@student.kit.edu

Das Hoare-Tripel

Karlsruher Institut für Technologie

Gültigkeit von Hoare-Tripeln

$\{P\}S\{Q\}$ ist gültig, wenn für jede gültige Interpretation (D, I) und Variablenbelegung β gilt:
Aus

- $val_{D,I,\beta}(P) = w$
- β' ist Variablenbelegung nach Ausführung von S

folgt $val_{D,I,\beta'}(Q) = w$

Lukas Bach, lukas.bach@student.kit.edu

Zuweisung

Karlsruher Institut für Technologie

Axiom HT-A

- Sei $x \leftarrow E$ eine Zuweisung
- Q eine Nachbedingung von $x \leftarrow E$ und
- $\sigma_{\{x/E\}}$ kollisionsfrei für Q

Dann ist $\sigma_{\{x/E\}}(Q)x \leftarrow E\{Q\}$ ein gültiges Hoare-Tripel

Bemerkung

- $\sigma_{\{x/E\}}$ ist die Substitution von x mit E
- Bei Anwendung der Regel rückwärts vorgehen

Lukas Bach, lukas.bach@student.kit.edu

Beispiel

Betrachte die Zuweisung

$x \leftarrow x + 1$

und die Nachbedingung

$\{x \doteq 1\}$

Nach HT-A gilt

$\{x + 1 \doteq 1\}\ x \leftarrow x + 1\ \{x \doteq 1\}$ ist ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu

Ableitungsregeln: HT-E

- Verstärkung der Vorbedingung
- Abschwächung der Nachbedingung

HT-E

Wenn $\{P\}S\{Q\}$ ein gültiges Hoare-Tripel ist und $P' \vdash P$ und $Q \vdash Q'$ gelten, dann folgt:
$\{P'\}S\{Q'\}$ ist ein gültiges Hoare-Tripel.

Bemerkung

$B \vdash A :\Leftrightarrow$ Aussage A ist syntaktisch aus Aussage B ableitbar

Lukas Bach, lukas.bach@student.kit.edu

Beispiel

Angenommen es sei $\{y > 3\}\ x \leftarrow y - 1\ \{x > 1\}$ ein gültiges Hoare-Tripel.
Es gilt $\{(y > 4)\} \vdash \{(y > 3)\}$ und $\{(x > 1)\} \vdash \{(x > 0)\}$.
Also folgt nach HT-E:
$\{y > 4\}\ x \leftarrow y - 1\ \{x > 0\}$ ist ein gültiges Hoare-Tripel.

Bemerkung

Es müssen sich nicht unbedingt beide Bedingungen ändern!
Aus $\{(y > 3)\} \vdash \{(y > 3)\}$ und $\{(x > 1)\} \vdash \{(x > 0)\}$
folgt nach HT-E auch
$\{y > 3\}\ x \leftarrow y - 1\ \{x > 0\}$ ist ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu

Ableitungsregeln: HT-S

Hintereinanderausführung von durch Hoare-Triple bewiesene Code Segmente sind selbst gültig.

HT-S

Wenn $\{P\}S_1\{Q\}$ und $\{Q\}S_2\{R\}$ gültige Hoare-Tripel sind, dann folgt:$\{P\}S_1; S_2\{R\}$ ist ein gültiges Hoare-Tripel.

Bemerkung

";" trennt hier zwei Programmstücke

Lukas Bach, lukas.bach@student.kit.edu

Beispiel

Angenommen es seien $\{y > 3\}\ x \leftarrow y - 1\ \{x > 1\}$ und $\{x > 1\}\ z \leftarrow x - 1\ \{z > -1\}$ gültige Hoare-Tripel.

Dann folgt nach HT-S:

$\{y > 3\}\ x \leftarrow y - 1; z \leftarrow x - 1\ \{z > -1\}$ ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu

Bedingte Anweisungen

Karlsruher Institut für Technologie

HT-I

Wenn $\{P \land B\} S_1 \{Q\}$ und $\{P \land \neg B\} S_2 \{Q\}$ gültige Hoare-Tripel sind, dann folgt:

$\{P\}$
  if B **then** S_1
  else S_2
  fi
$\{Q\}$

ist ein gültiges Hoare-Tripel.

Beispiel

Lukas Bach, lukas.bach@student.kit.edu

$\{ x = a \wedge y = b \}$

if $x > y$

then

$\{ \dots \}$

$z \leftarrow x$

$\{ \dots \}$

else

$\{ \dots \}$

$z \leftarrow y$

$\{ \dots \}$

fi

$\{ z = \min(a, b) \}$

Beispiel

Lukas Bach, lukas.bach@student.kit.edu

$\{ x = a \wedge y = b \}$

if $x > y$

then

$\{ x = a \wedge y = b \wedge \neg(x > y) \}$

$\{ x = \min(a,b) \}$

$z \leftarrow x$

$\{ z = \min(a,b) \}$

else

$\{ x = a \wedge y = b \wedge x > y \}$

$\{ y = \min(a,b) \}$

$z \leftarrow y$

$\{ z = \min(a,b) \}$

fi

$\{ z = \min(a,b) \}$

Lukas Bach, lukas.bach@student.kit.edu

Schleifen

Karlsruher Institut für Technologie

HT-W

Wenn $\{I \wedge B\}S\{I\}$ ein gültiges Hoare-Tripel ist, dann folgt:

$\{I\}$
while B **do** S
od
$\{I \wedge \neg B\}$

ist ein gültiges Hoare-Tripel.

Schleifeninvariante

Lukas Bach, lukas.bach@student.kit.edu

- Eine spezielle Zusicherung
- Schleifeninvarianten müssen **vor**, **während** und **nach** jedem Schleifendurchlauf gelten
- Garantiert, dass die Schleife nicht während einem beliebigen Durchlauf “kaputt” geht.

Lukas Bach, lukas.bach@student.kit.edu

Beispiel

$\{\, x = a \wedge y = b \,\}$

$\{ \dots \}$

while $y \neq 0$

do

$\{ \dots \}$

$y \leftarrow y - 1$

$\{ \dots \}$

$x \leftarrow x + 1$

$\{ \dots \}$

od

$\{ \dots \}$

$\{\, x = a + b \,\}$

Lukas Bach, lukas.bach@student.kit.edu

Beispiel

$\{ x = a \wedge y = b \}$

$\{ x + y = a + b \}$

while $y \neq 0$

do

$\quad \{ x + y = a + b \wedge y \neq 0 \}$

$\quad \{ x + 1 + y - 1 = a + b \}$

$\quad y \leftarrow y - 1$

$\quad \{ x + 1 + y = a + b \}$

$\quad x \leftarrow x + 1$

$\quad \{ x + y = a + b \}$

od

$\{ x + y = a + b \wedge \neg(y \neq 0) \}$

$\{ x = a + b \}$

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 12.01.2017

Lukas Bach, lu-
kas.bach@student.kit.edu

Graphen

Lukas Bach, lukas.bach@student.kit.edu

Graphen

Karlsruher Institut für Technologie

Definition: Graph

Ein Graph $G = (V, E)$ ist ein Tupel aus:

- Einer endlichen, nichtleeren Knotenmenge V
- Einer endlichen Kantenmenge $E \subseteq V \times V$

Beispiel: Knotenmenge $V := \{a, b, c, d\}$. Kantenmenge könnte zum Beispiel sein...

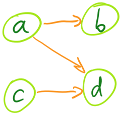
- $E := \{(a, b), (c, d), (a, d)\}$
- $E := \{(a, a), (b, b), (c, c)\}$
- $E := \emptyset$

Lukas Bach, lukas.bach@student.kit.edu

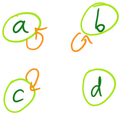
Wie sehen diese Graphen aus?

Beispiel: Knotenmenge $V := \{a, b, c, d\}$. Kantenmenge könnte zum Beispiel sein...

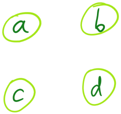
- $V := \{a, b, c, d\}, E := \{(a, b), (c, d), (a, d)\}$



- $V := \{a, b, c, d\}, E := \{(a, a), (b, b), (c, c)\}$



- $V := \{a, b, c, d\}, E := \emptyset$

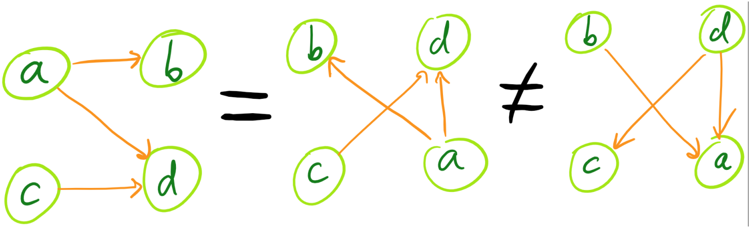


Lukas Bach, lukas.bach@student.kit.edu

Wann Angabe als Menge, wann als Visualisierung?

Wir verwenden gezeichnete Graphen und deren Definition als Mengen als äquivalent.

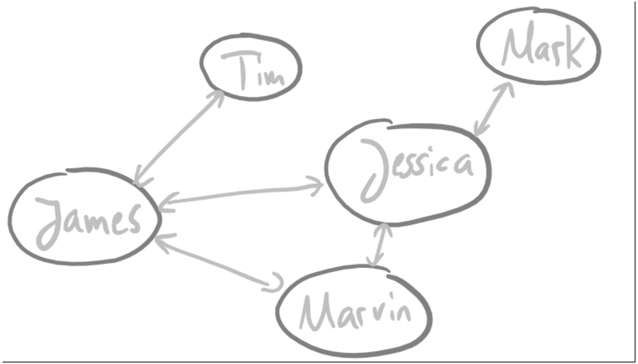
- $\{(a,b),(c,d),(a,d)\} = \{(a,b),(a,d),(c,d)\} \neq \{(b,a),(d,c),(d,a)\}$, also Kantenmenge mit unterschiedlichen Reihenfolgen darstellbar. Genauso die Knotenmenge.



Es kann also in jedem Fall der Graph sowohl als “Visualisierung” oder als Menge angegeben werden, beide Varianten sind formal korrekt.

Lukas Bach, lukas.bach@student.kit.edu

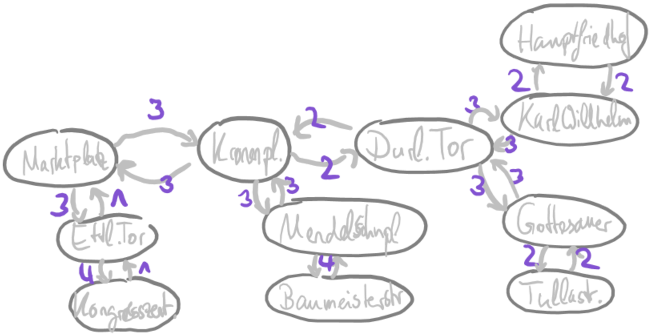
Praxisbeispiel: Soziales Netzwerk



- Ist Person A direkt mit Person B befreundet? $\Leftrightarrow$ Gibt es eine Kante (A, B)?
- Ist Person A über maximal 2 verschiedene Leute mit Person B befreundet?$\Leftrightarrow$ Gibt es einen Pfad von A nach B mit maximaler Länge 3?
- Wieviele Freunde hat Person A?$\Leftrightarrow$ Welchen Grad hat Person $A \in V$?

Lukas Bach, lukas.bach@student.kit.edu

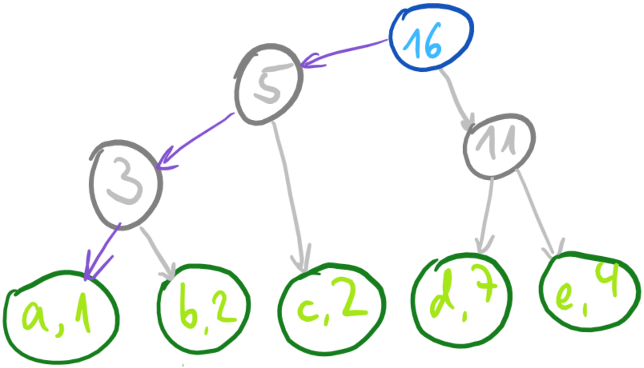
Praxisbeispiel: Wie kommt man am schnellsten von A nach B



- Kantengewichtung: Jeder Kante wird eine Zahl $c \in \mathbb{R}$ zugewiesen.
- Wie lange dauert der kürzeste Weg von Kongresszentrum nach Hauptfriedhof? $\Leftrightarrow$ Wie lang ist ein kürzester Pfad von Kongresszentrum nach Hauptfriedhof?
- Wo kommt man von Kronenplatz überall innerhalb von 5 Zeiteinheiten hin? $\Leftrightarrow$ Für welche Orte $v \in V$ existiert ein Pfad $(\mathit{Kronenplatz}, ..., v)$ mit einer Länge von maximal 5?

Lukas Bach, lukas.bach@student.kit.edu

Praxisbeispiel: Huffman-Bäume



- Wie lang ist die Kodierung vom Zeichen *c*?⇔ Wie lang ist der Pfad von Wurzel zu Knoten *c*? In diesem Fall 2.
- Wie viele Zeichen werden kodiert? ⇔ Wie viele Knoten sind von der Wurzel erreichbar, die selbst keine ausgehenden Kanten haben? ⇔ Wie viele Blätter hat der Baum?

Lukas Bach, lukas.bach@student.kit.edu

Ungerichtete Graphen

Karlsruher Institut für Technologie

- Bis jetzt: Gerichtete Graphen, dh. Kanten (u, v) hatten eine Richtung von Knoten u nach Knoten v.

Ungerichteter Graph

Ein ungerichteter Graph ist ein Graph, dessen Kanten Mengen, und keine Tupel sind.

- Beispiel: Statt Kante (u, v) jetzt Kante $\{u, v\} = \{v, u\}$.
- Information über Richtung geht also verloren, Kanten verbinden nur noch Knoten, ohne sich zu merken, welcher Knoten Start und welcher Ziel ist.

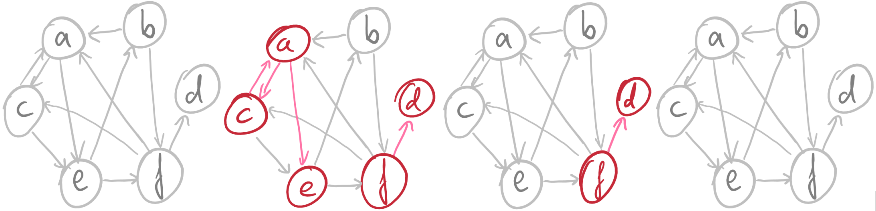
Lukas Bach, lukas.bach@student.kit.edu

Teilgraph

Teilgraph

Zu einem Graph $G := (V, E)$ ist ein Teilgraph definiert als $G' = (V', E')$, falls gilt $V' \subseteq V$ und $E' \subseteq E$.

- Beispiel: Sei $G := (V, E)$ mit $V := \{a, b, c, d, e, f\}$ und $E := \{(b, a), (b, f), (f, d), (e, f), (f, a), (e, b), (a, e), (f, c), (a, c), (c, a), (c, e)\}$
- Ist ein Graph mit $V_1 := \{a, c, d, e, f\}$, $E_1 := \{(a, c), (c, a), (a, e), (f, d)\}$ ein Teilgraph von G?
- Ist ein Graph mit $V_2 := \{d, f\}$, $E_2 := \{(f, d)\}$ ein Teilgraph von G?
- Ist ein Graph mit $V_3 = E_3 = \emptyset$ ein Teilgraph von G?



Lukas Bach, lukas.bach@student.kit.edu

Teilgraph

Karlsruher Institut für Technologie

Teilgraph

Zu einem Graph $G := (V, E)$ ist ein Teilgraph definiert als $G' = (V', E')$, falls gilt $V' \subseteq V$ und $E' \subseteq E$.

- Beispiel: Sei $G := (V, E)$ mit $V := \{a, b, c, d, e, f\}$ und $E := \{(b, a), (b, f), (f, d), (e, f), (f, a), (e, b), (a, e), (f, c), (a, c), (c, a), (c, e)\}$
- Ist ein Graph mit $V_4 := \{a, b\}$, $E_4 := \{(f, d)\}$ ein Teilgraph von G?
- Ist ein Graph mit $V_5 := \{g, a\}$, $E_5 := \{(g, a), (a, g)\}$ ein Teilgraph von G?

Weg/Pfad

Lukas Bach, lukas.bach@student.kit.edu

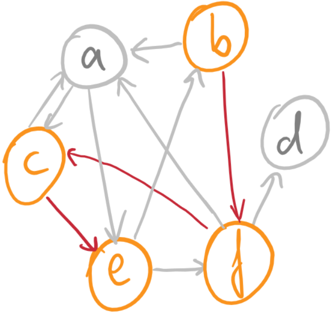
Pfad informell

Ein Pfad $(u, ..., v)$ ist eine Aneinanderreihung von Knoten, die jeweils mit Kanten verbunden sind, sodass man über das traversieren der Kanten vom Startknoten $u \in V$ zum Zielknoten $v \in V$ kommt.

Anmerkung: Wenn man sich einen Knoten $x \in V$ merkt und eine Kante $(x, y) \in E$ traversiert, so gelangt man zu Knoten y.

Pfad formell

Ein Pfad $P := (v_0, v_1, ..., v_n)$ der Länge n ist eine Permutation auf V , wobei gilt: $\forall i \in \mathbb{Z}_n : (v_i, v_{i+1}) \in E$.



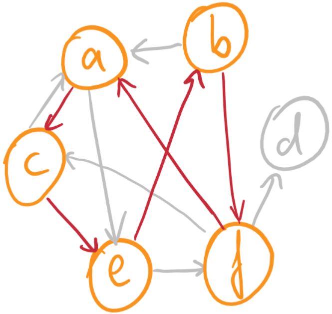
Der Pfad (b, f, c, e) ist ein möglicher Pfad von b nach e der Länge 3.
Gibt es noch andere solcher Pfade?

Lukas Bach, lukas.bach@student.kit.edu

Zyklus

Zyklus

Ein Zyklus ist ein Pfad $(v_1, ..., v_n)$ mit $v_1 = v_n$.



Der Pfad (b, f, a, c, e) ist ein möglicher Zyklus.
Gibt es noch andere Zyklen?

Lukas Bach, lukas.bach@student.kit.edu

Zusammenhängend

Zusammenhängender Graph

Ein ungerichteter Graph heißt zusammenhängend, wenn gilt: $\forall u, v \in V \exists$ Pfad von u nach v.

Stark zusammenhängender Graph

Ein gerichteter Graph heißt stark zusammenhängend, wenn gilt: $\forall u, v \in V \exists$ Pfad von u nach v.

Schwach zusammenhängender Graph

Ein gerichteter Graph heißt schwach zusammenhängend, wenn der zugehörige ungerichteter Graph zusammenhängend ist.

Lukas Bach, lukas.bach@student.kit.edu

Knotengrad

Karlsruher Institut für Technologie

Eingangsgrad

Der Eingangsgrad eines Knoten $u \in V$ ist definiert als:
$d_{-}(u) := |\{(v, u) \in E : v \in V\}|$, also die Anzahl der Kanten, die in den Knoten u zeigen.

Ausgangsgrad

Der Ausgangsgrad eines Knoten $u \in V$ ist definiert als:
$d_{+}(u) := |\{(u, v) \in E : v \in V\}|$, also die Anzahl der Kanten, die vom Knoten u aus weg zeigen.

Grad

Der Grad eines Knoten u ist definiert als: $d(u) := d_{+}(u) + d_{-}(u)$, also die Anzahl der Kanten, über die u verbunden ist.

Lukas Bach, lukas.bach@student.kit.edu

Gerichtete Bäume

- Kennt ihr schon: Huffman-Baum

Gerichteter Baum

Ein gerichteter Baum ist ein schwach zusammenhängender kreisfreier gerichteter Graph.

Ungerichteter Baum

Ein ungerichteter Baum ist ein zusammenhängender kreisfreier ungerichteter Graph.

- Bäume haben immer einen Wurzelknoten, von dem alle anderen Knoten ausgehen.
- Ungerichtete Bäume können mehrere Wurzeln haben.
- Knoten mit Grad 1 heißen Blätter.

Lukas Bach, lukas.bach@student.kit.edu

Randfälle

- Wieviele Kanten kann ein Graph mit n Knoten maximal haben? n^2
- Wieviele Kanten kann ein schlingenfreier Graph mit n Knoten maximal haben? $n^2 - n = n(n-1)$
- Wieviele Kanten kann ein ungerichteter Graph mit n Knoten maximal haben? $\frac{n(n-1)}{2} + n = \frac{n(n+1)}{2}$
- Wieviele Kanten kann ein ungerichteter schlingenfreier Graph mit n Knoten maximal haben? $\frac{n(n-1)}{2}$

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 19.01.2017

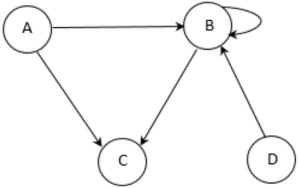
Lukas Bach, lukas.bach@student.kit.edu

Repräsentation von Graphen

Lukas Bach, lukas.bach@student.kit.edu

Repräsentation von Graphen

Wie stellen wir Graphen da?



Anschaulich ja, aber wie können wir Graphen z.B. mit Java realisieren?

Lukas Bach, lukas.bach@student.kit.edu

Objektorientierte Repräsentation von Graphen

Klassenmodell?

```
class Vertex {
   String name; //Genauer Inhalt interessiert uns nicht
}

class Edge {
   Vertex start;
   Vertex end;
}

class Graph {
   Vertex[] vertices;
   Edge[] edges;
}
```

Lukas Bach, lukas.bach@student.kit.edu

Objektorientierte Repräsentation von Graphen

+ Intuitiv
– Es lassen sich nur schwer Algorithmen hierfür entwerfen (z.B. gilt $(x, y) \in E$?)

Lukas Bach, lukas.bach@student.kit.edu

Repräsentation mit Adjazenzlisten

Karlsruher Institut für Technologie

Jeder Knoten speichert seine Nachbarn:

```
class Vertex {
    String name; //Genauer Inhalt interessiert uns nicht
    Vertex[] neighbours; //Alle Nachbarknoten
}

class Graph {
    Vertex[] vertices;
    Edge[] edges;
}
```

Lukas Bach, lukas.bach@student.kit.edu

Repräsentation mit Adjazenzlisten

- \+ Speicherplatzeffizient bei wenigen Kanten im Vergleich zur Knotenanzahl ($|E| << |V|^2$)
- \+ Flexibel mit verketteten Listen statt Arrays (Leichtes Hinzufügen und Entfernen)

Lukas Bach, lukas.bach@student.kit.edu

Repräsentation mit Adjazenzmatrix

- Was ist eine Adjazenzmatrix?
- Zu allen Paaren (i, j) mit $i, j \in V$ wird gespeichert, ob $(i, j) \in E$ gilt
- Zweidimensionales Array

```
class Graph {
    boolean[][] edges; //Größe |V| × |V|
}
```

Lukas Bach, lukas.bach@student.kit.edu

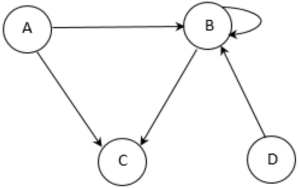
Repräsentation mit Adjazenzmatrix

- **+** Speicherplatzeffizient bei annähernd maximaler Anzahl von Kanten ($|E| \approx |V|^2$)
- $+$ Algorithmen aus linearer Algebra können verwendet werden (Matrizenrechnung)
- **–** nicht flexibel

Lukas Bach, lukas.bach@student.kit.edu

Aufgabe

Gebe alle Adjazenlisten und die Adjazenzmatrix für diesen Graphen an:



$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Lukas Bach, lukas.bach@student.kit.edu

Repräsentation von zweistelligen Relationen durch Matrizen

Wir können jede endliche zweistellige Relation durch eine Matrix darstellen!

Aufgabe

Stelle die Kleiner-Gleich-Relation auf der Menge $\{0, 1, 2, 3\}$ dar!

$$R_{\leq} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Lukas Bach, lu-
kas.bach@student.kit.edu

Erreichbarkeit

Lukas Bach, lukas.bach@student.kit.edu

Wege-Problem

- Algorithmisches Problem
- Intuitiv: Gibt es einen Weg von i nach j?

Wege-Problem

Gegeben einem Graphen $G = (V, E)$. Ist für $i, j \in V$ auch $(i, j) \in E^*$?

Ziel

- Gegeben: Adjazenzmatrix
- Gesucht: Zugehörige **Wegematrix**, für die gilt:

$$W_{ij} = \begin{cases} 1 & \text{, falls ein Weg von i nach j existiert} \\ 0 & \text{, sonst} \end{cases}$$

Lukas Bach, lukas.bach@student.kit.edu

Einschub Matrizen

Was wisst ihr zu folgenden Begriffen?

- Matrizenmultiplikation
- Matrizenaddition
- Potenzieren
- Einheitsmatrix
- Nullmatrix

Lukas Bach, lukas.bach@student.kit.edu

Quadrierte Adjazenzmatrix

Karlsruher Institut für Technologie

Aufgabe

Quadriere die Adjazenzmatrix von vorhin: $A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

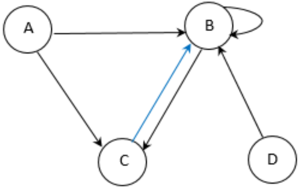
Ergebnis

$$A^2 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Lukas Bach, lukas.bach@student.kit.edu

Aufgabe

Bilde und quadriere die Adjazenzmatrix des veränderten Graphen:

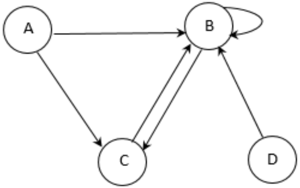


$$A' = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \text{ und } A'^2 = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Lukas Bach, lukas.bach@student.kit.edu

Aufgabe

Was fällt euch auf? Wann steht in A'^2 eine 1, wann eine 2 und was bedeutet das für unseren Graphen?



$$A' = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad A'^2 = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$\begin{array}{ccc|ccc} & & & b_{11} & b_{12} & b_{13} \\ & & & b_{21} & b_{22} & b_{23} \\ & & & b_{31} & b_{32} & b_{33} \\ \hline a_{11} & a_{12} & a_{13} & \boxed{c_{11}} & c_{12} & c_{31} \\ a_{21} & a_{22} & a_{23} & c_{21} & c_{22} & c_{23} \\ a_{31} & a_{32} & a_{33} & c_{31} & c_{32} & c_{33} \end{array}$$

Tipp: $c_{11} = a_{11} \cdot b_{11} + a_{12} \cdot b_{21} + a_{13} \cdot b_{31}$

Lukas Bach, lukas.bach@student.kit.edu

Lösung

In der i-ten Zeile und j-ten Spalte von A^2 steht die Anzahl der Wege von i nach j der Länge zwei.

$\rightarrow$ $(A^2)_{ij}$ = Anzahl der Pfade von i nach j der Länge zwei.

Aufgabe

Habt ihr Ideen, wie man herausfindet, zwischen welchen Knoten Pfade der Länge n existieren?

Lösung

Betrachte A^n!

Lukas Bach, lukas.bach@student.kit.edu

Zwei-Erreichbarkeit

Eigentlich interessiert uns nur, ob ein Pfad der Länge zwei existiert und nicht wie viele...

Definition Signum-Funktion

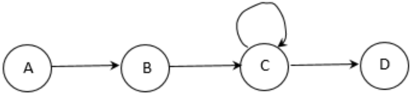
$sgn : \mathbb{R} \rightarrow \mathbb{R}$

$$x \mapsto \begin{cases} 1 & \text{, falls } x > 0 \\ 0 & \text{, falls } x = 0 \\ -1 & \text{, falls } x < 0 \end{cases}$$

$sgn(A^2)$ liefert uns die Zwei-Erreichbarkeitsmatrix

Lukas Bach, lukas.bach@student.kit.edu

Aufgabe



Gebe A^0, A^2 und die Wegematrix W an!

$$A^0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad A^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad W = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Lukas Bach, lukas.bach@student.kit.edu

Erreichbarkeit

Für Pfade beliebiger Länge erhalten wir:

$$W = sgn(A^0 + A^1 + A^2 + A^3 + ...) = sgn(\sum_{i=0}^{\infty} A^i)$$

Wir können nicht unendlich lange addieren... Ist das ein **Problem**?

Lukas Bach, lukas.bach@student.kit.edu

Erreichbarkeit- unendlich addieren?

Karlsruher Institut für Technologie

Wenn ein Pfad p der Länge $\geq n := |V|$ zwischen $i \neq j$ existiert, muss mindestens ein Knoten doppelt vorgekommen sein! Der Pfad p enthält also einen Zyklus, den wir raus kürzen können.

Ergebnis

Wenn ein Pfad p der Länge $\geq n := |V|$ zwischen $i \neq j$ existiert, existiert auch ein Pfad p' der Länge $< n$.

Für Pfade beliebiger Länge erhalten wir:

$$W = sgn(A^0 + A^1 + A^2 + A^3 + ...) = sgn(\sum_{i=0}^{\infty} A^i) = sgn(\sum_{i=0}^{n-1} A^i)$$

Lukas Bach, lukas.bach@student.kit.edu

Einfacher Algorithmus zu Berechnung der Wegematrix

Karlsruher Institut für Technologie

⟨*Matrix A sei die Adjazenzmatrix*⟩
$W \leftarrow 0$
for $i \leftarrow 0$ **to** $n-1$ **do**
  $M \leftarrow I$
  for $j \leftarrow 1$ **to** i **do**
    $M \leftarrow M \cdot A$
  od
  $W \leftarrow W + M$
od
$W \leftarrow \mathrm{sgn}(W)$

⟨*Matrix A sei die Adjazenzmatrix*⟩
$W \leftarrow 0$
for $i \leftarrow 0$ **to** $n-1$ **do**
  $M \leftarrow I$
  for $j \leftarrow 1$ **to** i **do**
    $M \leftarrow M \cdot A$
  od
  $\{ M = A^i \}$
  $W \leftarrow W + M$
  $\{ W = \sum_{k=0}^{i} A^k \}$
od
$W \leftarrow \mathrm{sgn}(W)$
$\{ W$ ist die Wegematrix $\}$

Aufwand:

$$\left(\sum_{i=0}^{n-1} i\right) \cdot$$

Wie könnte man diesen Algorithmus schneller machen?

$W \leftarrow 0$
$M \leftarrow I$

für A^i kann man A^{i-1} wiederverwenden

Lukas Bach, lukas.bach@student.kit.edu

Komplexitätstheorie

Lukas Bach, lukas.bach@student.kit.edu

Komplexitätstheorie

Wichtige Komplexitätsmaße:

- Speicherplatzbedarf
- Rechen- bzw. Laufzeit

Unterscheidung in

- Best Case (oft uninteressant)
- Average Case (schwierig zu finden, deswegen selten angegeben)
- Worst Case (meistens angegeben)

Lukas Bach, lukas.bach@student.kit.edu

Ignorieren konstanter Faktoren

Karlsruher Institut für Technologie

Definition

Seien $g, f : \mathbb{N}_0 \to \mathbb{R}_0^+$ Funktionen. Dann wächst g asymptotisch genauso schnell wie f genau dann, wenn gilt:
$\exists c, c' \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : c \cdot f(n) \leq g(n) \leq c' \cdot f(n)$

Notation

$f \asymp g$ oder $f(n) \asymp g(n)$ (äsymptotisch gleich")

Bemerkung

$\asymp$ ist eine Äquivalenzrelation

Lukas Bach, lukas.bach@student.kit.edu

Theta

Karlsruher Institut für Technologie

Definition

$\Theta(f) = \{g | g \asymp f\}$

Satz

$\forall a, b \in \mathbb{R}_+ : \Theta(a \cdot f) = \Theta(b \cdot f)$

Lukas Bach, lukas.bach@student.kit.edu

Obere und untere Schranke

Obere Schranke (Worst-Case Approximation)

$O(f) = \{g | \exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : g(n) \leq c \cdot f(n)\}$

Untere Schranke (Best-Case Approximation)

$\Omega(f) = \{g | \exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}$

Notation

- $g \preccurlyeq f$ falls $g \in O(f)$ bzw. g wächst asymptotisch höchstens so schnell wie f
- $g \succcurlyeq f$ falls $g \in \Omega(f)$ bzw. g wächst asymptotisch mindestens so schnell wie f

Bemerkung

Es gilt $\Theta(f) = O(f) \cap \Omega(f)$

Lukas Bach, lukas.bach@student.kit.edu

Lemma

$log_a n \in \Theta(log_b n)$

Beispiel

$log_2 n \in \Theta(log_8 n)$

Beweis

$$\frac{1}{3} log_2 n = \frac{1}{log_2 8} log_2 n = \frac{log_2 n}{log_2 8} = log_8 n \leq log_2 n$$

Lukas Bach, lukas.bach@student.kit.edu

Aufgabe

Gilt $log_2(n^{20}) \in \Theta(logn)$

Lösung

Ja, denn $log_2(n^{20}) = 20 \cdot log_2 n$

Lukas Bach, lu-
kas.bach@student.kit.edu

Probeklausur

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 26.01.2017

Lukas Bach, lu-
kas.bach@student.kit.edu

Komplexitätstheorie

Lukas Bach, lukas.bach@student.kit.edu

Rückblick

- Was ist $\Omega(f), \Theta(f), \mathcal{O}(f)$?
- Wieso messen wir nicht einfach Laufzeit in “Anzahl Operationen”?

Lukas Bach, lukas.bach@student.kit.edu

Obere und untere Schranke

Obere Schranke (Worst-Case Approximation)

$O(f) = \{g \mid \exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : g(n) \leq c \cdot f(n)\}$

Untere Schranke (Best-Case Approximation)

$\Omega(f) = \{g \mid \exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}$

Average-Case Approximation

$\Theta(f) = \{g \mid \exists c, c' \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : c \cdot f(n) \leq g(n) \leq c' \cdot f(n)\}$

Auf welche Weise wird hier approximiert?

Lukas Bach, lukas.bach@student.kit.edu

Gelten folgende Approximationen?

- $4n^2 + \pi n + 2\sqrt{n} \in \Theta(n^2)?$ Ja.
- $5n^2 + \pi n + 2\sqrt{n} \in \Theta(n^2)?$ Ja.
- $4n^{2,1} + \pi n + 2\sqrt{n} \in \Theta(n^2)?$ Nein.

Es sind immer nur die höchsten Faktoren interessant!

- $4n^4 + 3c^6 \in \Theta(n^4)$? Ja, c ist eine Konstante, $3c^6 = (3c^6)n^0$ hat eine kleinere Potenz als n^4.
- $\log_{4213}(n) \in \Theta(\log_2(n)$ Ja, die Basis des Logarithmus ist im O-Kalkül egal.
  - Grund: $\mathcal{O}(\log_b n) = \mathcal{O}(\frac{\log_a n}{\log_a b}) = \mathcal{O}(\frac{1}{\log_a b} \log_a n) = \mathcal{O}(\log_a n)$.
- $n! \in \Theta(n^{\pi e 2000})$ Nein, Fakultät wächst asymptotisch schneller als fast alles andere.

Lukas Bach, lukas.bach@student.kit.edu

Gelten folgende Approximationen?

- $4n^3 + 2n^2 \in \mathcal{O}(n^5)$? Ja.
- $4n^3 + 2n^2 \in \mathcal{O}(n^4)$? Ja.
- $4n^3 + 2n^2 \in \mathcal{O}(n^3)$? Ja.
- $4n^3 + 2n^2 \in \mathcal{O}(n^2)$? Nein.
- $4n^3 + 2n^2 \in \Omega(n^5)$? Nein.
- $4n^3 + 2n^2 \in \Omega(n^4)$? Nein.
- $4n^3 + 2n^2 \in \Omega(n^3)$? Ja.
- $4n^3 + 2n^2 \in \Omega(n^2)$? Ja.

Aufgabe

Lukas Bach, lukas.bach@student.kit.edu

Karlsruher Institut für Technologie

Übungsaufgabe

Entscheide für jede Zelle, ob die Formel der Zeile in der Menge der Spalte liegt.

	$\mathcal{O}(n^3)$	$\mathcal{O}(n)$	$\Theta(c!)$	$\Theta(n^\pi)$	$\Omega(n^6)$	$\Omega(n!)$
$2n^2 + 4n$	$\in$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$
π	$\in$	$\in$	$\in$	$\notin$	$\notin$	$\notin$
$\log(n)$	$\in$	$\in$	$\notin$	$\notin$	$\notin$	$\notin$
$n\log(n)$	$\in$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$
n^π	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\notin$
$12n^3 + 7000n^2$	$\in$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$
n^3	$\in$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$
$n!$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\in$

Lukas Bach, lukas.bach@student.kit.edu

- $\mathcal{O}(n^2) \cap \mathcal{O}(n) = \mathcal{O}(?)$? $= \mathcal{O}(n)$.
- $\mathcal{O}(n^2) \cap \Omega(n^3) = \emptyset$

Lukas Bach, lukas.bach@student.kit.edu

Grundlegende Reihenfolge von Größen

$$1 \preccurlyeq \log n \preccurlyeq n \log n \preccurlyeq n^2 \preccurlyeq n^3 \preccurlyeq n^{10000} \preccurlyeq n^2 \preccurlyeq 3^n \preccurlyeq 1000^n \preccurlyeq n! \preccurlyeq n^n$$

Lukas Bach, lukas.bach@student.kit.edu

Mathematische Definitionen

$$f(n) \in \Omega(g(n)) \Leftrightarrow 0 < \liminf_{n \to \infty} \frac{f(n)}{g(n)} \leq \infty$$

$$f(n) \in \Theta(g(n)) \Leftarrow 0 < \lim_{n \to \infty} \frac{f(n)}{g(n)} = c < \infty$$

$$f(n) \in \mathcal{O}(g(n)) \Leftrightarrow 0 \leq \limsup_{n \to \infty} \frac{f(n)}{g(n)} = c < \infty$$

Z

eige:

- $3n^2 + 14n + 159 \in \Theta(n^2)$
- $\log n^2 \in \Theta(\log n^3)$
- $\log^2 n \in \mathcal{O}(\log^3 n)$

Lukas Bach, lukas.bach@student.kit.edu

Komplexität mit vollständiger Induktion beweisen

Z

eige mittels vollständiger Induktion:

- $2^n \in \Theta(n^3)$
- $(n+1)! \in \Theta(n! + 2^n)$

Lukas Bach, lukas.bach@student.kit.edu

Größenordnung	Bezeichnung
$\mathcal{O}(1)$	konstante Laufzeit
$\mathcal{O}(\log n)$	logarithmische Laufzeit
$\mathcal{O}(\log^2 n)$	quadratisch logarithmische Laufzeit
$\mathcal{O}(n)$	lineare Laufzeit
$\mathcal{O}(n^2)$	quadratische Laufzeit
$\mathcal{O}(n^3)$	kubische Laufzeit
$\mathcal{O}(n^k)$	polynomielle Laufzeit

Lukas Bach, lukas.bach@student.kit.edu

```
r ← 0
for i ← 0 to n/2 do
  s ← 0
  for j ← i to n – i do
    s ← s + j
  od
  r ← s + n * i
  r ← r + s
od
```

- Wie oft wird die innere Schleife durchlaufen? $n - 2i + 1$ mal.
- Wie kommen wir jetzt auf die Gesamtlaufzeit?
- $\sum_{i=0}^{n/2}(n - 2i + 1) = \frac{n}{2}n - 2\sum_{i=0}^{n/2} i + \frac{n}{2} = \frac{n^2}{2} + \frac{n}{2} - 2\frac{\frac{n}{2}\cdot\left(\frac{n}{2}+1\right)}{2} = \frac{n^2}{2} + \frac{n}{2} - \frac{n^2}{4} - \frac{n}{2} = \frac{1}{4}n^2$
- Kann man das einfacher machen?

Lukas Bach, lukas.bach@student.kit.edu

Mastertheorem

Karlsruher Institut für Technologie

Formel für Mastertheorem

Rekursive Komplexitätsformeln der Form

$$T(n) = a \cdot T(\tfrac{n}{b}) + f(n)$$

lassen sich mit dem Mastertheorem Komplexitätsklassen zuordnen.

Auflösung des Mastertheorem

Fall 1: Wenn $f \in \mathcal{O}(n^{\log_b a - \varepsilon})$ für ein $\varepsilon > 0$ ist, dann ist $T \in \Theta(n^{\log_b a})$.

Fall 2: Wenn $f \in \Theta(n^{\log_b a})$ ist, dann ist $T \in \Theta(n^{\log_b a} \log n)$.

Fall 3: Wenn $f \in \mathcal{O}(n^{\log_b a + \varepsilon})$ für ein $\varepsilon > 0$ ist, und wenn es eine Konstante d gibt mit $0 < d < 1$, so dass für alle hinreichend großen n gilt $af(n/b) \leq df$, dann ist $T \in \Theta(f)$.

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben zum Mastertheorem

- $T(n) := 2T(\frac{n}{4}) + \sqrt{n}$, also $a = 2, b = 4, f(n) = \sqrt{n}$, also zweiter Fall des Mastertheorems. $T \in \Theta(\sqrt{n} \log n)$
- $T(n) := 3T(\frac{n}{2}) + n \log n$, also $a = 3, b = 2, f(n) = n \log n$, also erster Fall des Mastertheorems, $T \in \Theta(n^{\log_2 3})$
- $T(n) := 4T(\frac{n}{2}) + n^2\sqrt{n}$, also $a = 4, b = 2, f(n) = n^2\sqrt{n}$, also dritter Fall des Mastertheorems, $T \in \Theta(n^2\sqrt{n})$.

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

Lukas Bach, lukas.bach@student.kit.edu

Definition eines endlichen Automaten

Endlicher Automat

Ein endlicher Automat ist ein Tupel $A = (Z, z_0, X, f, Y, g)$ mit...

- endliche Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Eingabealphabet X
- Zustandsübergangsfunktion $f : Z \times X \to Z$
- Ausgabealphabet Y
- Ausgabefunktion
  - Mealy-Automat: $g : Z \times X \to Y^*$
  - Moore-Automat: $h : Z \to Y^*$

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 02.02.2017

Lukas Bach, lukas.bach@student.kit.edu

Automaten

Lukas Bach, lukas.bach@student.kit.edu

Mealy-Automat

Karlsruher Institut für Technologie

Mealy-Automat

Ein Mealy-Automat ist ein Tupel $A = (Z, z_0, X, f, Y, h)$ mit...

- endliche Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Eingabealphabet X
- Zustandsübergangsfunktion $f : Z \times X \to Z$
- Ausgabealphabet Y
- Ausgabefunktion $h : Z \times X \to Y^*$

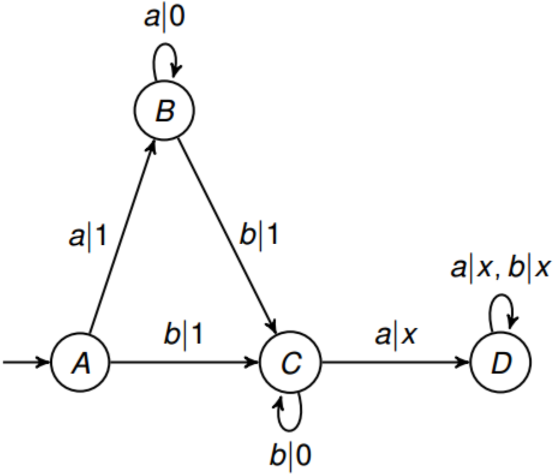
Darstellung als Graph

- Zustände $\to$ Knoten
- Startzustand $\to$ Pfeil an diesen Knoten (ohne Anfang)
- Zustandsüberführungsfunktion $\to$ Kanten mit Beschriftung
- Ausgabefunktion $\to$ zusätzliche Kantenbeschriftung

Grundbegriffe der Informatik

Lukas Bach, lukas.bach@student.kit.edu

Beispiel Mealy-Automat



Lukas Bach, lukas.bach@student.kit.edu

Moore-Automat

Karlsruher Institut für Technologie

Moore-Automat

Ein Moore-Automat ist ein Tupel $A = (Z, z_0, X, f, Y, h)$ mit...

- endliche Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Eingabealphabet X
- Zustandsübergangsfunktion $f : Z \times X \to Z$
- Ausgabealphabet Y

$\to$ Bis hierhin alles wie bei Mealy!

- Ausgabefunktion $h : Z \to Y^*$

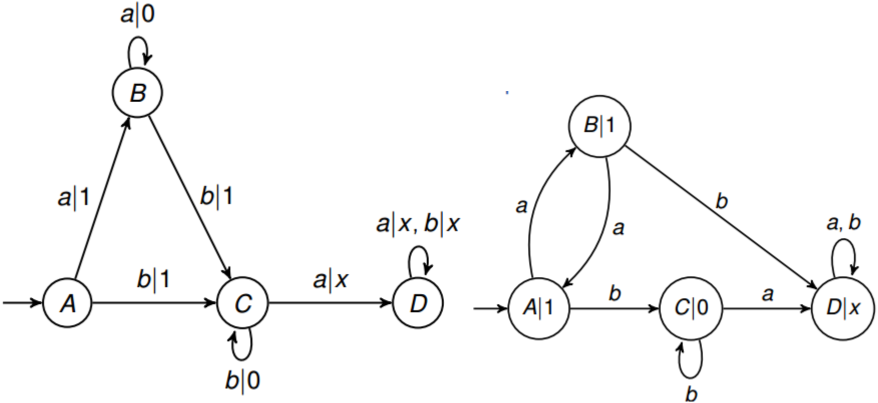
Bemerkung

Für jeden Mealy-Automaten kann man einen Moore-Automaten konstruieren, der genau die gleiche Aufgabe erfüllt, und umgekehrt.

Lukas Bach, lukas.bach@student.kit.edu

Umwandlung Mealy- in Moore-Automat

Links ein Mealy-, rechts ein Moore-Automat



Aufgabe

Wie sieht der Mealy-Automat als äquivalenter Moore-Automat aus, wie sieht der Moore-Automat als äquivalenter Mealy-Automat aus?

Lukas Bach, lukas.bach@student.kit.edu

Endliche Akzeptoren

- Sonderfall von Moore-Automaten
- Bei einem Akzeptor will man nur wissen, ob die Eingabe akzeptiert wurde oder nicht (also reicht ein Bit als Ausgabealphabet)
- Statt der Ausgabefunktion h schreibt man einfach die Menge der akzeptierenden Zustände $F \subseteq Z$ auf
- Zustände, die nicht akzeptieren, heißen ablehnend
- Im Graphen werden akzeptierende Zustände einfach mit einem doppelten Kringel gekennzeichnet 

Lukas Bach, lukas.bach@student.kit.edu

Akzeptierte Wörter und Sprachen

Karlsruher Institut für Technologie

Akzeptierte Wörter

Ein Wort $w \in X^*$ wird vom endlichen Akzeptor akzeptiert, wenn man ausgehend vom Anfangszustand bei Eingabe von w in einem akzeptierenden Zustand endet.

Bemerkung

- Wird ein Wort nicht akzeptiert, dann wurde es abgelehnt

Akzeptierte formale Sprache

Die von einem Akzeptor A akzeptierte formale Sprache $L(A)$ ist die Menge aller von ihm akzeptierten Wörter.

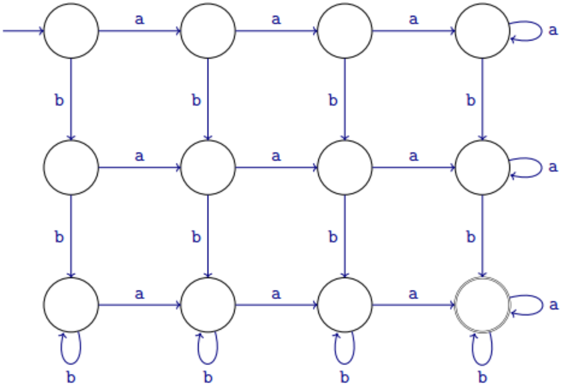
Lukas Bach, lukas.bach@student.kit.edu

Endliche Akzeptoren

Aufgabe zu endlichen Akzeptoren

Konstruiere einen endlichen Akzeptor, der die Sprache $L_1(A) = \{w \in \{a, b\}^* : (N_a(w) \geq 3 \wedge N_b(w) \geq (2)\}$ erkennt.

Lösung



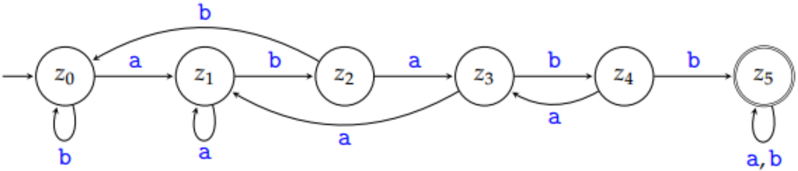
Lukas Bach, lukas.bach@student.kit.edu

Endliche Akzeptoren

Aufgabe zu endlichen Akzeptoren

Konstruiere einen endlichen Akzeptor, der die Sprache
$L_2(A) = \{w_1 ababbw_2 | w_1, w_2 \in \{a, b\}^*\}$ erkennt.

Lösung



Aufgabe

Konstuiere einen endlichen Akzeptor der die Sprache
$L_3 = \{w \in \{a, b\}^* | w \notin L_2\}$ akzeptiert.

Lösung

Ablehnende Zustände wereden zu akzeptierenden und andersrum.

Lukas Bach, lukas.bach@student.kit.edu

Endliche Akzeptoren

Aufgaben zu endlichen Akzeptoren

- Gebe für den unten stehenden Automaten an, welche Sprache dieser akzeptiert.
- Gebe für die folgende Sprache über dem Alphabet $\{a, b\}$ einen endlichen Akzeptor an: $L = \{w \in \Sigma^* | N_a(w) \bmod 3 > N_b(w) \bmod 2\}$

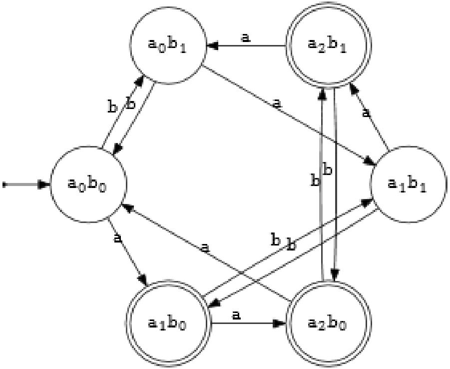
Lukas Bach, lukas.bach@student.kit.edu

Lösungen

Lösung 1

$L = \{w \in \Sigma^* \mid |w| \bmod 2 = 1\}$ (Worte ungerader Länger)

Lösung 2



Lukas Bach, lukas.bach@student.kit.edu

Endliche Akzeptoren

Wann wird das leere Wort ε von einem endlichen Akzeptor akzeptiert?

$\varepsilon \in L(A)$ gilt genau dann, wenn der Startzustand akzeptiert wird.

Lukas Bach, lu-
kas.bach@student.kit.edu

Reguläre Ausdrücke

Lukas Bach, lukas.bach@student.kit.edu

Regulärer Ausdruck

Karlsruher Institut für Technologie

Regulärer Ausdruck

- Alphabet $Z = \{|, (,), *, \emptyset\}$ von "Hilfssymbolen"
- Alphabet A enthalten keine Zeichen aus Z
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften
  - $\emptyset$ ist ein RA
  - Für jedes $x \in A$ ist x ein RA
  - Wenn R_1 und R_2 RA sind, dann auch $(R_1 | R_2)$ und $(R_1 R_2)$
  - Wenn R ein RA ist, dann auch $(R*)$

Lukas Bach, lukas.bach@student.kit.edu

Klammerregeln

- "Stern- vor Punktrechnung"
- "Punkt- vor Strichrechnung"

$\longrightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\longrightarrow$ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Aufgabe

Entferne so viele Klammern wie möglich, ohne die Bedeutung des RA zu verändern.

- $(((((ab)b)*)*)|(\emptyset*)) \longrightarrow (abb)**|\emptyset*$
- $((a(a|b))|b) \longrightarrow a(a|b)|b$

Lukas Bach, lukas.bach@student.kit.edu

Alternative Definition

Wir können die Syntax von regulären Ausdrücken auch über eine kontextfreie Grammatik definieren.

Aufgabe

Vervollständigt die folgende Grammatik.

$G = (\{R\}, \{|, (,), *, \emptyset\} \cup A, R, P)$
mit $P = \{R \to \emptyset, R \to x$ (mit $x \in A$),
$R \to (R|R), R \to (RR)$,
$R \to (R*)$
$R \to \varepsilon\}$

Wieso brauchen wir ε?

Lukas Bach, lukas.bach@student.kit.edu

Durch R beschriebene Sprache

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle = \{\}$
- $\langle x \rangle = \{x\}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_1 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$
- $\langle R* \rangle = \langle R \rangle *$

Lukas Bach, lukas.bach@student.kit.edu

Charakterisierung regulärer Sprachen

Satz

Für jede formale Sprache L sind äquivalent:

1. L kann von einem endlichen Akzeptor erkannt werden.
2. L kann durch einen regulären Ausdruck beschrieben werden
3. L kann von einer rechtslinearen Grammatik erzeugt werden.

Solche Sprachen heißten regulär.

Anwendung von regulären Ausdrücken

Lukas Bach, lukas.bach@student.kit.edu

Zum selbst probieren:
http://regexr.com/

Achtung: Reguläre Ausdrücke in praktischer Programmierung funktionieren zwar ähnlich, haben aber eine andere Syntax und können teils mehr!

Lukas Bach, lukas.bach@student.kit.edu

Rechtslineare Grammatiken

Lukas Bach, lukas.bach@student.kit.edu

Rechtslineare Grammatiken

Karlsruher Institut für Technologie

Definition

Eine rechtslineare Grammatik ist eine reguläre Grammatik $G = (N, T, S, P)$ mit der Einschränkung, dass alle Produktionen die folgende Form haben:

- $X \to w$ mit $w \in T^*$ oder
- $x \to wY$ mit $w \in T^*$, $Y \in N$

Lukas Bach, lukas.bach@student.kit.edu

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0,1\}^* | \exists k \in \mathbb{N}_0 : Num_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Lösung

- $R = (0*10)|(0*1(0)*1) = 0*10|0*10*1$
- $G = (\{S, A\}, \{0, 1\}, S, \{S \to 0S|10|1A, A \to 0A|1\})$

Grundbegriffe der Informatik

Lukas Bach, lukas.bach@student.kit.edu

Tutorium vom 09.02.2017

Lukas Bach, lu-
kas.bach@student.kit.edu

Turingmaschinen

Lukas Bach, lukas.bach@student.kit.edu

Turingmaschinen

Was sind Turingmaschinen?

- Sehr mächtige Erweiterung Automat
  - Was heißt mächtig?
  - Turingmaschinen können eine große Vielfalt von Problemen lösen, einschließlich vieler in GBI besprochener Probleme
- Gesteuert durch einen endlichen Automaten, aber mit einem unendlichen Arbeitsband zum Zwischenspeichern von Informationen
- Besitzen einen Kopf um auf dem Band zu lesen und zu schreiben
- Turingmaschinen sind sozusagen genauso mächtig wie Computer
  - können also benutzt werden, um für Probleme zu entscheiden, ob sie gelöst werden können oder nicht

Lukas Bach, lukas.bach@student.kit.edu

Definition von Turingmaschinen

Karlsruher Institut für Technologie

Definition von Turingmschinen

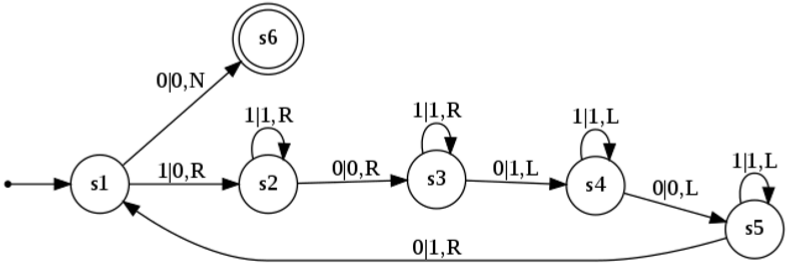
Eine Turingmaschine $T = (Z, z_0, X, f, g, m)$ besteht aus:

- Z Zustandsmenge
- $z_0 \in Z$ Startzustand
- X Bandalphabet
- $\Box$ Blanksymbol (sozusagen Markierung für Leerzeichen)
- $f : Z \times X \longrightarrow Z$ partielle Zustandsübergangsfunktion
- $g : Z \times X \longrightarrow X$ partielle Ausgabefunktion
- $m : Z \times X \longrightarrow \{L, N, R\}$ partielle Bewegungsfunktion

Anmerkung: partielle Funktionen sind nicht linkstotal, also manche Elemente des Definitionsbereichs werden nicht abgebildet.

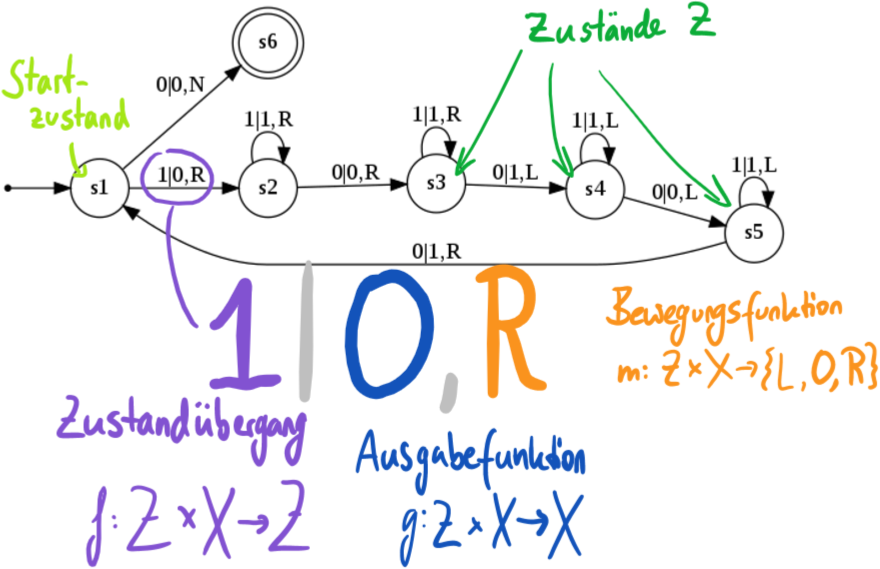
Lukas Bach, lukas.bach@student.kit.edu

Beispiel einer Turingmaschine



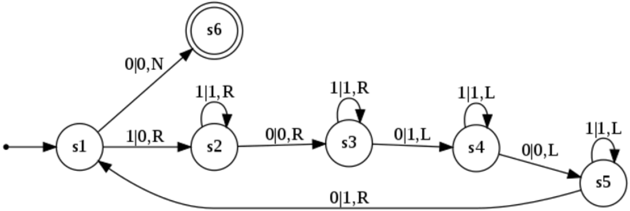
Lukas Bach, lukas.bach@student.kit.edu

Beispiel einer Turingmaschine



Funktionen von Turing Maschinen

Lukas Bach, lukas.bach@student.kit.edu



Wie sehen die konkreten Abbildungsvorschriften der linken vier Pfeile aus?

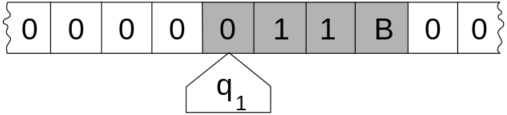
- $f : Z \times X \to Z$
- $g : Z \times X \to X$
- $m : Z \times X \to \{L, N, R\}$

	f	g	m
$0\|0, N$	$(s1, 0) \mapsto s6$	$(s1, 0) \mapsto 0$	$(s1, 0) \mapsto N$
$1\|0, R$	$(s1, 1) \mapsto s2$	$(s1, 1) \mapsto 0$	$(s1, 1) \mapsto R$
$1\|1, R$	$(s2, 1) \mapsto s2$	$(s2, 1) \mapsto 1$	$(s2, 1) \mapsto R$
$0\|0, R$	$(s2, 1) \mapsto s3$	$(s2, 1) \mapsto 0$	$(s2, 1) \mapsto R$

Lukas Bach, lukas.bach@student.kit.edu

Das Band einer Turingmaschine

- Unendliche Anreihung von Zeichen, die nach links und rechts unbegrenzt weiter geht



- Die Turingmaschine hat einen Kopf, mit dem sie das aktuelle Zeichen lesen oder überschreiben kann, oder kann ihn nach links oder rechts bewegen.
- Das Band einer Turingmaschine wird benutzt als...
  - Erhalten der Eingabe: Bevor die Turingmaschine startet, steht das Eingabewort auf dem Band, der Kopf steht auf dem ersten Zeichen der Eingabe.
  - Rückgabe der Ausgabe: Nach Beenden steht auf dem Band die Ausgabe (und der Kopf irgendwo).
  - Zwischenspeicher: Die Turingmaschine kann überall Informationen zwischenspeichern, diese müssen von der TM am Ende aber gelöscht werden.

Lukas Bach, lukas.bach@student.kit.edu

Beispielabarbeitungen

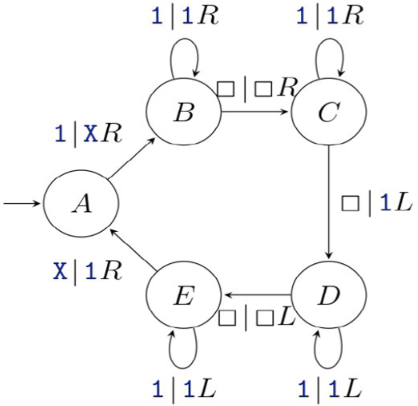
Karlsruher Institut für Technologie

Gemeinsame Übung

Arbeite folgende Wörter mit der Turingmaschine ab:

- 0
- 1
- 11
- 111

Was macht die Turingmaschine?



Die Turingmaschine macht aus 1^k die Ausgabe $1^k\square 1^k$.

Lukas Bach, lukas.bach@student.kit.edu

Halten von Turingmaschinen

Halten einer Turingmaschine

Wenn eine Turingmaschine in einem Zustand ist, für den das nächste Eingabezeichen durch die Übergangsfunktion f nicht definiert ist, hält die Maschine.

Wann hält also eine Turingmaschine nicht?

Nicht-Halten einer Turingmmaschine

Wenn eine Turingmaschine in eine endlose Schleife gerät, so hält sie nicht.

Lukas Bach, lukas.bach@student.kit.edu

Entscheidbarkeit

Karlsruher Institut für Technologie

Durch Turingmaschine akzeptierte Sprache

Eine Turingmaschine akzeptiert eine formale Sprache L, wenn sie für jedes Wort $w \in L$ in einem akzeptierenden Zustand hält.

Entscheidbare Sprache

Eine formale Sprache L heißt entscheidbar, wenn es eine Turingmaschine gibt, die immer hält und L akzeptiert.

Aufzählbare Sprache

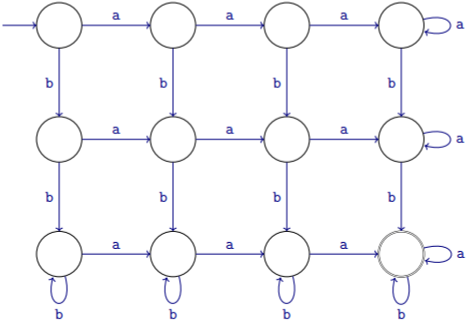
Eine formale Sprache L heißt aufzählbar, wenn es eine Turingmaschine gibt, die L akzeptiert.

Lukas Bach, lukas.bach@student.kit.edu

Vom endlichen Akzeptor zur Turingmaschine

Akzeptieren von Turingmaschinen

Wie kann man aus dem gegebenen endlichen Akzeptor eine Turingmaschine machen, die dieselbe Sprache akzeptiert?



Lukas Bach, lukas.bach@student.kit.edu

Lösung

Karlsruher Institut für Technologie

Einfach gesagt: mache aus jedem Übergang a einen Turingmaschinen-Übergang der Art $a|a, R$, also bei jedem Zeichen mache den Zustandsübergang, überschreibe aber das Zeichen nicht und gehe zum nächsten Zeichen.

Formaler ausgedrückt?

- Für allgemeinen endlichen Akzeptor (Z, z_0, X, f, Y, h), definiere eine Turingmaschine $T := (Z, z_0, X \cup Y, f, g, h)$, also Z, z_0, f gleich und mit Bandalphabet $=$ Eingabealphabet $\cup$ Ausgabealphabet
- $g(z, x) := x \quad \forall (z, x)$ in f definiert
- $m(z, x) := R \quad \forall (z, y)$ in f definiert

Jeder endliche Akzeptor kann so zu einer Turingmaschine umgeformt werden, die dieselbe Sprache akzeptiert.

Lukas Bach, lukas.bach@student.kit.edu

Über endliche Akzeptoren hinaus

Karlsruher Institut für Technologie

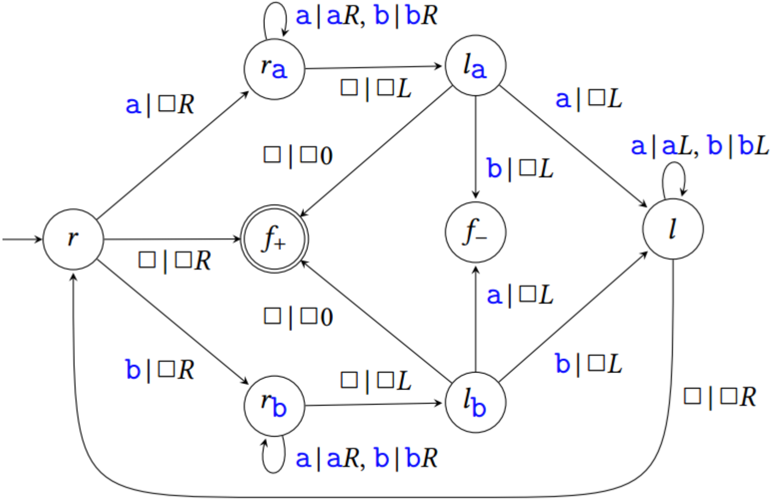
Sei L die Sprache von Palindromen über $\{a, b\}$ ($L = \{aabaa, bbababb, aa, \varepsilon\}$).

- Ist die Sprache regulär, also gibt es einen endlichen Akzeptor, der diese akzeptiert? Nein.
- Ist die Sprache entscheidbar, also gibt es eine stets haltende Turingmaschine, die L akzeptiert?

Lukas Bach, lukas.bach@student.kit.edu

Palindromerkennung mit Turingmaschinen

Ja, nämlich:



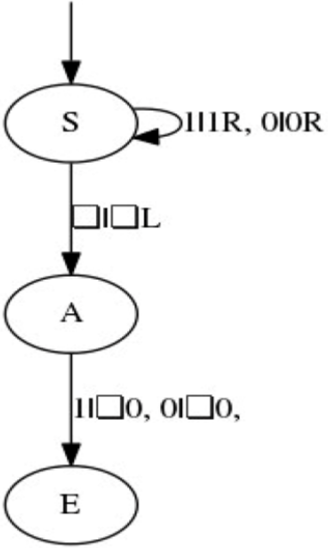
Turingmaschinen Entwurfsaufgabe

Lukas Bach, lukas.bach@student.kit.edu

Turingmaschine Entwurf

Entwerfe eine Turingmaschine, die...

- als Eingabe eine Binärzahl auf dem Band erhält
- als Ausgabe diese Zahl restlos durch zwei teilt und auf dem Band stehen lässt



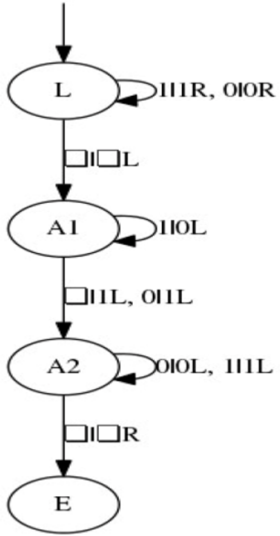
Turingmaschinen Entwurfsaufgabe

Lukas Bach, lukas.bach@student.kit.edu

Turingmaschine Entwurf

Entwerfe eine Turingmaschine, die...

- als Eingabe eine Binärzahl auf dem Band erhält
- als Ausgabe diese Zahl um eins erhöht auf dem Band stehen lässt
- den Kopf der Turingmaschine auf dem ersten Zeichen der Ausgabe stehen hat.

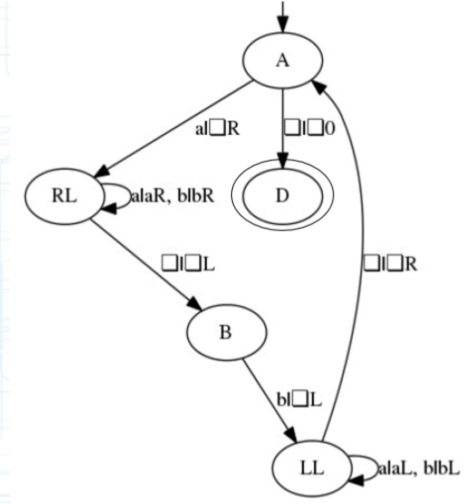


Lukas Bach, lukas.bach@student.kit.edu

Turingmaschine Entwurfsaufgabe

Turingmaschine Entwurf

Entwerfe eine Turingmaschine, die die Sprache $\{a^k b^k : k \in \mathbb{N}_0\}$ erkennt.



Lukas Bach, lukas.bach@student.kit.edu

Konfiguration von Turingmaschinen

Angenommen, man kennt eine Turingmaschine, hat mit der Abarbeitung eines Wortes angefangen, will aber pausieren, um später weiterzumachen...

Was muss man sich alles merken, um später weiter zu machen?

- Derzeitiger Zustand, in dem die Turingmaschine steht
- Inhalt des Bandes

Konfiguration von Turingmaschinen

Wenn während dem Arbeiten einer Turingmaschine auf dem Band das Wort $w_1 a w_2$ mit $w_1, w_2 \in X^*, a \in X$ steht, der Kopf der Turingmaschine auf das Zeichen a zeigt und die Turingmaschine im Zustand z ist, so schreibt man die Konfiguration der Turingmaschine als $\square w_1 (z) a w_2 \square$.

Lukas Bach, lukas.bach@student.kit.edu

Konfiguration von Turingmaschinen

Beispiel:

$$\Box\Box\Box\Box\Box\Box\Box\Box abcbabb\,d\,aabc\Box\Box\Box\Box\Box$$

$$\uparrow$$

KOPF

...sei das Band der Turingmaschine während Abarbeitung der Eingabe, dazu steht sie im Zustand z_4.

Dann sieht sieht die Konfiguration der Turingmaschine so aus:
$\Box abcbabb(z_4)daabc\Box$

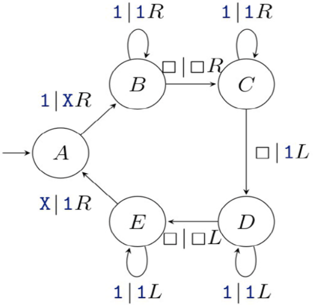
Lukas Bach, lukas.bach@student.kit.edu

Dokumentation einer Abarbeitung mit Konfigurationen

Karlsruher Institut für Technologie

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.



$\square(A)11\square$

$\rightarrow \square X(B)1\square$

$\rightarrow \square X1(B)\square$

$\rightarrow \square X1\square(C)\square$

$\rightarrow \square X1(D)\square 1\square$

$\rightarrow \square X(E)1\square 1\square$

$\rightarrow \square(E)X1\square 1\square$

$\rightarrow \square 1(A)1\square 1\square$

$\rightarrow \square 1X(B)\square 1\square$

$\rightarrow \square 1X\square(C)1\square$

$\rightarrow \square 1X\square 1(C)\square$

$\rightarrow \square 1X\square(D)11\square$

$\rightarrow \square 1X(D)\square 11\square$

$\rightarrow \square 1(E)X\square 11\square$

$\rightarrow \square 11(A)\square 11\square$

Lukas Bach, lukas.bach@student.kit.edu

Halteproblem

Karlsruher Institut für Technologie

Halteproblem: Für einen gegebenen Algorithmus, gelingt dieser bei seiner Abarbeitung zu einem Ende und hält?

- Algorithmen können durch Turingmaschinen durchgeführt werden
- Turingmaschinen können durch sogenannte universelle Turingmaschinen simuliert werden
  - Wenn eine Turingmaschine T kodiert ist mit dem Wort w, dann ist $T_w : X \longrightarrow X$ eine Funktion, die Eingaben auf die Ausgabe der Turingmaschine T mappt.
  - Also mit $X = \{1, 0\}$ gibt z.B. $T_w(100101) = 001$ genau dann zurück, wenn, sofern man 100101 als Eingabe an die Turingmaschine mit der Kodierung w gibt, diese hält und als Ausgabe 001 erzeugt.

Dann lässt sich das Halteproblem auch als Sprache formulieren:

$$H = \{w \in A^* : w \text{ ist eine TM-Codierung und } T_w(w) \text{ hält.}\}$$

bzw. als allgemeinerer Fall:

$$\widehat{H} = \{(w, x) \in A^* \times A^* : w \text{ ist eine TM-Codierung und } T_w(x) \text{ hält.}\}$$

Lukas Bach, lukas.bach@student.kit.edu

Halteproblem

Das Halteproblem ist unentscheidbar, dh. es gibt keine Turingmaschine, die H entscheidet.

Lukas Bach, lukas.bach@student.kit.edu

Busy Beaver

Karlsruher Institut für Technologie

Busy Beaver TM ist eine Turingmaschine mit n Zuständen, die möglichst viele Einsen auf das Band schreibt und hält.

- Also nicht einfach ewig Einsen aufschreibt und nie aufhört.

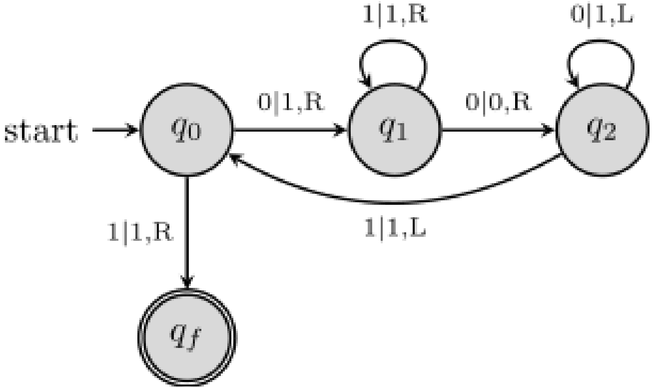
Busy Beaver Problem: Für eine gegebene Turingmaschine mit n Zuständen, die Einsen aufschreibt und hält: Schreibt sie auch maximal viele Einsen auf?

Das Busy Beaver Problem ist nicht entscheidbar, bzw. die Busy Beaver Funktion $bb(n)$, die definiert, wieviele einsen von einer Busy Beaver TM maximal geschrieben werden können, ist nicht berechenbar.

Beispielwerte von bb:
$bb(1) = 1, bb(2) = 4, bb(5) \geq 4098, bb(6) > 10^{18276}$.

Busy Beaver für $n = 3$

Lukas Bach, lukas.bach@student.kit.edu



Lukas Bach, lukas.bach@student.kit.edu

Organisatorisches

Karlsruher Institut für Technologie

- Alle Folien und Folienpaket jetzt online.
- Fragen zur Klausur oder zur Vorbereitung?

Lukas Bach, lukas.bach@student.kit.edu

Informationen

Zum Tutorium

- Lukas Bach
- Tutorienfolien auf:
  - `http://gbi.lukasbach.com`
- Tutorium findet statt:
  - Donnerstags, 14:00 - 15:30
  - 50.34 Informatikbau, -107

Mehr Material

- Ehemalige GBI Webseite:
  - `http://gbi.ira.uka.de`
  - Altklausuren!

Zur Veranstaltung

- Grundbegriffe der Informatik
- Klausurtermin:
  - 06.03.2017, 11:00
  - Zwei Stunden Bearbeitungszeit
  - 6 ECTS für Informatiker und Informationswirte, 4 ECTS für Mathematiker und Physiker

Zum Übungsschein

- Übungsblatt jede Woche
- Ab 50% insgesamt hat man den Übungsschein
- Keine Voraussetzung für die Klausur, aber für das Modul