Informatik I

Lutz Donnerhacke
lutz@iks-jena.de

PGP:db089309
1c1c 6311 ef09 d819 e029 65be bfb6 c9cb

IKS

Semesterübersicht

- Grundbegriffe der theoretischen Informatik
- Übersicht über Funktionen von Betriebsystemen
  - Praxisüberungen mit einem Unix
- Programmiersprachen
  - Programmierpraxis mit C
- Algorithmen und Datenstrukturen
- ✔ Schriftliche Klausur

Übersicht „Theoretische Informatik“

- Automatentheorie
  - Reguläre Ausdrücke und endliche Automaten
  - Kontextfreie Grammatik und Stackmaschinen
- Berechenbarkeit
  - Turing-, Registermaschinen und λ-Kalkül
- Komplexitätstheorie
  - Zeitliche und räumliche Komplexitätsklassen

Übersicht Betriebsysteme

- Trennung in administrative Bereiche
- Ressourcenverwaltung
- Prozeßtrennung
- Hardwarevirtualisierung
- Ereignisgesteuerte Verarbeitung
- Harte und weiche Echtzeit
- Unterstützung von Standardprotokollen

Übersicht Programmiersprachen

- Maschinen- und verschiedene Hochsprachen
- Typen, starke und schwache Typisierung
- Grammatik und Semantik
- Standardbibliotheken
- Speicherverwaltung
- Debugging
- ➔ Lehrsprache ist Haskell, Prüfungssprache ist C

Übersicht (1)
Algorithmen und Datenstrukturen

- Elementares: Array, Liste, Stack, Queue
- Bäume: Binärbäume, rekursive Traversierung
- Sortieren: Selection, Insertion, Bubble, Shell, Quick, Digital, Heap, Merge, große Datenmengen
- Suchen: Linear, Binär, Rot-Schwarz-Bäume, Hashing, Tries, Patricia, B-Bäume
- Stringsuche: Boyer-Moore, Pattern-Matching, Parsing, Huffman

Übersicht (2)
Algorithmen und Datenstrukturen

- Geometrie: Punkte, Strecken, Polygone, Hüllen, Schnitte, Voronoi-Diagramme
- Graphen: Traversierung von Labyrinthen, Zusammenhang, Spannbaum, kürzester Pfad, topologische Sortierung, Flußsteuerung

- ✗ Algorithmen werden kurz vorgestellt
- ✗ Implementation im Selbststudium
- ✗ Was nicht zu schaffen ist, wird weggelassen.

Ablauf Selbststudium

- Aufgabenstellungen werden ausgeteilt
- Fertige Compilierumgebung auf Unix
  - Benötigt Haskell- und C-Compiler
- Lösungen bis Ende der zweiten Woche einsenden
- Rückfragen per E-Mail
- Lösungen werden nicht bewertet, es gibt Hinweise

➔ Entscheidend ist die Klausur

Literatur

- Michael Sipser: Introduction to the Theory of Computation; PWS Publishing Company; 1997
- Bryan O'Sullivan, Don Stewart, and John Goerzen: Real World Haskell; O'Reilly; 2008
- Kernighan/Ritchie: Programmieren in C; Carl Hanser Verlag; 1990
- Robert Sedgwick: Algorithmen; Addision-Wesley
- Donald E. Knuth: The Art of Computer Programming; Addison-Wesley; 1968-2015

Netzressourcen

- http://de.wikipedia.org/ Begriffsüberblicke
- http://www.haskell.org/ Doku und Compiler
- news:///de.comp.lang.*/ Deutschsprachige Diskussionen
- Suchmaschinen
- Bauen Sie sich ein eigenes Wiki!

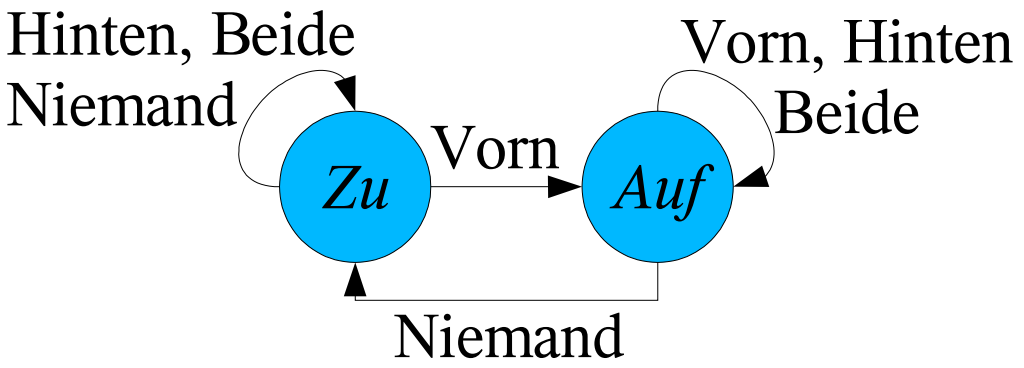
Theoretische Informatik

Automatentheorie

Was ist eigentlich ein Computer?

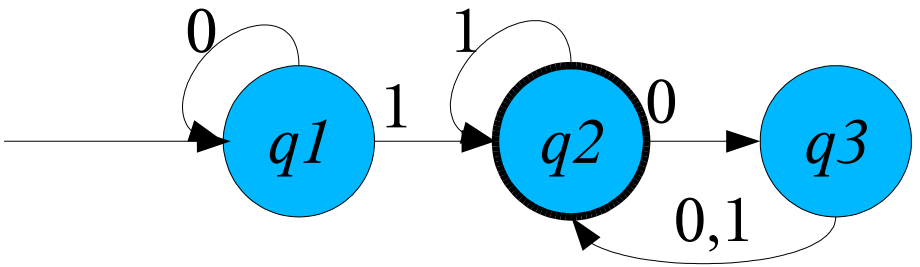
Endliche Automaten

- **Modell** zur Erklärung und Beweisführung
- Exakt definiert, Grundlage von Beweisen
- Modell einer Tür



Endliche Automaten

- **Status Diagramm**: Start, Ende, Übergänge
- Wenn am Ende der Eingabe im Endzustand: Ok
- Alle andern Zustände am Ende: Abweisung
- Welche Zeichenfolgen akzeptiert dieser Automat?



Endliche Automaten

- **Definition**: Ein *endlicher Automat* (DFA) ist das Tupel $(Q, \Sigma, \delta, s, F)$:
  1. Q ist eine endliche Menge von Zuständen
  2. Σ ist eine endliche Menge, das Alphabet
  3. δ: $Q \times \Sigma \rightarrow Q$ ist die Übergangsfunktion
  4. Das Element s aus Q ist der Startzustand
  5. Die Teilmenge F aus Q sind Endzustände

Endliche Automaten

- **Definition**: Sei w aus Σ^n ein String über dem Alphabet. Dann wird w von einem endlichen Automaten M *akzeptiert*, genau dann wenn:

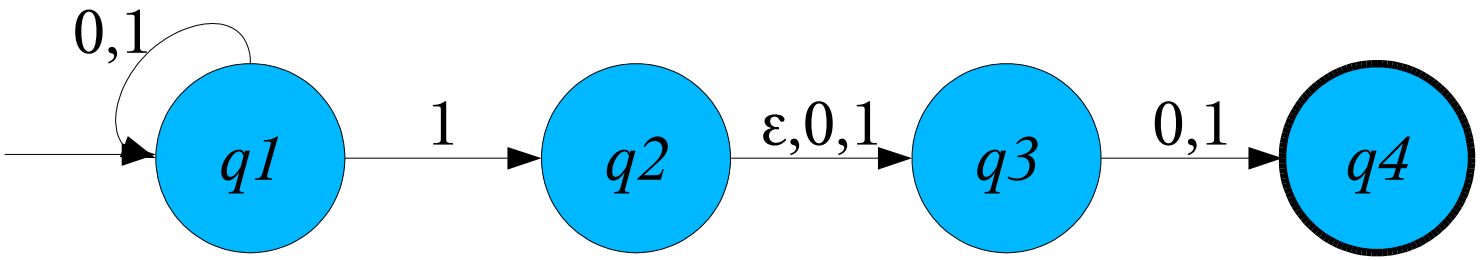
  1. $r(0) = q$
  2. $r(i) = \delta(r(i-1), w(i))$ für $i = 1 .. n$
  3. $r(n)$ in F

- **Definition**: M akzeptiert die *Sprache*
  [w | M akzeptiert w]
- **Definition**: Eine Sprache heißt *regulär*, wenn sie von einem endlichen Automaten akzeptiert wird.

IKS

Nichtdeterministische Endliche Automaten

- Nichtdeterministischer endlicher Automat (NFA) hat Übergangsabbildung (nicht Funktion)
- Zusätzlich auch ε-Übergang (ohne Eingabe)



Nichtdeterministische Endliche Automaten

- NFA und DFA sind äquivalent
- Beweisidee: Eine Abbildung ist eine Funktion in die Potenzmenge.
  - Zustände des DFA sind Zustandskombinationen
- n-Zustand NFA => 2^n-Zustand DFA
- Zustandsbereinigung oft sehr effektiv

Reguläre Sprachen

- Abschluß von regulären Sprachen unter:
  - Alternative: A|B = [x | x in A oder y in B]
  - Verkettung: AB = [xy | x in A und y in B]
  - Stern: A* = [x1,x2,..xk | k ≥ 0 und x in A]
  - Komplement: ^A = [x | x in Σ\A]

Reguläre Ausdrücke

- Kompakte Form eines Suchausducks
  - *a* - paßt genau auf das Zeichen des Alphabets
  - ^ - Beginn der Eingabe
  - *$* - Ende der Eingabe
  - *A*|*B* - paßt auf einen der Ausdrücke A oder B
  - *AB* - paßt erst auf A und der Rest auf B
  - *A** - paßt auf den Ausdruck A beliebig oft
  - *(A)* - paßt auf den Ausdruck A

Reguläre Ausdrücke

- Nützliche Zusatzdefinitionen
  - *[ab]* = a|b (zusätzlich *[g-j]* = [ghij])
  - *A+* = AA*
  - *A{x,y}* – A^n mit $x \leq n \leq y$ (x oder y kann entfallen)
  - *A?* = (|A) = A{0,1}
  - . - beliebiges Zeichen des Alphabets
- Unter Unix die Suchmuster von egrep

Reguläre Ausdrücke

- Reguläre Ausdrücke akzeptieren reguläre Sprachen
- NFA aus regulärem Ausdruck direkt konstruierbar
- Regulärer Ausdruck aus NFA durch schrittweises Entfernen von Knoten konstruierbar: Übergänge über den Knoten werden als Alternative von Verkettungen zu einem neuen Übergang zwischen den Restknoten.
- Reguläre Ausdrücke sind ohne Speicher implementierbar, also nur ein Suchprogramm

Pumping Lemma

- **Pumping Lemma**: Jeder String einer regulären Sprache ab einer festen sprachabhängigen Länge p läßt sich die Form xyz zerlegen, wobei:
  1. y ist nicht leer.
  2. xy ist kürzer als p.
  3. Für jedes $n \geq 0$ gehört xy^nz zur Sprache.
- Idee des Lemmas: y umfaßt die Größe des Automaten, dann muß sich alles wiederholen.
- Besonders nützlich, um Nichtregularität zu zeigen.

Pumping Lemma

- Prüfe, ob [0*1* | n > 0] regulär ist.
- Prüfe, ob ε = [] regulär ist.
- Prüfe, ob [0^n1^n | n > 0] regulär ist.
- Prüfe, ob [$0^n1^n2^n$ | n > 0] regulär ist.
- Prüfe, ob [w | w aus [0,1]* und w enthält genausoviel 0en wie 1en] regulär ist.
- Prüfe, ob [ww | w aus [0,1]*] regulär ist.

Kontextfreie Sprachen

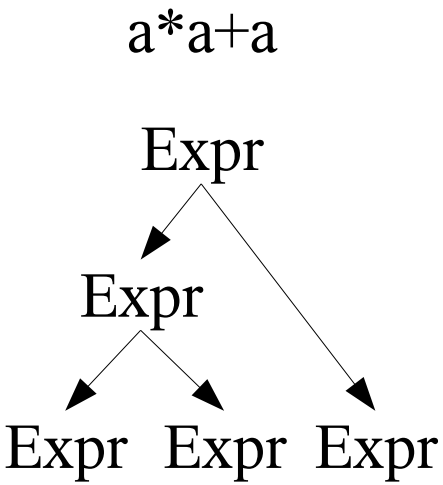
- **Definition**: Eine *kontextfreie Grammatik* ist das Tupel (V, Σ, R, S):
  1. V ist ein endliche Menge von Variablen
  2. Σ ist ein endliche Menge von Terminalen ≠ V
  3. R ist eine endliche Menge von Ersetzungsregeln, die jede einer Variablen eine Verkettung von Variablen und Terminalen zuweist
  4. S ist die Startvariable aus V
- Sie akzeptieren kontextfreie Sprachen

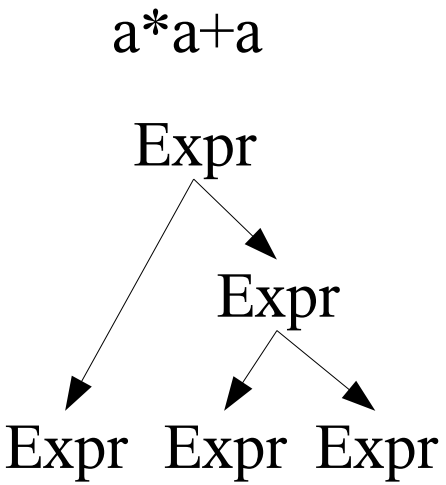
Kontextfreie Sprachen

- Beispiel [$0^n1^n \mid n > 0$] hat folgende Grammatik:
  - S = 01|0S1
- Was beschreibt diese Grammatik?
  - Σ = [a,+,*,(,)] / V = [Expr, Term, Factor] / S = Expr
  - Expr = Expr + Term | Term
  - Term = Term * Factor | Factor
  - Factor = (Expr) | a

Kontextfreie Sprachen

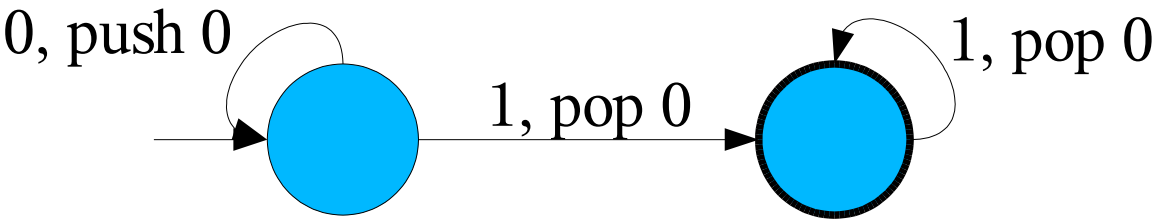
- Kontextfreie Sprachen leiden an *Mehrdeutigkeit*
- Expr = Expr * Expr | Expr + Expr | (Expr) | a





Stackautomat

- Endlicher Automat mit einem Stack
- *Push* – Auf den Stack legen
- *Pop* – Vom Stack als extra Eingabe lesen
- Benötigt zusätzlichen Speicher
- Stackautomaten akzeptieren kontextfreie Sprachen



Pumping Lemma

- **Pumping Lemma**: Jeder String einer kontextfreien Sprache ab einer festen sprachabhängigen Länge p läßt sich die Form vwxyz zerlegen, wobei:
  1. wy ist nicht leer.
  2. wxy ist kürzer als p.
  3. Für jedes $n \geq 0$ gehört vw^nxy^nz zur Sprache.
- Idee des Lemmas: w,y sind Grammatikteile, die keinen höheren Zusammenhang (v,z - Kontext) kennen und deswegen beliebig geschachtelt sind.

Pumping Lemma

- Prüfe, ob [0*1* | n > 0] contextfrei ist.
- Prüfe, ob ε = [] contextfrei ist.
- Prüfe, ob [0^n1^n | n > 0] contextfrei ist.
- Prüfe, ob [$0^n1^n2^n$ | n > 0] contextfrei ist.
- Prüfe, ob [w | w aus [0,1]* und w enthält genausoviel 0en wie 1en] contextfrei ist.
- Prüfe, ob [ww | w aus [0,1]*] contextfrei ist.