Kapitel 3.3 Syntaxdefinitionen

Syntax

Syntax: Festlegung des Satzbaus.

Beispiele syntaktisch falscher deutscher Sätze:

Kai liest eine Buch. Buch lesen Kai. Kai pr1&. Kai liest ein Buch, weil ihr ist langweilig.

Beispiele syntaktisch falscher OCAML-Phrasen (Frasen?):

```
let let x = 3 in 2;;
let if = 1;;
2 + * 3;;
2 : 3;;
```

Semantik

Semantik: Festlegung der Bedeutung eines Satzes.

Semantische Fragen im Deutschen: Worauf bezieht sich ein Relativpronomen? Welchen Einfluss haben Fragepartikel wie “eigentlich”, “denn”? Wann verwendet man welche Zeitform?

Semantische Fragen bei OCAML: Was ist der Wert von Ausdrücken wie

```
let x = 1 in let y = x in let x = 2 in y;;
let x = 1./.0. in 2;;
let rec f x = f x in if true then 1 else f 0;;
```

Grundfrage der Semantik von Programmiersprachen: Welche Wirkung hat ein syntaktisch korrektes Programm?

Aus *historischen Gründen* werden Fragen der Typüberprüfung auch der Semantik zugerechnet.

Formale Syntax

- Ein *Alphabet* ist eine endliche Menge, deren Elemente *Symbole* genannt werden.
- Eine *Zeichenkette* (auch *Wort*) über einem Alphabet Σ ist eine endliche Folge von Elementen $\sigma_1, \ldots, \sigma_n$ von Σ, wobei $n \geq 0$. Man schreibt ein Wort als $\sigma_1\sigma_2 \ldots \sigma_n$. Der Fall $n = 0$ bezeichnet das *leere Wort* geschrieben ε.
- Die Menge aller Wörter über Σ wird mit Σ^* bezeichnet. Zu zwei Wörtern $w = \sigma_1 \ldots \sigma_n$ und $w' = \sigma'_1 \ldots \sigma'_m$ bildet man die *Verkettung* (Konkatenation) $ww' = \sigma_1 \ldots \sigma_n\sigma'_1 \ldots \sigma'_m$. Es ist $\varepsilon w = w\varepsilon = w$ und $(ww')w'' = w(w'w'')$.
- Eine *formale Sprache* ist eine Teilmenge von Σ^*.

Beispiele

$\Sigma = \{a, b\}$.

Wörter über Σ: $aaba, baab, bababa, baba, \varepsilon, bbbb$.

Sprachen über Σ: $\emptyset$, $\{a, b\}$, $\{a^n b^n \mid n \in \mathbb{N}\}$.

$\Sigma = \{\texttt{0}, \texttt{1}, \ldots, \texttt{9}, \texttt{e}, \texttt{-}, \texttt{+}, \texttt{.}, \texttt{E}\}$.

Wörter über Σ: `-1E98`, `--2e--`, `32.e`

Sprachen über Σ: Syntaktisch korrekte `float` Konstanten,
$\{\texttt{e}^n \mid n$ gerade$\}$.

$\Sigma = \{\texttt{0}, \ldots, \texttt{9}, \texttt{if}, \texttt{then}, \texttt{let}, \ldots\}$

Sprache über Σ: alle syntaktisch korrekten OCAML Phrasen.

Beispiel: OCAML-Bezeichner

- Bezeichner sind Zeichenketten über dem Alphabet $\Sigma = \{\texttt{A}, \ldots, \texttt{Z}, \texttt{a}, \ldots, \texttt{z}, \texttt{0}, \ldots, \texttt{9}, \texttt{'}, \texttt{\_}\}$. Bezeichner müssen mit einem Kleinbuchstaben oder dem Zeichen _ beginnen.
- Ein *Buchstabe* ist ein *Kleinbuchstabe* oder ein *Großbuchstabe*
- Ein *Kleinbuchstabe* ist ein Zeichen $\texttt{a}, \ldots, \texttt{z}$.
- Ein *Großbuchstabe* ist ein Zeichen $\texttt{A}, \ldots, \texttt{Z}$.
- Eine *Ziffer* ist ein Zeichen $\texttt{0}, \ldots, \texttt{9}$.

Formale Definition als BNF-Grammatik

BNF = *Backus-Naur Form*

$$
\begin{array}{lcl}
\langle \textit{Bezeichner} \rangle & ::= & \{\langle \textit{KlBuchst} \rangle \mid \_\}\{\langle \textit{Buchst} \rangle \mid \langle \textit{Ziffer} \rangle \mid \texttt{'} \mid \_\}^* \\
\langle \textit{Buchst} \rangle & ::= & \langle \textit{KlBuchst} \rangle \mid \langle \textit{GrBuchst} \rangle \\
\langle \textit{KlBuchst} \rangle & ::= & \texttt{a} \mid \texttt{b} \mid \texttt{c} \mid \texttt{d} \mid \texttt{e} \mid \texttt{f} \mid \texttt{g} \mid \texttt{h} \mid \texttt{i} \mid \texttt{j} \mid \texttt{k} \mid \texttt{l} \mid \texttt{m} \mid \texttt{n} \mid \texttt{o} \\
 & & \mid \texttt{p} \mid \texttt{q} \mid \texttt{r} \mid \texttt{s} \mid \texttt{t} \mid \texttt{u} \mid \texttt{v} \mid \texttt{w} \mid \texttt{x} \mid \texttt{y} \mid \texttt{z} \\
\langle \textit{GrBuchst} \rangle & ::= & \texttt{A} \mid \texttt{B} \mid \texttt{C} \mid \texttt{D} \mid \texttt{E} \mid \texttt{F} \mid \texttt{G} \mid \texttt{H} \mid \texttt{I} \mid \texttt{J} \mid \texttt{K} \mid \texttt{L} \mid \texttt{M} \mid \texttt{N} \mid \texttt{O} \\
 & & \mid \texttt{P} \mid \texttt{Q} \mid \texttt{R} \mid \texttt{S} \mid \texttt{T} \mid \texttt{U} \mid \texttt{V} \mid \texttt{W} \mid \texttt{X} \mid \texttt{Y} \mid \texttt{Z} \\
\langle \textit{Ziffer} \rangle & ::= & \texttt{0} \mid \texttt{1} \mid \texttt{2} \mid \texttt{3} \mid \texttt{4} \mid \texttt{5} \mid \texttt{6} \mid \texttt{7} \mid \texttt{8} \mid \texttt{9}
\end{array}
$$

`float`-Literale

Literal = Konstante in einer Programmiersprache.

$$
\begin{array}{lcl}
\langle \textit{float-Literal} \rangle & ::= & [\langle \textit{Vorzeichen} \rangle]\langle \textit{Mantisse} \rangle[\langle \textit{Exponent} \rangle] \\
\langle \textit{Vorzeichen} \rangle & ::= & \texttt{-} \mid \texttt{+} \\
\langle \textit{Mantisse} \rangle & ::= & \{\langle \textit{Ziffer} \rangle\}^{+}[\texttt{.}\{\langle \textit{Ziffer} \rangle\}^{*}] \\
\langle \textit{Exponent} \rangle & ::= & \{\texttt{e} \mid \texttt{E}\}[\langle \textit{Vorzeichen} \rangle]\{\langle \textit{Ziffer} \rangle\}^{+}
\end{array}
$$

Zusätzliche *Kontextbedingung*: Entweder ein Dezimalpunkt, oder ein `e` oder ein `E` muss vorhanden sein.

Übung: man verbessere die BNF Darstellung so, dass diese Kontextbedingung wegfallen kann.

Syntax der BNF

Eine BNF-Grammatik ist ein Quadrupel $G = (\Sigma, V, S, P)$.

- Σ ist die Menge der Terminalsymbole, meist in `Courier` gesetzt.
- V ist die Menge der Nichtterminalsymbole, meist in spitze Klammern gesetzt. Im Beispiel:
  $V = \{\langle \textit{float-Literal} \rangle, \langle \textit{Mantisse} \rangle, \langle \textit{Vorzeichen} \rangle, \langle \textit{Exponent} \rangle\}$.
- $S \in V$ ist ein ausgezeichnetes Nichtterminalsymbol, das *Startsymbol*. Im Beispiel: $S = \langle \textit{float-Literal} \rangle$.
- P ist eine endliche Menge von *Produktionen* der Form $X ::= \delta$, wobei δ eine *BNF-Satzform*, s.u., ist.

BNF-Satzformen

- Jedes Symbol in $V \cup \Sigma$ ist eine BNF Satzform.
- Sind $\gamma_1, \ldots, \gamma_n$ BNF-Satzformen, so auch $\gamma_1 \mid \cdots \mid \gamma_n$ (*Auswahl*).
- Sind $\gamma_1, \ldots, \gamma_n$ BNF-Satzformen, so auch $\gamma_1 \ldots \gamma_n$ (*Verkettung*).
- Ist γ eine BNF Satzform, so auch $\{\gamma\}$ (*Klammerung*).
- Ist γ eine BNF Satzform, so auch $\{\gamma\}^*$ (*Iteration*).
- Ist γ eine BNF Satzform, so auch $\{\gamma\}^+$ (*nichtleere Iteration*).
- Ist γ eine BNF Satzform, so auch $[\gamma]$ (*Option*).

Eine Satzform der Gestalt $\gamma_1 \mid \cdots \mid \gamma_n$ muss immer geklammert werden, es sei denn, sie tritt unmittelbar als rechte Seite einer Produktion auf.

Semantik der BNF

Gegeben sei eine EBNF Grammatik $G = (\Sigma, V, S, P)$. Jedem Nichtterminalsymbol X in V und jeder BNF Satzform δ wird eine Sprache $\mathcal{L}(X)$, bzw. $\mathcal{L}(\delta)$ wie folgt zugeordnet.

- Falls $x \in \Sigma$, so ist $\mathcal{L}(x) = \{x\}$.
- Falls $X \in V$, so ist $\mathcal{L}(X) = \bigcup_{\text{“}X ::= \delta\text{”} \in P} \mathcal{L}(\delta)$.
- $\mathcal{L}(\gamma_1 \mid \cdots \mid \gamma_n) = \mathcal{L}(\gamma_1) \cup \cdots \cup \mathcal{L}(\gamma_n)$.
- $\mathcal{L}(\gamma_1 \ldots \gamma_n) = \{w_1 w_2 \ldots w_n \mid w_1 \in \mathcal{L}(\gamma_1), \ldots, w_n \in \mathcal{L}(\gamma_n)\}$.
- $\mathcal{L}(\{\gamma\}^*) = \{w_1 w_2 \ldots w_n \mid n \in \mathbb{N}, w_i \in \mathcal{L}(\gamma)\}$
- $\mathcal{L}(\{\gamma\}^+) = \{w_1 w_2 \ldots w_n \mid n \in \mathbb{N}, n > 0, w_i \in \mathcal{L}(\gamma)\}$
- $\mathcal{L}([\gamma]) = \mathcal{L}(\gamma) \cup \{\varepsilon\}$

Semantik durch Ersetzung

Satz: Ein Wort w ist in $\mathcal{L}(S)$, wenn man es aus S durch die folgenden Ersetzungsoperationen erhalten kann:

- Falls $X ::= \gamma_1 \mid \cdots \mid \gamma_n$ eine Produktion ist, so darf man ein Vorkommen von X durch eines der γ_i ersetzen.
- Ein Vorkommen von $\{\gamma_1 \mid \cdots \mid \gamma_n\}$ darf man durch eines der γ_i ersetzen.
- Ein Vorkommen von $\{\gamma\}^*$ darf man durch $\overbrace{\{\gamma\}\{\gamma\}\ldots\{\gamma\}}^{n\text{-mal}}$ mit $n \geq 0$ ersetzen.
- Ein Vorkommen von $\{\gamma\}^+$ darf man durch $\overbrace{\{\gamma\}\{\gamma\}\ldots\{\gamma\}}^{n\text{-mal}}$ mit $n > 0$ ersetzen.
- Ein Vorkommen von $\{\gamma\}$ darf man durch γ ersetzen, wenn γ nicht von der Form $\gamma_1 \mid \cdots \mid \gamma_n$ ist.
- Ein Vorkommen von $[\gamma]$ darf man durch $\{\gamma\}$ ersetzen, oder ersatzlos streichen.

Beispiel

$\langle$*float-Literal*$\rangle \to [\langle$*Vorzeichen*$\rangle]\langle$*Mantisse*$\rangle[\langle$*Exponent*$\rangle] \to$
$\langle$*Mantisse*$\rangle[\langle$*Exponent*$\rangle] \to \{\langle$*Ziffer*$\rangle\}^+[\texttt{.}\{\langle$*Ziffer*$\rangle\}^*][\langle$*Exponent*$\rangle] \to$
$\{\langle$*Ziffer*$\rangle\}^+\texttt{.}\{\langle$*Ziffer*$\rangle\}^*[\langle$*Exponent*$\rangle] \to$
$\{\langle$*Ziffer*$\rangle\}^+\texttt{.}\{\langle$*Ziffer*$\rangle\}\{\langle$*Ziffer*$\rangle\}[\langle$*Exponent*$\rangle] \to$
$\{\langle$*Ziffer*$\rangle\}\texttt{.}\{\langle$*Ziffer*$\rangle\}\{\langle$*Ziffer*$\rangle\}[\langle$*Exponent*$\rangle] \to \texttt{2.71}\langle$*Exponent*$\rangle \to$
$\texttt{2.71}\{\texttt{e} \mid \texttt{E}\}[\langle$*Vorzeichen*$\rangle]\{\langle$*Ziffer*$\rangle\}^+ \to$
$\texttt{2.71E}\langle$*Vorzeichen*$\rangle\{\langle$*Ziffer*$\rangle\}\{\langle$*Ziffer*$\rangle\}\{\langle$*Ziffer*$\rangle\} \to$
$\texttt{2.71E}\{\texttt{-} \mid \texttt{+}\}\texttt{001} \to \texttt{2.71E-001}$

Also $\texttt{2.71E-001} \in \mathcal{L}(\langle$*float-Literal*$\rangle)$.

Kontextbedingungen

Manche syntaktische Bedingungen lassen sich mit BNF nur schwer oder gar nicht formulieren.

Man gibt daher manchmal zusätzliche *Kontextbedingungen* an, denen die syntaktisch korrekten Wörter zusätzlich genügen müssen.

Beispiele:

- Bezeichner dürfen nicht zu den Schlüsselwörtern gehören wie z.B. `let`, `if`, etc.
- float-Literale müssen `.`, `e`, oder `E` enthalten.

Andere Bedingungen, wie korrekte Typisierung oder rechtzeitige Definition von Bezeichnern werden, wie schon gesagt, der Semantik zugerechnet.

Varianten

Häufig wird statt $\{\gamma\}^*$ nur $\{\gamma\}$ geschrieben. Für die Klammerung verwendet man dann runde Klammern.

Die spitzen Klammern zur Kennzeichnung der Nichtterminalsymbole werden oft weggelassen.

Steht kein `Courier` Zeichensatz zur Verfügung, so schließt man die Terminalsymbole in “Anführungszeichen” ein.

Ableitungsbäume

Ableitungen in einer BNF lassen sich grafisch durch *Ableitungsbäume* darstellen.

Beispiele von Ableitungsbäumen finden Sie in [Kröger].

Diese Ableitungsbäume sind für die Festlegung der Semantik von Bedeutung.

Ein *Parser* berechnet zu einem vorgegebenen Wort einen Ableitungsbaum, falls das Wort in $\mathcal{L}(S)$ ist, und erzeugt eine Fehlermeldung, falls nicht.

Diese Aufgabe bezeichnet man als *Syntaxanalyse*.

Syntaxdiagramme

Man kann BNF Syntaxdefinitionen auch grafisch durch *Syntaxdiagramme* darstellen.

Terminalzeichen werden als Kreise dargestellt.

Für jedes Nichtterminalzeichen X (als Kasten dargestellt) ein Diagramm. Eine Zeichenreihe aus $\mathcal{L}(X)$ erhält man, indem man im Diagramm für X einen beliebigen Weg vom Anfang zum Ausgang wählt, dabei jedes auftretende Terminalzeichen aufsammelt und jedes auftretende Nichtterminalzeichen Y durch ein entsprechend gefundenes Wort aus $\mathcal{L}(Y)$ ersetzt.

Siehe [Kröger] für Beispiele von Syntaxdiagrammen.

OCAML-Sprachdefinition (Fragment)

⟨*toplevel-command*⟩ ::= `let` ⟨*let-binding*⟩{ `and` ⟨*let-binding*⟩}*`;;`
| `let rec` ⟨*let-binding*⟩{ `and` ⟨*let-binding*⟩}*`;;`
| ⟨*expression*⟩`;;`

⟨*expression*⟩ ::= ⟨*constant*⟩
| `(`⟨*expression*⟩`)`
| ⟨*expression*⟩⟨*infix-op*⟩⟨*expression*⟩
| `if` ⟨*expression*⟩ `then` ⟨*expression*⟩ `else` ⟨*expression*⟩
| `let` ⟨*let-binding*⟩{ `and` ⟨*let-binding*⟩}* `in` ⟨*expression*⟩
| `let rec`⟨*let-binding*⟩{ `and` ⟨*let-binding*⟩}* `in` ⟨*expression*⟩
| `function` ⟨*ident*⟩ `->` ⟨*expression*⟩
| {⟨*expression*⟩}$^+$
| ⟨*expression*⟩`,`⟨*expression*⟩

⟨*infix-op*⟩ ::= `+` | `-` | `*` | `/` | `+.` | `-.` | `*.` | `/.` | `&&` | `||` | `mod`
| `<` | `>` | `<>` | `<=` | `>=` | `<>`

⟨*let-binding*⟩ ::= {⟨*ident*⟩}$^+$ `=` ⟨*expression*⟩

Parsergeneratoren

Erinnerung: *Parser* : $\Sigma^* \to$ *Ableitungsbäume* $\cup$ *Fehlermeldungen*.

Ein *Parsergenerator* erzeugt aus einer BNF-Grammatik automatisch einen Parser.

Der bekannteste Parsergenerator heißt “yacc” (*yet another compiler-compiler*). Er erzeugt aus einer BNF-Grammatik einen in der Programmiersprache C geschriebenen Parser.

Für OCAML gibt es “ocamlyacc”. Es wird ein OCAML-Programm erzeugt.

Geschichte

Grammatikformalismen, die syntaktisch korrekte Wörter durch einen Erzeugungsprozess definieren (wie die BNF) heißen *generative Grammatiken*.

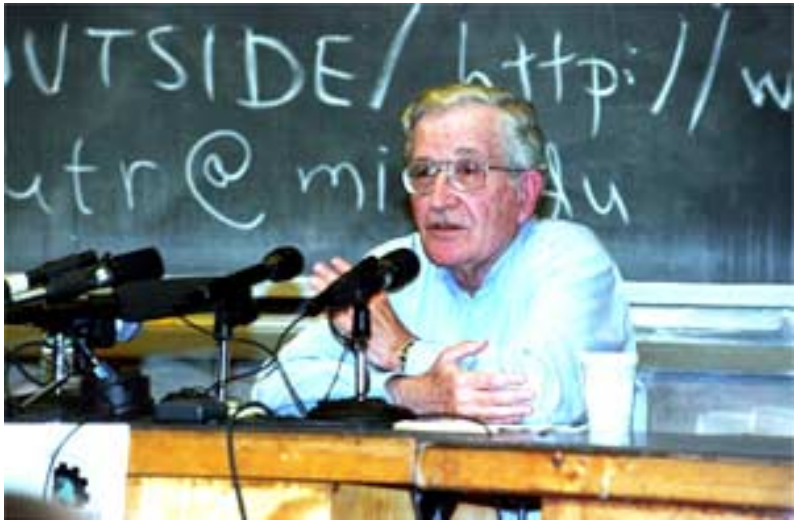
Sie gehen auf den Sprachforscher NOAM CHOMSKY (1928–) zurück.

Eine *kontextfreie Chomsky-Grammatik* ist eine BNF-Grammatik ohne die Konstrukte $\{\ \}^*, [\ ], |, \{\ \}^+$. Man kann jede BNF-Grammatik durch eine kontextfreie Chomsky-Grammatik simulieren.

Chomsky betrachtet u.a. auch kontextsensitive Grammatiken.

Die Backus-Naur-Form wurde von den Informatikern JOHN BACKUS und PETER NAUR entwickelt, die die Bedeutung von Chomskys generativen Grammatiken für Programmiersprachensyntax erkannten.

Noam Chomsky



NOAM CHOMSKY

$$
\begin{array}{lcl}
\langle \mathit{Bezeichner} \rangle & ::= & \langle \mathit{KlBuchst} \rangle \langle \mathit{Folge} \rangle \\
\langle \mathit{Bezeichner} \rangle & ::= & \_ \langle \mathit{Folge} \rangle \\
\langle \mathit{Folge} \rangle & ::= & \langle \mathit{Zeichen} \rangle \langle \mathit{Folge} \rangle \\
\langle \mathit{Folge} \rangle & ::= & \varepsilon \\
\langle \mathit{Zeichen} \rangle & ::= & ' \\
\langle \mathit{Zeichen} \rangle & ::= & \_ \\
\langle \mathit{Zeichen} \rangle & ::= & \langle \mathit{Buchst} \rangle \\
\langle \mathit{Zeichen} \rangle & ::= & \langle \mathit{Ziffer} \rangle \\
\langle \mathit{Buchst} \rangle & ::= & \langle \mathit{KlBuchst} \rangle \\
\langle \mathit{Buchst} \rangle & ::= & \langle \mathit{GrBuchst} \rangle
\end{array}
$$