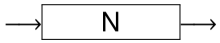
Unterabschnitt 2.2.2

Sprachdefinition mit Syntaxdiagrammen

Begriffsklärung: (Syntaxdiagramme)

Ein ***Syntaxdiagramm*** ist ein Flussgraph bestehend aus ***Nichtterminalknoten***:

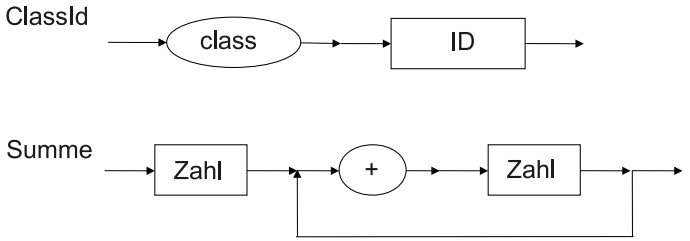


Terminalknoten:

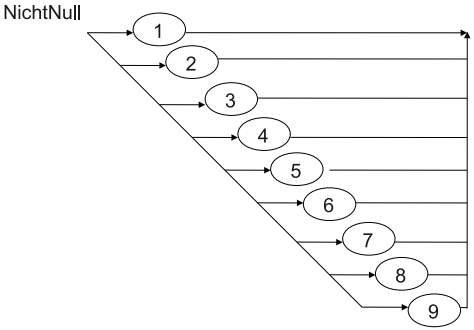
Extrakanten:

mit definiertem Eingang und Ausgängen. Die Knoten sind mit *Symbolen* markiert. Jedes Syntaxdiagramm hat ein Symbol als ***Namen***.

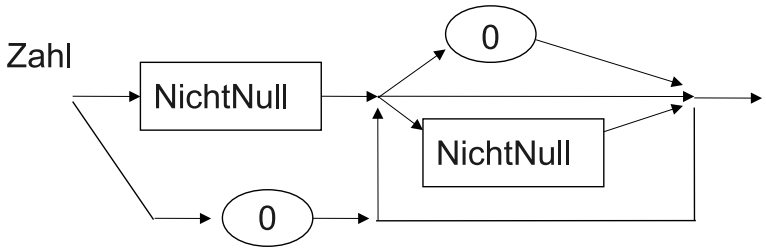
Beispiele: (Syntaxdiagramme)



Beispiele: (Syntaxdiagramme) (2)

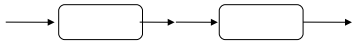


Beispiele: (Syntaxdiagramme) (3)

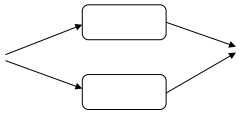


Grundtypen von Syntaxdiagrammen

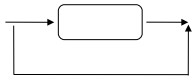
Sequenz:



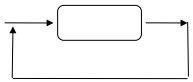
Alternative:

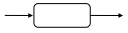


Option:



Wiederholung:



wobei  ein Nichtterminal- oder Terminalknoten ist.

Bemerkungen:

- Die Kantenführung wird meist freier gehandhabt.
- In der Literatur werden auch andere Knoten- und Kantenformen benutzt.

Definition: (Sprachdefinition mit Syntaxdiagrammen)

Seien T und N zwei disjunkte Alphabete.
Die Elemente von T nennen wir ***Terminalsymbole***, die von N ***Nichtterminalsymbole***.

Sei Δ eine endliche Menge von Syntaxdiagrammen mit Namen aus N, in denen Terminalknoten mit Terminalsymbolen und Nichtterminalknoten mit Nichtterminalsymbolen markiert sind.

Sei $S \in N$; S heißt ***Startsymbol*** oder ***Axiom***.

Dann heißt $\Gamma = (N, T, \Delta, S)$ eine ***Sprachdefinition mit Syntaxdiagrammen***, kurz SDmSD.

Begriffsklärung: (durch SDmSD definierte Sprache)

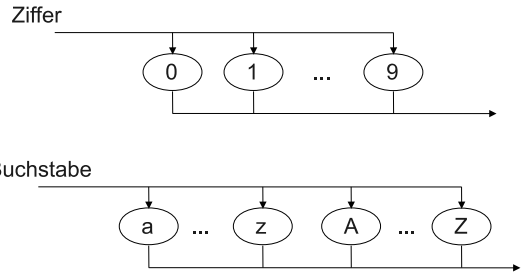
Die von einer ***Sprachdefinition mit Syntaxdiagrammen definierte Sprache*** ist die Menge der Zeichenreihen über T, die man durch „Ablaufen“ der Syntaxdiagramme ausgehend vom Syntaxdiagramm mit Namen S erhält.

Beispiel: (SDmSD)

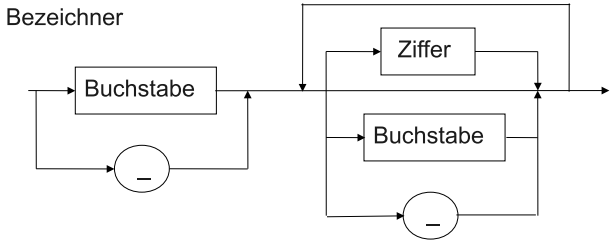
$T = \{a, b, ..., z, A, ..., Z, 0, 1, ..., 9, \_\}$

$N = \{\text{Bezeichner}, \text{ Buchstabe}, \text{ Ziffer}\}$

Sei Δ die Menge der folgenden drei Syntaxdiagramme:



Beispiel: (SDmSD) (2)



Dann definiert (N, T, Δ, Bezeichner) eine Menge von Worten, die in Programmiersprachen häufig als Menge der zulässigen Namen/Bezeichner verwendet wird.

Beispiel: (Programmiersprache „Femto “)

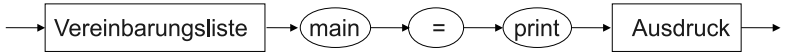
Als realistischeres Beispiel für eine Sprachdefinition betrachteten wir Vereinbarungslisten mit nachfolgender Ausgabevereinbarung.

Hier ein Beispiel für ein Femto-Programm:

```
a    = 73 ;
b    = ( 8 * a ) ;
main = print ( a + b )
```

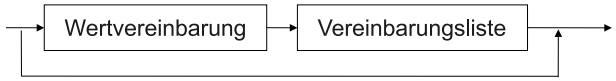
Definition mit Syntaxdiagrammen:

Programm

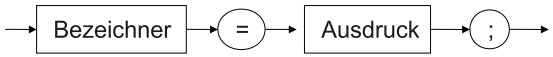


Beispiel: (Programmiersprache „Femto “) (2)

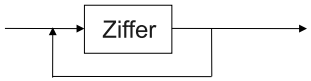
Vereinbarungsliste



Wertvereinbarung

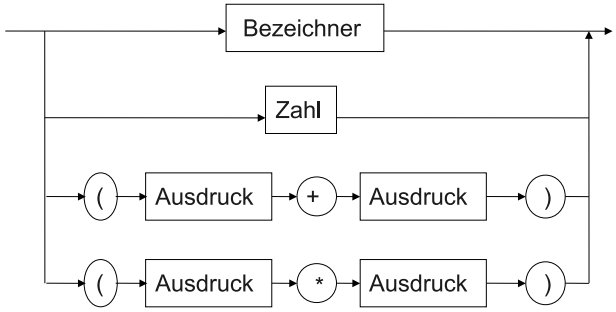


Zahl



Beispiel: (Programmiersprache „Femto “) (3)

Ausdruck



Beispiel: (Programmiersprache „Femto “) (4)

Seien T , N, Δ wie im vorherigen Beispiel und

- $TF = T \cup \{\text{main}, \text{print}, (,), *, +, =, ;\}$
- $NF = N \cup \{$ Programm, Vereinbarungsliste, Wertvereinbarung, Ausdruck, Zahl $\}$
- Φ die Menge der obigen Syntaxdiagramme

Dann beschreibt $(NF, TF, \Delta \cup \Phi, \text{Programm})$ die Syntax der Femto-Programme.

Unterabschnitt 2.2.3

Sprachdefinition mit kontextfreien Grammatiken

Definition: (Kontextfreie Grammatik)

Seien T und N zwei disjunkte Alphabete. Die Elemente von T nennen wir Terminalsymbole, die von N Nichtterminalsymbole.

Sei $\sqcap$ eine endliche Teilmenge von $N \times (N \cup T)^*$; die Elemente von $\sqcap$ heißen Produktionen oder Regeln.

Sei $S \in N$; S heißt Startsymbol oder Axiom.

Dann heißt $\Gamma = (N, T, \Pi, S)$ eine kontextfreie Grammatik, kurz kfG.

Notationen

- Produktionen werden in der Form $A \to \alpha$ notiert.
- Abkürzend steht $A \to \alpha \mid \beta \mid \gamma \mid \ldots$ für $A \to \alpha$, $A \to \beta$, $A \to \gamma$, $A \to \ldots$

Beispiele: (Kontextfreie Grammatik)

1. Grammatik für balancierte Klammerausdrücke:
   $T = \{(,)\}$
   $N = \{S\}$
   $\Pi = \{S \to SS \mid (S) \mid \varepsilon\}$
   Nichtterminal S ist Startsymbol.

Beispiele: (Kontextfreie Grammatik) (2)

2. Grammatik für Bezeichner in Programmiersprachen:

$$
\begin{array}{lcll}
T & = & \{ & a, b, ..., z, A, ..., Z, 0, 1, ..., 9, \_\} \\
N & = & \{ & \text{Bezeichner, Buchstabe, Ziffer, BezeichnerRest,} \\
 & & & \qquad\qquad \text{BustaUnstri } \} \\
\Pi & = & \{ & \text{Ziffer} \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 , \\
 & & & \text{Buchstabe} \to a \mid b \mid \ldots \mid z \mid A \mid \ldots \mid Z , \\
 & & & \text{Bezeichner} \to \text{BustaUnstri BezeichnerRest} , \\
 & & & \text{BustaUnstri} \to \text{Buchstabe} \mid \text{'\_'} , \\
 & & & \text{BezeichnerRest} \to \varepsilon \\
 & & & \qquad \mid \text{Ziffer BezeicherRest} \\
 & & & \qquad \mid \text{BustaUnstri BezeichnerRest} \\
 & & \} &
\end{array}
$$

Definitionen: (Begriffe zur Ableitbarkeit)

Sei $\Gamma = (N, T, \Pi, S)$ gegeben. Im Folgenden seien

- $A, B, C, \ldots$ aus N,
- $a, b, c, \ldots$ aus T,
- $x, y, z, \ldots$ aus T^* und
- $\alpha, \beta, \gamma, \psi, \varphi, \sigma, \tau, \ldots$ aus $(N \cup T)^*$.

- ψ ist mit Γ aus φ ***direkt ableitbar***, in Zeichen $\varphi \Rightarrow \psi$, wenn es Zerlegungen $\sigma A \tau$ von φ und $\sigma\alpha\tau$ von ψ gibt und $A \to \alpha$ in Π.

Definitionen: (Begriffe zur Ableitbarkeit) (2)

- ψ ist mit Γ aus φ ***ableitbar***, in Zeichen $\varphi \Rightarrow^* \psi$, wenn es $\varphi_0, \ldots, \varphi_n$ gibt mit $\varphi = \varphi_0$, $\varphi_n = \psi$ und für alle $i \in \{0, \ldots, n-1\} : \varphi_i \Rightarrow \varphi_{i+1}$

  $\varphi_0 \ldots \varphi_n$ heißt dann ***Ableitung*** von ψ aus φ.

- Eine Ableitung $\varphi \ldots \varphi_n$ heißt ***Linksableitung*** (bzw. ***Rechtsableitung***), wenn in φ_i jeweils nur das am weitesten links (bzw. rechts) stehende Nichtterminal ersetzt wird.

  Linksableitungsschritte werden als $\varphi \underset{lm}{\Rightarrow} \psi$,

  Rechtsableitungsschritte als $\varphi \underset{rm}{\Rightarrow} \psi$ notiert.

- Die baumartige Darstellung einer Ableitung nennen wir ***Syntaxbaum***.

Definitionen: (Begriffe zur Ableitbarkeit) (3)

- $L(\Gamma) = \{z \text{ in } T^* \mid S \Rightarrow^* z\}$ heißt die von Γ *erzeugte* ***Sprache***.
- x in $L(\Gamma)$ heißt ein ***Satz*** von Γ.
- ψ in $(N \cup T)^*$ mit $S \Rightarrow^* \psi$ heißt eine ***Satzform*** von Γ.
- Ein Satz heißt ***mehrdeutig***, wenn er mehr als einen Syntaxbaum besitzt.
- Eine Grammatik heißt ***mehrdeutig***, wenn sie einen mehrdeutigen Satz besitzt; andernfalls ***eindeutig***.

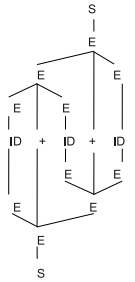
Beispiel: (Linksableitung)

Sei Γ die Grammatik der balancierten Klammerausdrücke (s.o.); die folgende Folge von Satzformen ist eine Linksableitung:

$$
\begin{aligned}
& S \\
\Rightarrow\ & (S) \\
\Rightarrow\ & (SS) \\
\Rightarrow\ & ((S)S) \\
\Rightarrow\ & (()S) \\
\Rightarrow\ & (())
\end{aligned}
$$

Beispiel: (mehrdeutige Grammatik)

Die Ausdrucksgrammatik
$\Gamma_0 : S \to E,\ E \to E + E \mid E * E \mid (E) \mid ID$
ist ein Beispiel für eine mehrdeutige Grammatik :



d.h. es gibt zwei Ableitungen von $ID + ID + ID$.

Bemerkungen:

- Jeder Ableitung entspricht genau ein Syntaxbaum. Umgekehrt kann es zu einem Syntaxbaum mehrere Ableitungen geben.
- Anstatt von Syntaxbaum spricht man häufig auch von ***Struktur-*** oder ***Ableitungsbaum***.
- Zusammenhang zwischen Sprache und Grammatik:
  Die Abbildung L : Grammatik $\to$ Sprache ist im Allg. nicht injektiv; d.h. zu einer Sprache gibt es im Allg. mehrere erzeugende Grammatiken.
- $\Rightarrow^*$ ist die reflexive und transitive Hülle von $\Rightarrow$.

Bemerkungen: (2)

- **Fakt**: Ein Satz ist genau dann eindeutig, wenn er genau eine Linksableitung (bzw. Rechtsableitung) besitzt.
- Bei Programmiersprachen spielen die *eindeutigen* Grammatiken die zentrale Rolle, da die Semantik(und Übersetzung) der Sprache über die syntaktische Struktur definiert wird.
- Mit kfGn lassen sich genau die Sprachen definieren, die man auch mit Syntaxdiagrammen definieren kann.

Unterabschnitt 2.2.4

Zur Semantik formaler Sprachen

Begriffsklärung: (Syntax, Semantik)

Die ***Syntax*** einer formalen Sprache gibt an, wie die Elemente der Sprache (Sätze, Worte, Diagramme) zusammengesetzt sind, d.h. welche *Form* sie haben.
Die ***Semantik*** gibt den Elementen einer formalen Sprache eine *Bedeutung*.

Begriffsklärung: (Syntax, Semantik) (2)

Form — Semantik ⟶ — Bedeutung

Syntaktische Gebilde:
- Zeichenreihen
- Diagramme
- Programme
- Terme
- Formeln
- Spezifikationen
- ...

Semantische Gebilde:
- Zahlen
- Mengen
- Listen
- Funktionen
- Datenstrukturen
- Algebren
- ...

Bemerkungen:

- Es gibt formale Sprachen ohne Semantik (z.B. die Sprache der balancierten Klammerausdrücke).
- Es gibt Sprachen mit formaler Syntax und informeller Semantik.
- Moderne Programmier- und Spezifikationssprachen besitzen formale Syntax und wohldefinierte Semantik.
- Syntax und Semantik zu unterscheiden bedeutet zwischen Beschreibung und Beschriebenem zu unterscheiden.
- Unterschiedliche Beschreibungen können die gleiche Semantik haben.
- Maschinen verarbeiten Beschreibungen.
- Salopp gesagt: „Semantik ist das, was ich will, Syntax ist der Weg, es der Maschine/den Mitmenschen mitzuteilen.“

Beispiel: (Informelle Semantik von Femto)

- Auf den Folien 135ff wurde die kontextfreie Syntax der Sprache Femto festgelegt.
- Zur Klärung der Semantik von Femto sind weitere syntaktische Einschränkungen notwendig (***Kontextbedingungen***):
  - Ein Bezeichner darf nur dann in einem Ausdruck angewendet werden, wenn er vorher vereinbart wurde.
  - Die Zahlen müssen Werte im Bereich von -2^{31} bis $2^{31} - 1$ beschreiben.

Beispiel: (Informelle Semantik von Femto) (2)

- Die Semantik weist jedem Ausdruck und dem ganzen Programm einen Wert im Bereich $[-2^{31}, 2^{31} - 1]$ zu:
  - angewandter Bezeichner: Liefert den an den Bezeichner gebundenen Wert.
  - Zahl: Liefert den von der Zahl beschriebenen Wert.
  - zusammengesetzter Ausdruck:
    - → Werte die Teilausdrücke aus; die Ergebnisse seien *lw* und *rw*.
    - → Ist das Operatorzeichen $'+'$, addiere *lw* und *rw*; ist das Operatorzeichen $'*'$, multipliziere *lw* und *rw*; das Ergebnis sei mit *w* bezeichnet.
    - → Liegt *w* im Bereich -2^{31} bis $2^{31} - 1$, ist *w* das Ergebnis der Ausdrucksauswertung; andernfalls ist das Ergebnis unbestimmt.
  - Wertvereinbarung: Werte den Ausdruck aus und binde das Ergebnis an den Bezeichner.
  - Programm: Drucke den Wert des Ausdrucks in der Ausgabevereinbarung.

Bemerkungen:

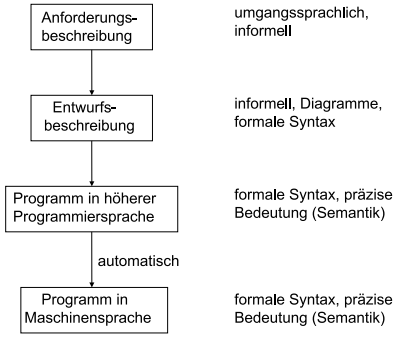
- Syntaktisch inkorrekte Programme haben keine Semantik: z.B. : main $a == a$; a;
- Syntaxfehler werden vom Übersetzer bzw. Interpreter entdeckt.
- Eine formale Semantik definiert die Bedeutung mit mathematischen Mitteln.
- Die Implementierung einer Sprache in Form von Übersetzern/Interpretern liefert auch eine Semantik, die allerdings oft rechnerabhängig ist.

Unterabschnitt 2.2.5

Beschreibung von SW-Systemen

Beschreibung von SW-Systemen

Bei der Entwicklung von SW-Systemen werden auf vielen Ebenen Beschreibungen eingesetzt (vgl. Folie 91):



Dabei werden informelle und formale Sprachen kombiniert.

Diskussion: (informelle/formale Beschr.)

- Informelle Beschreibungen:
  - besser lesbar und besser vermittelbar
  - oft kompakter
  - flexibler und mächtiger, d.h. sie können sich auch auf Bereiche beziehen, die sich formal nicht oder nur schwer ausdrücken lassen
  - weniger Vorkenntnisse
- Formale Beschreibungen:
  - präziser
  - maschinell auswertbar (!!!!!)
  - zugänglich für mathematische Methoden
- Kombination formaler und informeller Sprachen:
  - Anforderungen anfangs informell
  - schrittweiser Übergang möglich und wichtig (!)

Bemerkungen:

- Softwareentwicklung hat sehr viel mit genauem Verstehen und Beschreiben zu tun.
- Spezielle Entwicklungsmethoden bauen auf die Integration der unterschiedlichen Beschreibungsebenen oder Beschreibungsstile; z.B. ist *literate programming* eine Methode, die die Einbettung der Programme in die allgemeine Dokumentation unterstützt.

Unterabschnitt 2.2.6

Paradigmen der Programmierung

Paradigmen der Programmierung

Paradigmen werden charakterisiert durch das Zusammenwirken bestimmter Konzepte, Vorgehensweisen, Techniken, Theorien und Standards.
In der Softwareentwicklung und Programmierung unterscheidet man:

- deklarative Programmierung
- prozedurale Programmierung
- objektorientierte Programmierung

Deklaratives Paradigma

Ansatz:
Verwende mathematische Beschreibungsmittel zur Modellierung und Programmierung:

- Listen, Mengen, Produkt-, Summenbildung
- Relationen, Funktionen
- Algebren, Funktoren,
- Terme, Formeln, logisches Schließen

Deklaratives Paradigma (2)

Umsetzung:

- funktionale Programmierung
- logische Programmierung
- Spezifikationssprachen
- relationale Datenbanken
- Entity Relationship Modellierung

Größte praktische Bedeutung des deklarativen Paradigmas liegt derzeit im Bereich der Datenmodellierung und als Grundlage und Abfragesprachen (SQL) für Datenbanken.

Beispiel: (logisches Programmieren)

In der logischen Programmierung wird ein Programm als eine Ansammlung von Fakten und Folgerungsbeziehungen gegeben; z.B. in Prolog:

```
professor( breuel ).
mensch( sokrates ) .
mensch( X )  :--  professor( X ) .
sterblich( X )  :--  mensch( X ) .
```

Die Anwendung eines logischen Programms wird über Anfragen ausgelöst, z.B.:

```
mensch( sokrates )?
sterblich( breuel )?
sterblich( X )?
sterblich( beethoven )?
```


Idee des deklarativen Paradigmas

Grundlegende Idee des deklarativen Paradigmas:

- Nutze die Stärke mathematischer Begriffsbildung. Abstrahiere von den Verarbeitungsschritten, d.h. betrachte nur das Verhältnis von Ein- und Ausgabe.

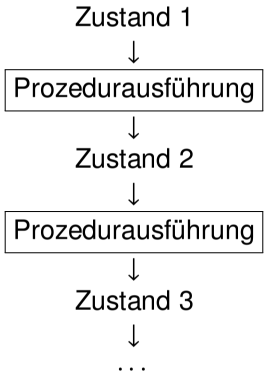
Erschwert aber die Modellierung

- nicht-terminierender, paralleler und verteilter Verarbeitung von Informationen;
- graphischer, geflechtartiger Strukturen;
- gleichartiger Objekte mit unterschiedlichen Identitäten.

Prozedurales Paradigma

- Modelliere Information mit Variablen und Daten. Die Variablen beschreiben die Systemzustände.
- Zerlege den Verarbeitungsprozess in Schritte und beschreibe diese Schritte durch Prozeduren.
- Konzepte entstanden als Abstraktion der Arbeitsweise von Rechnern.

Prozedurales Paradigma (2)



Objektorientiertes Paradigma

Das Paradigma der Objektorientierung bezieht seine konzeptionellen Grundlagen aus der realen Welt:

Für den Menschen besteht

- die physische/materielle Umgebung und
- die gedankliche Welt

aus logisch zusammengehörigen Objekten mit

- eigenständiger Identität
- einem Zustand, der sich mit der Zeit ändern kann, der Möglichkeit auf sie einwirken zu können bzw. der Fähigkeit zu kommunizieren und zu kooperieren.