Chomsky-Hierarchie

Sprachen, Grammatiken, Automaten

Eduard Mehofer

Fakultät für Informatik

Währinger Straße 29

Universität Wien

Definition einer Sprache (1)

Sei Σ ein Alphabet. Eine formale Sprache L ist als eine Teilmenge $L \subseteq \Sigma^*$ definiert. Wie lassen sich nun formale Sprachen präzise beschreiben? – Folgende Methoden existieren:

1. Auflistung/Aufzählung aller Sprachelemente

Die explizite Auflistung aller Sprachelemente ist nur für endliche Sprachen (mit relativ wenigen Elementen) möglich und daher für die meisten in der Praxis interessanten Sprachen irrelevant.

2. Spezifikation eines formalen Ausdrucks

Ein Beispiel für diese Methode sind reguläre Ausdrücke, womit beliebige reguläre Sprachen beschrieben werden können. Zum Beispiel kann man die Menge der Bezeichner einer Programmiersprache durch $L = B(B|Z)^*$ darstellen, wenn B die Menge der Buchstaben und Z die Menge der Ziffern darstellt.

Definition einer Sprache (2)

3. Grammatiken

Grammatiken lassen sich zur (rekursiven) Erzeugung der Sätze einer Sprache benutzen. Es wird ein Verfahren angegeben, das systematisch alle Sätze einer Sprache generiert.

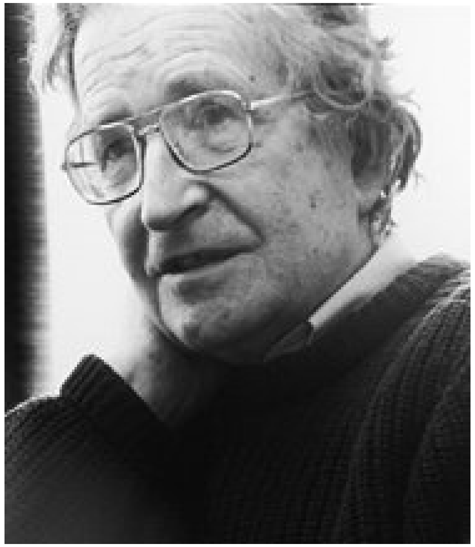
4. Automaten

Automaten lassen sich zur Erkennung der Sätze einer Sprache verwenden. Es wird ein Verfahren spezifiziert, das für jedes Wort w entscheidet, ob w in der Sprache enthalten ist oder nicht. Die von einem Automaten A akzeptierte Sprache L(A) ist die Menge aller akzeptierten Wörter.

Chomsky-Hierarchie

Die Chomsky Hierarchie (nach Noam Chomsky; 1956,1959) legt eine Hierarchie von vier Sprachklassen fest und formalisiert die Beziehung zu Grammatik- und Automatenklassen.

Noam Chomsky, geb. 1928,
bekannter amerikanischer Linguist,
emer. Prof. MIT.
Populär: Medienkritik, polit. Engagement



Grammatiken der Chomsky Hierarchie

Eine Chomsky Grammatik hat die Form $G = (N, \Sigma, P, S)$, wo N und Σ die Alphabete der Nichtterminal- bzw Terminalsymbole darstellen, P die Regelmenge ist und $S \in N$ das Startsymbol darstellt. Die einzelnen Grammatikklassen werden durch die Form ihrer Regeln unterschieden:

- **Typ-0 Grammatik** (allgemeine Grammatik): siehe folg. Folien.
- **Typ-1 Grammatik** (kontextsensitive Grammatik): siehe folg. Folien.
- **Typ-2 Grammatik** (kontextfreie Grammatik): bekannt
  - $A \to \alpha$ mit $A \in N$, $\alpha \in \Gamma^*$ $(\Gamma = N \cup \Sigma)$
- **Typ-3 Grammatik** (reguläre Grammatik): bekannt
  - rechtslinear: $A \to x$ oder $A \to xB$, mit $A, B \in N$ und $x \in \Sigma^*$.
  - linkslinear: $A \to x$ oder $A \to Bx$, mit $A, B \in N$ und $x \in \Sigma^*$.

Automaten der Chomsky Hierarchie

Den vier Grammatik- und Sprachklassen der Chomsky Hierarchie entsprechen vier Klassen von Automaten. Die 4 Automatenklassen heißen:

- **Typ-0: Turingmaschinen** (siehe folg. Folien)
- **Typ-1: Linear beschränkte Automaten** (siehe folg. Folien)
- **Typ-2: Kellerautomaten** (siehe folg. Folien)
- **Typ-3: Endliche Automaten** (bekannt)

Sätze zur Chomsky Hierarchie

- **Typ-i Automaten ($0 \leq i \leq 3$) erkennen genau die von Typ-i Grammatiken generierten Sprachen.**

- Eine Sprache L die von einer Typ-i Grammatik G erzeugt wird, i.e. L=L(G), bzw. von einem Typ-i Automaten M akzeptiert wird, i.e. L=L(M), heisst Typ-i Sprache.

Hierarchiesatz:

Bezeichne L_i, $0 \leq i \leq 3$, die Familie der Chomsky Typ-i Sprachen. Dann gilt (echte Teilmengenbeziehungen; L_3 echte Teilmenge von L_2 usw., d.h. L_2 umfasst mehr Sprachen als L_3):

$$\mathbf{L_3 \subset L_2 \subset L_1 \subset L_0}$$

Sätze

1. Zu jedem nichtdeterministischen endlichen Automaten A, gibt es einen deterministischen Automaten A´ mit L(A) = L(A´).

2. Zu jedem regulären Ausdruck α existiert ein endlicher Automat A mit L(A) = L(α).

3. Zu jedem endlichen Automaten A existiert ein regulären Ausdruck α mit L(α) = L(A).

4. Ist G eine reguläre Grammatik, dann gibt es einen endlichen Automaten A mit L(A) = L(G).

5. Ist A ein endlicher Automat, dann existiert eine reguläre Grammatik G mit L(G) = L(A).

Bemerkungen zur Hierarchieeigenschaft

- Die Sprache

$$\{ a^n b^n \mid n \geq 1 \}$$

ist kontextfrei, aber nicht regulär.

(Kontextfreie Grammatik: $G = (\{S\}, \{a, b\}, \{S \to aSb \mid ab \} , S)$.)

- Die Sprache

$$\{ a^n b^n c^n \mid n \geq 1 \}$$

ist kontextsensitiv, aber nicht kontextfrei.

Pumping Lemma:

Wird verwendet um zu zeigen, dass gewisse Sprachen nicht regulär bzw. nicht kontextfrei sind.

P.L.1: Pumping Lemma für reguläre Sprachen.

Sei L eine reguläre Sprache. Dann gibt es ein $\mathbf{n}$, so dass jedes Wort $w \in L$ mit $|w| \geq \mathbf{n}$ in $w = x\,y\,z$ zerlegt werden kann, wobei

1. $y \neq \varepsilon$
2. $|x\,y\,| \leq \mathbf{n}$
3. $x\,y^{\mathbf{i}}\,z \in L$ für alle $i \geq 0$

P.L.2: Pumping Lemma für kontextfreie Sprachen.

Sei L eine kontextfreie Sprache. Dann gibt es ein $\mathbf{n}$, so dass jedes Wort $z \in L$ mit $|z| \geq \mathbf{n}$ in $z = u\,v\,w\,x\,y$ zerlegt werden kann, wobei

1. $v\,x \neq \varepsilon$
2. $|v\,w\,x\,| \leq \mathbf{n}$
3. $u\,v^{\mathbf{i}}\,w\,x^{\mathbf{i}}\,y \in L$ für alle $i \geq 0$

P.L.1

Beh.: $L=\{0^i 1^i \mid i \geq 0\}$ ist nicht regulär.

Bew.: indirekt

Angenommen L ist regulär, dann $\exists$ n, sodass nach P.L.1 gilt:

$w = 0^n 1^n$, $w \in L$, $|w| \geq n$ und $w = x\,y\,z$ wobei $|xy| \leq n$, $|y| \geq 1$

- $|xy| \leq n$ bedeutet: $xy \in \{0\}^+$ (d.h. nur 0er),
- somit gilt: $w=\underbrace{0^s}_{x}\underbrace{0^t}_{y}\underbrace{0^p\,1^n}_{z}$

  mit $\mathbf{s+t+p=n}$ und $s+t \leq n$ (da $|xy| \leq n$), $t \geq 1$ (da $|y| \geq 1$), $p \geq 0$

Nach P.L.1 muss $x\,y^i\,z \in L$ für alle $i \geq 0$.

- Für i=0: $x\,y^0\,z = x\,z = 0^s\,0^p\,1^n = \mathbf{0^{s+p}\,1^n \notin L}$

  da $\mathbf{s+p \neq n}$ weil $\mathbf{t \geq 1}$ (s+t+p=s+p=n nur wenn t=0)
- Widerspruch und daher Annahme falsch

■

P.L.2 (1)

Beh.: $L=\{0^i 1^i 2^i \mid i \geq 0\}$ ist nicht kontextfrei.

Bew.: indirekt
Angenommen L ist kontextfrei, dann $\exists$ n, sodass nach P.L.2 gilt:
$z = 0^n 1^n 2^n$, $z \in L$, $|z| \geq n$ und $z = u\ v\ w\ x\ y$ wobei $|vwx| \leq n$, $|vx| \geq 1$

$|vwx| \leq n$ bedeutet: vwx kann nicht 0er und 2er enthalten, da letzte 0 und erste 2 von n+1 Positionen getrennt sind

Fall 1: vwx hat keine 2er.

- Daraus folgt: vx hat nur 0er und 1er und mind. eine 0 oder 1 ($|vx| \geq 1$)
- Nach P.L.1 muss $u\ v^i\ w\ x^i\ y \in L$ für alle $i \geq 0$.
  Für i=0: $u\ v^0\ w\ x^0\ y = u\ w\ y$ muss aus L sein,
  aber: $n_2(uwy)=n$ jedoch $n_0(uwy) < n$ oder $n_1(uwy) < n$ da $|vx| \geq 1$.
  Daher gilt: **$u\ w\ y \notin L$**
- Widerspruch

P.L.2 (2)

Bew.: (Beweis Fortführung)

Fall 2: vwx hat keine 0er.

- Analog zu Fall 1 erhält man:
  $n_0(uwy)=n$ jedoch $n_1(uwy) < n$ oder $n_2(uwy) < n$ da $|vx| \geq 1$.
  Daher gilt: **$u\ w\ y \notin L$**
- Widerspruch

Aus Widerspruch Fall 1 und Fall 2 folgern wir, dass die Annahme falsch ist.

Anm.: $n_x(w)$ … Anzahl von x in w

■

Typ-0 Grammatiken

Definition: Eine **Typ-0 Grammatik** (allgemeine Grammatik) ist ein Quadrupel
$G = (N, \Sigma, P, S)$

wobei N, Σ und S die gleiche Interpretation haben wie bei einer KFG. Wie bei einer regulären Grammatik unterscheidet sich eine Typ-0 Grammatik von einer KFG durch die Vorschriften für Produktionsregeln.

Für Regeln einer Typ-0 Grammatik gilt: (Gesamtalphabet $\Gamma = (N \cup \Sigma)$)

$$\mathbf{P} \subseteq \mathbf{\Gamma}^* \, \mathbf{N} \, \mathbf{\Gamma}^* \times \mathbf{\Gamma}^*$$

Regeln einer Typ-0 Grammatik sind also von der Form $\alpha \to \beta$, wobei α und β Worte über dem Gesamtalphabet sind, mit der einzigen Bedingung, dass α mindestens ein Nichtterminalsymbol enthalten muss. Wie bei KFG lassen sich Regeln als Vorschriften für Textersetzungen für Worte aus Γ^* interpretieren, wobei ganze Teilworte und nicht nur einzelne Nichtterminalsymbole ersetzt werden können. Man beachte, dass bei einer Typ-0 Grammatik i.a. die Konstruktion eines Ableitungsbaums nicht mehr möglich ist.

Beispiel: Typ-0 Grammatik

Beispiel: Gesucht ist eine Grammatik für die Sprache $L = \{ a^{2^i} \mid i \geq 1 \}$.

Typ-0 Grammatik G mit L(G) = L ist gegeben durch:
$G = (\{ S, A, B, C, D, E \}, \{ a \}, P, S)$

P = {

1: S → ACaB,	**5: aD → Da,**
2: Ca → aaC,	**6: AD → AC,**
3: CB → DB,	**7: aE → Ea,**
4: CB → E,	**8: AE → ε }**

A und B dienen als linke und rechte Markierungen für das Ende der Satzform. Var. C läuft über die aus a´s bestehende Zeichenkette von A nach B und verdoppelt dabei mit Regel 2) die Anzahl der a´s. Var. D läuft von B zurück nach A. Wenn man terminieren möchte, löscht man mit Regel 4) Var. B, Var. E läuft zurück nach A und löscht A.

S ⇒ ACaB ⇒ AaaCB ⇒ AaaE ⇒ AaEa ⇒ AEaa ⇒ aa

oder

S ⇒ ACaB ⇒ AaaCB ⇒ AaaDB ⇒ AaDaB ⇒ ADaaB ⇒ ACaaB ⇒ ...

Typ-1 Grammatiken

Definition: Eine Typ-1 Grammatik (kontextsensitive Grammatik) ist ein Quadrupel G = (N, Σ, P, S), wobei jede Regel in P eine der beiden Formen besitzt:

1. $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \alpha \alpha_2$

   mit $A \in N$, $\alpha_1, \alpha_2 \in \Gamma^*$ und $\alpha \in \Gamma^+$

2. $S \rightarrow \varepsilon$

   und S tritt nicht auf der rechten Seite einer Regel auf.

Da für kontextsensitive Grammatiken gilt, dass, mit Ausnahme von **$S \rightarrow \varepsilon$**, die linke Seite einer Produktionsregel nicht verkürzt wird, i.e. für $\alpha \rightarrow \beta$ gilt, dass $|\alpha| \leq |\beta|$, sind kontextsensitive Grammatiken auch monotone Grammatiken.

Beispiel: Typ-1 Grammatik - Erster Versuch (1)

Beispiel: Gesucht ist eine kontextsensitive Grammatik für die Sprache $L = \{a^n b^n c^n \mid n \geq 1\}$.

Lösung:
Idee: man erzeugt $\mathbf{a}^n(BC)^n$, alle C´s müssen nach hinten verschoben werden, B wird terminal auf **b** und C wird terminal auf **c** abgeleitet (Kontext in Prod. unterstrichen).

P = {
1: $S \to \mathbf{a}SBC$,
2: $S \to \mathbf{a}BC$,
3: $CB \to BC$,
4: $\underline{\mathbf{a}}B \to \underline{\mathbf{a}}\mathbf{b}$,
5: $\underline{\mathbf{b}}B \to \underline{\mathbf{b}}\mathbf{b}$,
6: $\underline{\mathbf{b}}C \to \underline{\mathbf{b}}\mathbf{c}$,
7: $\underline{\mathbf{c}}C \to \underline{\mathbf{c}}\mathbf{c}$
}

3a: $C\underline{B} \to C_1\underline{B}$
3b: $\underline{C_1}B \to \underline{C_1}C$
3c: $C_1\underline{C} \to B\underline{C}$

Produktion 3 muß durch 3a/3b/3c ersetzt werden!

$G = (\{S, B, C, C_1\}, \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}, P, S)$

Problem: Produktion 3 erfüllt nicht unsere Einschränkung für kontextsensitive Sprachen! Produktionen 1,2,3,4,5,6,7 definieren daher eine Typ-0 Grammatik. Die entsprechende Typ-1 Grammatik besteht aus den Produktionen 1,2,3a,3b,3c,4,5,6,7.

Beispiel: Typ-1 Grammatik - Erster Versuch (2)

P = {
1: S → **a**SBC,
2: S → **a**BC,
3: CB → BC,
4: $\underline{\mathbf{a}}$B → $\underline{\mathbf{a}}$**b**,
5: $\underline{\mathbf{b}}$B → $\underline{\mathbf{b}}$**b**,
6: $\underline{\mathbf{b}}$C → $\underline{\mathbf{b}}$**c**,
7: $\underline{\mathbf{c}}$C → $\underline{\mathbf{c}}$**c**
}

3a: $C\underline{B} \rightarrow C_1\underline{B}$
3b: $\underline{C_1}B \rightarrow \underline{C_1}C$
3c: $C_1\underline{C} \rightarrow B\underline{C}$

Produktion 3 muß durch 3a/3b/3c ersetzt werden!

G = ({S, B, C, C_1}, {**a**, **b**, **c**}, P, S)

Ableitung:

$n = 2$: $S \Rightarrow a\underline{S}BC \Rightarrow a\underline{aB}CBC \Rightarrow aa\underline{bC}BC \Rightarrow aabcBC \not\Rightarrow$

Problem: es wird das erste C zu früh terminal ausproduziert und blockiert B.

unterschiedliche Lösungsansätze möglich

Beispiel: Typ-1 Grammatik - Alternativ (1)

<u>Bsp</u>: Gesucht ist eine kontextsensitive Grammatik für die Sprache L= $\{a^n b^n c^n \mid n \geq 1\}$.

<u>Lösung</u>:
Idee: man erzeugt $\mathbf{a}^n\,(BC)^n$, alle B´s müssen nach links verschoben werden, B wird terminal auf **b**, C auf **c** (Kontext in Prod. unterstrichen).

P = {
1/2: $S \to \mathbf{a}T B\mathbf{c} \mid \mathbf{abc}$,
3/4: $T \to \mathbf{a}TBC \mid \mathbf{ab}C$,
5: $CB \to BC$,
6: $\underline{\mathbf{b}}B \to \underline{\mathbf{b}}\mathbf{b}$,
7: $C\underline{\mathbf{c}} \to \mathbf{c}\underline{\mathbf{c}}$
}
G = ({S,T,B,C,C_1}, {**a**, **b**, **c**}, P, S)

5a: $\underline{C}B \to \underline{C}C_1$
5b: $C\underline{C_1} \to B\underline{C_1}$
5c: $\underline{B}C_1 \to \underline{B}C$

Produktion 5 muß durch 5a/5b/5c ersetzt werden plus 5d !

5d: $\underline{\mathbf{b}}C_1 \to \underline{\mathbf{b}}C$

Problem: Produktion 5 erfüllt nicht unsere Einschränkung für kontextsensitive Sprachen! Produktionen 1,2,3,4,5,6,7 definieren daher eine Typ-0 Grammatik. Die entsprechende Typ-1 Grammatik besteht aus den Produktionen 1,2,3,4,5a,5b,5c,5d, 6,7.

Beispiel: Typ-1 Grammatik - Alternativ (2)

P = {

1/2: $S \rightarrow \mathbf{a}TB\mathbf{c} \mid \mathbf{abc}$,

3/4: $T \rightarrow \mathbf{a}TBC \mid \mathbf{ab}C$,

5: $CB \rightarrow BC$,

6: $\underline{\mathbf{b}}B \rightarrow \underline{\mathbf{b}}\mathbf{b}$,

7: $C\underline{\mathbf{c}} \rightarrow \mathbf{c}\underline{\mathbf{c}}$

}

$G = (\{S,T,B,C,C_1\}, \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}, P, S)$

5a: $\underline{C}B \rightarrow \underline{C}C_1$

5b: $C\underline{C_1} \rightarrow B\underline{C_1}$

5c: $\underline{B}C_1 \rightarrow \underline{B}C$

Produktion 5 muß durch 5a/5b/5c ersetzt werden plus 5d !

5d: $\underline{\mathbf{b}}C_1 \rightarrow \underline{\mathbf{b}}C$

Ableitungen:

$n = 1$: $S \Rightarrow abc$

$n = 2$: $S \Rightarrow a\underline{T}Bc \Rightarrow aab\underline{CB}c \Rightarrow aab\underline{CC_1}c \Rightarrow aab\underline{BC_1}c \Rightarrow$

(i): $\Rightarrow aabBCc \overset{*}{\Rightarrow} aabbcc$

(ii): $\Rightarrow aab\underline{bC_1}c \Rightarrow aabbCc \Rightarrow aabbcc$

$n = 3$: $S \overset{*}{\Rightarrow} aaa\boldsymbol{b}CBCB\boldsymbol{c} \overset{*}{\Rightarrow} aaabBBCCc \overset{*}{\Rightarrow} aaabbbccc$