6b
Der CYK-Algorithmus

PD Dr. Jan Johannsen

Institut für Informatik

Stand: 19. Mai 2026
Basierend auf Folien von PD Dr. David Sabel und Prof. Dr. Jasmin Blanchette

Wiederholung: Entscheidbarkeit

Definition

Eine Sprache L ist entscheidbar, wenn es einen Algorithmus gibt, der bei Eingabe eines Wortes w in endlicher Zeit feststellt, ob $w \in L$ gilt oder nicht.

Wortproblem für Typ 2-Grammatiken

Definition

Das Wortproblem für Typ i-Grammatiken ist die Frage, ob für eine gegebene Typ i-Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $w \in \Sigma^*$ $w \in L(G)$ gilt oder nicht.

Satz

Das Wortproblem für Typ 2-Grammatiken ist entscheidbar:
Es gibt einen Algorithmus, der bei Eingabe von Typ 2-Grammatik G und Wort w nach endlicher Zeit entscheidet, ob $w \in L(G)$ gilt oder nicht. Zudem entscheidet er das Wortproblem in Polynomialzeit.

Grundgedanke des Algorithmus

Der CYK-Algorithmus von Cocke, Younger und Kasami ist ein Polynomialzeitalgorithmus für das Wortproblem für Typ 2-Grammatiken.

Eingabe: Eine CFG $G = (V, \Sigma, P, S)$ in Chomsky-Normalform und ein Wort $w \in \Sigma^+$.

> G ist in Chomsky-Normalform, wenn für jede Produktion $A \to w \in P$ gilt: $w = a \in \Sigma$ oder $w = BC$ mit $B, C \in V$.
>
> Für jede CFG G mit $\varepsilon \notin L(G)$ kann eine Chomsky-Normalform G' mit $L(G') = L(G)$ berechnet werden.

Ausgabe: ja, wenn $w \in L(G)$, sonst nein.

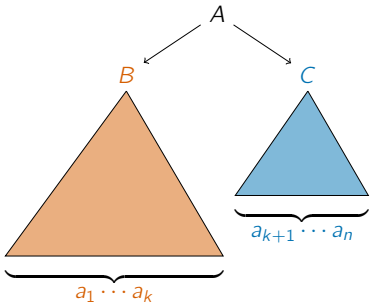
Grober Ansatz:

1. Teste für jede Variable A und Teilwort u von w, ob sie es erzeugt.
2. Verwende Test zum Prüfen, ob das Startsymbol S das Wort w erzeugt.

Grundgedanke des CYK-Algorithmus

Prüfe, ob $A \in V$ ein Wort $u = a_1 \cdots a_n$ $(n \geq 1)$ erzeugt:

- Wenn $u = a_1 \in \Sigma$, dann prüfe ob $A \to a_1 \in P$.
- Anderenfalls $(n > 1)$ kann u nur erzeugt werden, wenn
  - es eine Produktion $A \to BC \in P$ gibt und
  - es einen Index $1 \leq k < n$ gibt, sodass
    B erzeugt $a_1 \cdots a_k$ und
    C erzeugt $a_{k+1} \cdots a_n$.



- Daher prüfe für alle $A \to BC \in P$ und alle k mit $1 \leq k < n$ rekursiv, ob B das Wort $a_1 \cdots a_k$ und C das Wort $a_{k+1} \cdots a_n$ erzeugt.

Beispiel für naives rekursives Suchen

Seien die CFG $G = (V, \Sigma, P, S)$ mit

$$P = \{S \to AB \mid BA,\\ A \to AA \mid AB \mid a,\\ B \to BB \mid b$$

und das Wort *bbbaab*.

S erzeugt *bbbaab*, denn $S \to BA \in P$ und

- B erzeugt *bbb*, denn $B \to BB \in P$ und
  - B erzeugt *bb*, denn $B \to BB \in P$ und
    - B erzeugt *b*, denn $B \to b \in P$
    - B erzeugt *b*, denn $B \to b \in P$
  - B erzeugt *b*, denn $B \to b \in P$
- A erzeugt *aab*, denn $A \to AB \in P$ und
  - A erzeugt *aa*, denn $A \to AA \in P$ und
    - A erzeugt *a*, denn $A \to a \in P$ und
    - A erzeugt *a*, denn $A \to a \in P$
  - B erzeugt *b*, denn $B \to b \in P$.

Um an Effizienz zu gewinnen:
Statt Rekursion verwende dynamische Programmierung.

Grundgedanke des CYK-Algorithmus

Der Algorithmus berechnet Mengen $V(i,j) \subseteq V$, sodass

$$V(i,j) := \{A \in V \mid A \Rightarrow^* a_i \cdots a_{i+j-1}\}$$

Informell: $V(i,j)$ enthält alle Variablen $A \in V$, die $a_i \cdots a_{i+j-1}$
(= das Teilwort von w ab Position i mit Länge j) erzeugen.

Schritte:

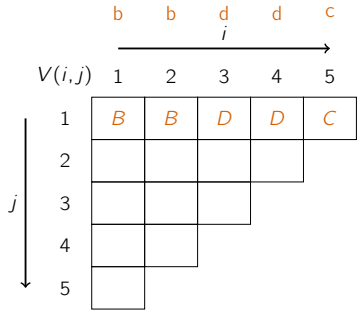
1. Beginne mit $V(i,1) = \{A \mid A \to a_i \in P\}$.
2. Berechne $V(i,j)$ für $j = 2$ bis n. Für $j > 1$ gilt:

   $A \in V(i,j)$ g.d.w. es gibt $k \in \{1,2,\ldots,j-1\}$, sodass
   $A \to BC \in P$, $B \in V(i,k)$ und $C \in V(i+k, j-k)$

3. Prüfe, ob $S \in V(1,n)$.

Beispiel für den CYK-Algorithmus

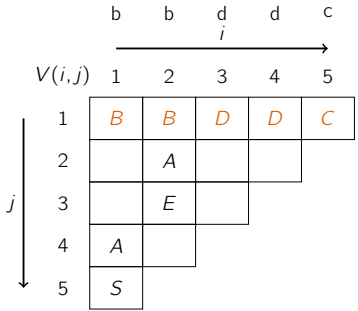
Seien $w = bbddc$ und $G = (\{S, A, B, C, D, E\}, \{b, c, d\}, P, S)$ mit
$P = \{S \to AC,\ A \to BE,\ A \to BD,\ E \to AD,\ C \to c,\ B \to b,\ D \to d\}$.



1. Schritt: Füllen der $V(i, 1)$-Einträge

Beispiel für den CYK-Algorithmus

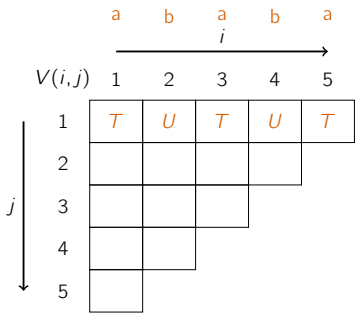
Seien $w = bbddc$ und $G = (\{S, A, B, C, D, E\}, \{b, c, d\}, P, S)$ mit $P = \{S \to AC,\ A \to BE,\ A \to BD,\ E \to AD,\ C \to c,\ B \to b,\ D \to d\}$.



3. Schritt: Da $S \in V(1,5)$, gilt $w \in L(G)$.

Weiteres Beispiel für den CYK-Algorithmus

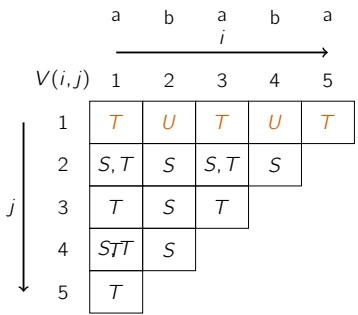
Seien $w = ababa$ und $G = (\{S, T, U\}, \{a, b, c\}, P, S)$ mit
$P = \{S \to TU,\ S \to UT,\ T \to TT,\ T \to TU,\ T \to a,\ U \to UU,\ U \to b\}$.



1. Schritt: Füllen der $V(i, 1)$-Einträge

Weiteres Beispiel für den CYK-Algorithmus

Seien $w = ababa$ und $G = (\{S, T, U\}, \{a, b, c\}, P, S)$ mit
$P = \{S \to TU,\ S \to UT,\ T \to TT,\ T \to TU,\ T \to a,\ U \to UU,\ U \to b\}$.



3. Schritt: Da $S \notin V(1,5)$, gilt $w \notin L(G)$.

Algorithmus 8: CYK-Algorithmus

Eingabe: CFG $G = (V, \Sigma, P, S)$ in Chomsky-Normalform und Wort $w = a_1 \cdots a_n \in \Sigma^+$
Ausgabe: `Ja`, wenn $w \in L(G)$, und `Nein`, wenn $w \notin L(G)$
Beginn

- **für** $i = 1$ *bis* n **tue**
  - $V(i,1) := \{A \in V \mid A \to a_i \in P\}$
- **für** $j = 2$ *bis* n **tue**
  - **für** $i = 1$ *bis* $n + 1 - j$ **tue**
    - $V(i,j) := \emptyset$;
    - **für** $k = 1$ *bis* $j - 1$ **tue**
      - $$V(i,j) := V(i,j) \cup \left\{ A \in V \;\middle|\; \begin{array}{l} A \to BC \in P, \\ B \in V(i,k), \\ C \in V(i+k, j-k) \end{array} \right\}$$
- **wenn** $S \in V(1,n)$ **dann**
  - return `Ja`
- **sonst**
  - return `Nein`

Laufzeit des CYK-Algorithmus

Theorem

Das Wortproblem für Typ 2-Grammatiken ist entscheidbar:
Es gibt einen Algorithmus, der bei Eingabe von Typ 2-Grammatik G und Wort w nach endlicher Zeit entscheidet, ob $w \in L(G)$ gilt oder nicht. Zudem entscheidet er das Wortproblem in Polynomialzeit.

Beweis Algorithmus 8 ist eine Entscheidungsprozedur.

Er besteht aus drei geschachtelte für-Schleifen. Im Inneren wird noch über alle Produktionen aus P iteriert. Die Laufzeitkomplexität kann daher mit $\mathsf{O}(n^3 \cdot |P|)$ abgeschätzt werden. □

Online-Tool

Tobias Lindebar hat ein Web-Tool zum Üben und Anschauen entwickelt:

www.cip.ifi.lmu.de/~lindebar/

Das Endlichkeitsproblem

Das Endlichkeitsproblem für Typ i-Grammatiken ist die Frage, ob für eine gegebene Typ i-Grammatik G die Ungleichheit $|L(G)| < \infty$ gilt.

Satz

Das Endlichkeitsproblem für kontextfreie Grammatiken ist entscheidbar.

Wir brauchen zuerst ein Lemma.

Entscheiden des Endlichkeitsproblems

Lemma

Sei G eine CFG in Chomsky-Normalform.
Sei n die Zahl aus dem Pumping-Lemma für kontextfreie Sprachen.
Es gilt $|L(G)| = \infty$ g.d.w. es ein Wort $z \in L(G)$ mit $n \leq |z| < 2n$ gibt.

Beweis Siehe Skript (nur FSK).

□

Entscheiden des Endlichkeitsproblems

Satz

Das Endlichkeitsproblem für kontextfreie Grammatiken ist entscheidbar.

Beweis Entscheidungsprozedur:

1. Berechne die Zahl n.
2. Teste mit dem CYK-Algorithmus für alle Wörter $w \in \Sigma^+$ der Länge $n \leq |w| < 2n$, ob $w \in L(G)$ gilt.
3. Wenn $w \in L(G)$ für eines der Wörter gilt, dann $|L(G)| = \infty$.
4. Sonst $|L(G)| < \infty$. □