I.2 Grundlagen von Programmiersprachen

Mittwoch, 19. Oktober 2016 09:00

Algorithmus:
Idee der Lösungs-
beschreibung

Programm:
Formulierung des Alg.
in einer konkreten (Programmier-) Sprache

Eigenschaften eines Algorithmus:

- in endlichem Text beschreibbar
- effektiv (durch eine Maschine ausführbar)
- deterministisch: besteht aus Elementaroperationen
  u. es liegt eindeutig fest, welche Elementarop.
  als nächstes ausgeführt wird
- determiniert: jeder Eingabe ist genau eine
  Ausgabe zugeordnet

Anzahl und Ausführungszeit der Elementaroperationen
ist beschränkt.

Algorithmus terminiert, wenn er immer nach endlich

vielen Schritten abbricht.

Auf die Eigenschaften Determinismus, Determiniertheit, Terminierung wird manchmal verzichtet.

1. Alg: Deterministisch
   (und daher determiniert)

2. Alg: Indeterministisch,
   aber trotzdem
   determiniert.

Weitere Eigenschaften eines guten Algorithmus:

- Allgemeinheit: Alg. sollte mögl. eine ganze Klasse von Problemen lösen.
- Änderbarkeit: sollte leicht modifizierbar für veränderte Aufgabenstellungen sein
- Effizienz: sollte möglichst wenig Zeit und sonstige Ressourcen verbrauchen
- Robustheit: sollte sich bei unzulässigen Eingaben wohldefiniert verhalten

Bsp für Syntax + Semantik:

Sprache der natürlichen Zahlen ohne 0.

Syntax: Folge von Ziffern $(0, 1, \ldots, 9)$, wobei die erste Ziffer $\neq 0$ ist.

z.B: 30243

Semantik: Wert einer Ziffer ist die Ziffer selbst.

Wert einer Zahl ist:
Wert der letzten Ziffer + 10-facher Wert der links davor stehenden Zahl.

z.B. $\text{Wert}(367) = 7 + 10 \cdot \text{Wert}(36)$
$= 7 + 10 \cdot (6 + 10 \cdot \text{Wert}(3))$
$= 7 + 10 \cdot (6 + 10 \cdot 3)$
$=$ Zahl 367.

Bsp für Alphabete:

- lat. Alphabet $\{a, .., z\}$
- ASCII - Code (128 Zeichen)
- $A_1 = \{0, 1\}$
- $A_2 = \{(,), +, -, *, /, a\}$

Bsp für Wörter:

- $A_1^* = \{\varepsilon, 0, 1, 00, 01, 010100, \ldots\}$
- $A_2^* = \{\varepsilon, (), (+-), \ldots\}$

Bsp für Sprachen

- $L = \{\varepsilon, 1, 10, 11, 100, \ldots\}$
  Binärzahlen ohne führende Nullen
- $EXPR = \{\varepsilon, (), (a+a), \ldots\}$
  alle korrekt geklammerten Ausdrücke

In dieser Vorlesung:

Grammatiken (um Prog-Sprache zu definieren)

Später (FoSAP):

Automaten (nötig für Compilerbau)

Bsp- Grammatik:

Nicht-Terminalsymbole:
Hilfssymbole zur Ableitung von Wörtern (z.B. Satz, Subjekt, ...)

Terminalsymbole:
(z.B der, die, ...)
Kommen in den Wörtern der Sprache vor.

Grammatik bekommt ein Startsymbol, aus dem alle Worte der Sprache abgeleitet werden können.
Hier: Startsymbol "Satz"

Satz ⇒
Subj Präd Obj ⇒
Art Attr Subst Präd Obj ⇒
der Attr Subst Pr Obj ⇒

der Subst Pr Obj ⇒

⋮ ⇒

der Hund jagt die Katze

Weitere Bspe:

der Kleine Katze jagt die Hund ✓

die Katze der Hund ✓

Ankündigungen

- Mo, 24.10., 16:15, Fo2: Vorlesung statt Übung
- Verteilung der Tutorien ist erledigt
  - siehe Üsg-System
  - erste Tutorien nächste Woche Mo + Di (24. + 25. 10.)
  - Gruppentausch können Sie selbst vornehmen
  - Tutoriumstausch ebenfalls möglich (falls "neue" Tutor einverstanden ist)
- Mail: Einladg. zum Codescape Spiel heute

Grammatik-Definition

Grammatik G ist 4-Tupel (N, T, P, S)

- N: endl. Menge von Nichtterminalsymbolen
  - Symbole für syntakt. Abstraktionen
  - Bsp: Satz, Subjekt, Prädikat, ...
  - kommen nicht in Wörtern der Sprache vor
  - werden durch Anwendung von Produktionsregeln ersetzt, bis nur noch Terminalsymbole übrig sind.
- T: endl. Menge von Terminalsymbolen, mit $N \cap T = \emptyset$
  - Zeichen des Alphabets, aus denen die Wörter der Sprache bestehen.
  - Bsp: der, die, das, kleine, Hund, ...
- P: endl. Menge von Produktionsregeln $x \to y$
  - $x \to y$ bedeutet: man kann das Teilwort x durch das Teilwort y ersetzen
  - $x \in V^* N V^*$, $y \in V^*$ wobei $V = N \cup T$

    d.h.: x enthält mindestens ein Nichtterminalsymbol.
  - Bsp: Prädikat $\to$ jagt

    $\in N$ $\in T$
- S: das Startsymbol, mit $S \in N$
  - Bsp: Startsymbol Satz

Ableitung

- Ableitungsprozess ist Relation

"$\Rightarrow$" auf V^*

- Für $u, v, y \in V^*$ und $X \in V^* N V^*$ gilt:

$$u X v \Rightarrow u y v \quad \text{falls,}$$

$$X \to y \quad \in P$$

- Bsp: der Substantiv jagt $\Rightarrow$

der Hund jagt

Die von einer Grammatik G erzeugte Sprache ist:

$$L(G) = \{ w \mid w \in T^*, \; S \Rightarrow \dots \Rightarrow w \}$$

d.h. alle Terminalwörter, die man mit Hilfe der Produktionsregeln aus dem Startsymbol ableiten kann.

2 Grammatiken G_1 und G_2 sind äquivalent gdw. ("genau dann wenn")

$$L(G_1) = L(G_2).$$

Sprache der Beispiel-Grammatik G:

$$A \Rightarrow a B b c \Rightarrow d c$$

$$\Downarrow$$

$$a a B b b c \Rightarrow a d b c$$

$$a^3 B b^3 c \Rightarrow a^2 d b^2 c$$
$$a^4 B b^4 c \Rightarrow \dots$$
$$L(G) = \{ a^n d b^n c \mid n \in \mathbb{N} \}$$

Man kann diese Sprache auch mit einer kontextfreien Grammatik erzeugen:

$A \to B\ c$

$B \to d$

$B \to a\ B\ b$

EBNF

- Aus $A \to y$ wird $A = y$
- Aus $A \to y_1, \dots, A \to y_n$ wird $A = (y_1 \mid \dots \mid y_n)$
- Aus $A \to xz$, $A \to xyz$ wird $A = x [y] z$
- Aus $A \to xA$, $A \to y$ wird $A = \{ x \} y$

Bsp-Grammatik in EBNF:

Bsp-Grammatik in EBNF:

Satz = Subj Präd Obj

Subj = Art Attr Substt

Art = [("der" | "die" | "das")]

Attr = {Adj}

Adj = ("kleine" | "bissige" | "große")

Substt = ("Hund" | "Katze")

Präd = "jagt"

Obj = Art Attr Substt

Syntaxdiagr. für Bsp-Grammatik: