I.2 Grundlagen von Programmiersprachen

Montag, 8. Oktober 2018 10:30

Algorithmus: Idee der Lösungsbeschreibung

Programm: Formulierung des Alg. in einer Programmiersprache

Software: Programm + Dokumentation

Hardware: Rechner + Peripherie

Algorithmus

- in endlichem Text beschreibbar
- effektiv, d.h. durch Maschine ausführbar

- besteht aus einzelnen Elementar-operationen u. meist muss eindeutig festliegen, welche Elementarop. jeweils als nächstes ausgeführt wird (Determinismus)
- erlaubt Ein- u. Ausgabe, wobei zu jeder Eingabe eine eindeutige Ausgabe gehört (Determiniertheit)
- Ein Algorithmus terminiert wenn er nach endlich vielen Schritten abbricht.

2. Bsp - Alg. ist indeterministisch, aber trotzdem determiniert.

Determinismus, Determiniertheit und Terminierung werden

nicht immer verlangt.

Kennzeichen eines guten Algorithmus:

- Allgemeinheit: Alg. sollte ganze Klasse von Problemen lösen.
- Änderbarkeit: Alg. sollte leicht modifizierbar (und verständlich) sein.
- Effizienz: Anzahl der benötigten Rechenschritte sollte mögl. gering sein.
- Robustheit: Wohldefiniertes Verhalten bei unzulässigen Eingaben.

Beispiele für Syntax +

Semantik

Bsp: Sprache der natürlichen Zahlen ohne 0

Syntax: Jede Zahl ist eine Folge von Ziffern (0, 1, .., 9), wobei erste Ziffer ≠ 0.

z.B. 5307

Semantik: Wert einer Zahl ist Wert der letzten Ziffer plus 10-facher Wert der links davon stehenden Zahl, Wert einer Ziffer ist die Ziffer selbst.

Wert (5307) =

Wert (7) + 10 · Wert (530) =

7

.... =

7 + 10 · (0 + 10 · (3 + 10 · 5))

Bsp für Alphabete

- lateinisches Alphabet (a, b, .., z)
- ASCII-Code (128 Zeichen)
  ↑ American Standard Code for Information Interchange
- $A_1 = \{0, 1\}$
- $A_2 = \{ (,), +, -, *, /, a \}$

Bsp für Worte

$A_1^* = \{ \varepsilon, 0, 01, 10, 11010, .. \}$

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

Bsp für Sprachen

- $L \subseteq A_1^*$ Binärzahlen ohne führende Nullen

$L = \{ \varepsilon, 1, 10, 11, 100, 101, ... \}$

- $EXPR \subseteq A_2^*$ Menge der korrekt

geklammerten Ausdrücke

EXPR = { ε, ( ), (a + (a - a)), ... }

Hier: nur Grammatiken

FoSAP (2. Sem): Grammatiken + Automaten

Bsp-Grammatik:

Satz ⇒ Subj Präd Obj

⇒ Subj jagt Objekt

⇒ Art Attr Subs jagt Objekt

⇒ ....

⇒ der große Hund jagt die kleine Katze

Satz ⇒ .... ⇒

der große Katze jagt

das kleine große Katze

Ableitung in sehr vielen Schritten

Satz $\not\in^*$ die Katze der Hund

Satz $\not\in^*$ der Hund jagt die Maus

Grammatik

- Grammatik $G = (N, T, P, S)$
- N: endl. Menge von Nichtterminalsymbolen

  Bsp: Satz, Subjekt, Artikel, ...

  Kommen nicht in Wörtern der Sprache vor, sondern werden durch Anwendung von Produktionsregeln solange ersetzt, bis nur noch Terminalsymbole übrig sind.
- T: endl. Menge von

Terminalsymbolen,

$N \cap T = \emptyset$

Bsp: der, die, Hund, kleine, ...

Aus diesen Zeichen des Alphabets bestehen die Wörter der Sprache.

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

mit $x \in V^* N V^*$, ← x enthält mindestens ein Nicht-terminal

$y \in V^*$

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

bedeutet: Teilwort x kann durch Teilwort y ersetzt werden

Bsp: der Hund Prädikat Objekt ⇒ der Hund jagt Objekt

- S: das Startsymbol, $S \in N$ ist ein spezielles Nichtterminal, aus dem alle Worte der Sprache hergeleitet werden können.

  Bsp: Satz

Ableitung

- Relation "$\Rightarrow$" auf V^*
- Für $u, v, y \in V^*$ und $x \in V^* \setminus V^*$ gilt:

$$u x v \Rightarrow u y v,$$

falls $(x \to y) \in P$

Von der Grammatik G erzeugte Sprache:

$$L(G) = \{ w \mid w \in T^*, $$

$S \Rightarrow \ldots \Rightarrow w \}$

dh: $L(G)$ sind alle Worte, die man erhält, indem man mit dem Startsymbol S beginnt und solange Produktionsregeln anwendet, bis nur noch Terminal-symbole übrig sind.

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

Eine Sprache ist kontext-frei, falls es eine kontextfreie Grammatik

gist, die sie erzeugt.

Bsp- Grammatik G

G ist nicht kontextfrei
wegen der Regel $aBb \to d$.

Was ist $L(G)$?

$A \Rightarrow aBbc \Rightarrow dc$

$\Downarrow$

$aaBbbc \Rightarrow adbc$

$\Downarrow$

$a^3Bb^3c \Rightarrow a^2db^2c$

$\Downarrow$

a^4Bb^4c ...

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

Ist diese Sprache kontextfrei,
d.h., existiert eine kontextfreie
Grammatik G' mit
$L(G') = L(G)$?

(G' ist äquivalent zu G)

Ja: $G' = (N, T, P', S)$

mit $P' = \{ A \to B\ c,$

$B \to d,$

$B \to a\ B\ b \}$

Aus B kann man alle Worte der Form $a^n d\ b^n$ herleiten.

EBNF

- aus $A \to y$ wird $A = y$
- aus $A \to y_1, \ldots, A \to y_n$ wird

  $A = (y_1 | \ldots | y_n)$

  (Abkürzung für $A = (y_1 | (y_2 | \ldots (y_{n-1} | y_n) \ldots))$)
- aus $A \to x\ z$, $A \to x\ y\ z$ wird

  $A = x\ [y]\ z$

- aus $A \to xA$, $A \to y$ wird

$A = \{x\}\, y$

Bsp-Grammatik in EBNF

Satz = Subj Präd Obj

Subj = Art Attr Subst

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

Attr = {Adj}

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

Subst = ("Hund" | "Katze")

Präd = "jagt"

Obj = Art Attr Subst

Syntaxdiagramme können auch rekursiv sein, d.h. das Syntaxdiagramm für das Nichtterminal Attribut könnte selbst wieder

[Attribut]

Attribut

enthalten.