I.2 Grundlagen von Programmiersprachen

Montag, 2. November 2020 10:30

Algorithmus: Idee der Lösungsbeschreibung

Programm: Formulierung d. Alg. in einer Programmiersprache

Software: Programm + Dokumentation

Hardware: Rechner + Peripherie

Eigenschaften eines Algorithmus:

- in endlichem Text beschreibbar
- effektiv (d.h. durch eine Maschine ausführbar)
- Determinismus: es liegt eindeutig fest, welche Elementaroperation als nächstes ausgeführt wird
- Determiniertheit: zu jeder Eingabe gibt es genau eine Ausgabe
- Terminierung: Alg. hält immer nach endl. vielen Schritten

Determinismus, Determiniertheit + Terminierung werden nicht immer verlangt.

Indet. Bsp.-Alg. ist trotzdem determiniert.

Eigenschaften eines "guten" Algorithmus:

- Allgemeinheit: Alg. sollte nicht nur ein einziges Problem, sd. Klasse von Problemen lösen.
- Änderbarkeit: Alg. sollte leicht modifizierbar sein

- Änderbarkeit: Alg. sollte leicht modifizierbar sein
- Effizienz: Anzahl d. benötigten Rechenschritte sollte mögl. gering sein.
- Robustheit: sollte sich bei unzulässigen Eingaben wohldefiniert verhalten.

Bsp für Syntax + Semantik: Sprache der nat. Zahlen ohne 0.

Syntax: Jede Zahl ist Folge v. Ziffern (0, 1, ..., 9), wobei erste Ziffer $\neq 0$.

Semantik: Wert einer Zahl ist Wert d. letzten Ziffer plus 10-facher Wert der Zahl links davor. Wert einer Ziffer ist die Ziffer selbst

z.B. 367

$$
\begin{aligned}
&\text{Wert}(367)\\
&= \underbrace{\text{Wert}(7)}_{7} + 10 \cdot \text{Wert}(36)\\
&= 7 + 10 \cdot \big(\underbrace{\text{Wert}(6)}_{6} + 10 \cdot \underbrace{\text{Wert}(3)}_{3}\big)\\
&= 7 + 10 \cdot (6 + 10 \cdot 3)\\
&= 367
\end{aligned}
$$

Dagegen 007 gehört nicht zur Sprache.

Bsp:

Alphabet

- lat. Alphabet (a, b, ..., z)

- ASCII-Code (128 Zeichen) American Standard Code for Information Interchange

- $A_1 = \{0, 1\}$
- $A_2 = \{\ (,\ ),\ +,\ -,\ *,\ /,\ a\ \}$

<u>Wörter</u>

$A_1^* = \{\varepsilon, 0, 1, 00, 01, 101100, \dots\}$

$A_2^* = \{\varepsilon, (\,), ((a+a)*a), (-+, \dots\}$

<u>Sprachen</u>

$L = \{\varepsilon, 1, 101100, \dots\}$ Binärzahlen ohne führende 0en

$EXPR = \{\varepsilon, ((a)), ((a+a)*a), \dots\}$ Menge der korrekt geklammerten Ausdrücke

Hier: nur Grammatiken

Automaten in Vorlesung FoSAP (2. Sem.)

Bsp:

Satz ⇒ Subj. <u>Präd.</u> Obj.

⇒ <u>Subj.</u> jagt Obj.

⇒ Art. Attr. Subst. jagt Objekt

⇒ ...

⇒ der große bissige Hund jagt die kleine Katze

Satz ~~⇒ ...~~ ⇒ die Katze der Hund

Satz ~~⇒ ...~~ ⇒ der Hund jagt die Maus

Satz $\Rightarrow \dots \Rightarrow$ das große Hund jagt die große Hund

Grammatik

$G = (N, T, P, S)$

- N = endl. Menge von Nichtterminalsymbolen

  Bsp: Satz, Subjekt, ...
  - kommen in Wörtern der Sprache nicht vor
  - werden durch wiederholte Anwendung der Produktionsregeln ersetzt.

- T = endl. Menge von Terminalsymbolen

  mit $N \cap T = \emptyset$

  Bsp: der, die, kleine, Hund, ...
  - Zeichen des Alphabets, aus denen die Wörter der Sprache bestehen

- P = endl. Menge von Produktionsregeln $\quad x \to y$

  mit $\quad x \in V^* N V^*$ ← Wort x enthält mindestens ein Nichtterminal

  $\quad\quad\; y \in V^*$

  wobei: $V = N \cup T$ (Vokabular)

  Regel bedeutet, dass man das Teilwort x durch das Teilwort y ersetzen kann.

  Bsp: Subj. Präd. Objekt $\Rightarrow$ Subj. jagt Objekt

- $S \in N$ das Startsymbol

  S ist ein spezielles Nichtterminal, aus dem alle Wörter der Sprache hergeleitet werden.

  Bsp: Satz

Ableitung:

ist eine Relation $\Rightarrow$ auf V^*

Für $u, v, y \in V^*$
$x \in V^* N V^*$ gilt:

$$u x v \Rightarrow u y v \quad , \text{ falls } \quad x \rightarrow y \in P$$

Sprache der Grammatik G:

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

d.h. alle Terminalwörter, die aus dem Startsymbol S mit Hilfe der Produktionsregeln P ableitbar sind.

Zwei Grammatiken sind äquivalent, wenn sie die gleiche Sprache erzeugen.

Eine Sprache ist kontextfrei, falls sie von einer kontextfreien Grammatik erzeugt wird.

Bsp-Grammatik

Grammatik G ist nicht kontextfrei.

Was ist $L(G)$?

$$A \Rightarrow aBbc \Rightarrow dc$$

$$aa\underline{B}bbc \Rightarrow adbc$$

$$\Downarrow$$

$$a^3 B b^3 c \Rightarrow a^2 d b^2 c$$

$$\Downarrow$$

$$a^4 B b^4 c \quad \ldots$$

$L(G) = \{ a^n d b^n c \mid n \in \mathbb{N} \}$

Diese Sprache ist trotzdem kontextfrei, denn es gibt eine andere kontextfreie Grammatik G', die äquivalent zu G ist.

Idee: Baue die Wörter von hinten auf.

$$A \to B\, c$$

B muss alle Wörter der Art $a^n d b^n$ erzeugen.

$$B \to d$$
$$B \to a\, B\, b$$

EBNF

- Aus $A \to Y$ wird $A = Y$
- Aus $A \to Y_1, \dots, A \to Y_n$ wird $A = (Y_1 \mid Y_2 \mid \dots \mid Y_n)$

  (Kurznotation für
  $(Y_1 \mid (Y_2 \mid (\dots \quad (Y_{n-1} \mid Y_n) \dots)))$.)

  Klammern lässt man oft weg.
- Aus $A \to XZ$ und $A \to XYZ$ wird $A = X\,[Y]\,Z$
- Aus $A \to XA$ und $A \to Y$ wird $A = \{X\}\,Y$

Bsp-Grammatik in EBNF

Satz = Subj. Präd. Obj

Subj = Art Attr Subst

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

Attr = { Adjektiv }

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

Subst = ("Hund" | "Katze")

Präd = "jagt"

Obj = Art Attr Subst

Syntaxdiagramme

für jedes Nichtterminal ein Syntaxdiagramm.

Die Pfeile geben an, welche Wörter aus dem Nichtterminal abgeleitet werden können.

Syntaxdiagramme können auch rekursiv sein, d.h. ein Nichtterminal kann in seinem eigenen Syntaxdiagr. auftreten.