Theoretische Informatik III (T3INF2002)

Formale Sprachen und Automaten | Einführung Compilerbau

Vorlesung im Wintersemester 2022/23

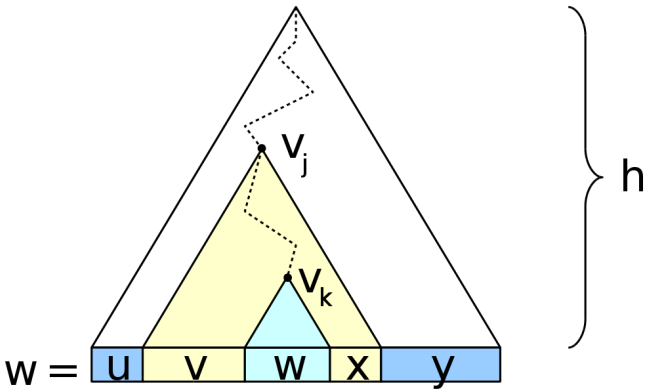
Formale Sprachen und Automaten

- Pumping-Lemma für kontextfreie Sprachen
- CYK-Algorithmus (Cocke, Younger, Kasami)

Pumping-Lemma für kontextfreie Sprachen

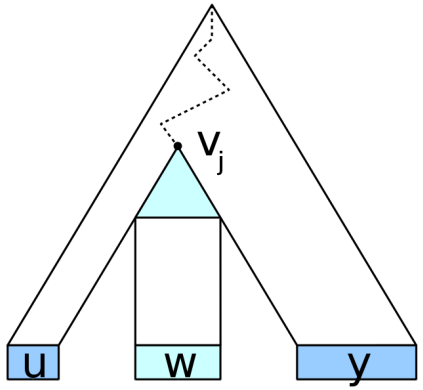
Pumping-Lemma

- Klasse regulärer Sprachen hat mit dem Pumping-Lemma ein Instrument, um die Nichtregularität von Sprachen zu zeigen.
- Ähnliches Lemma lässt sich auch für die Klasse der kontextfreien Sprachen herleiten.

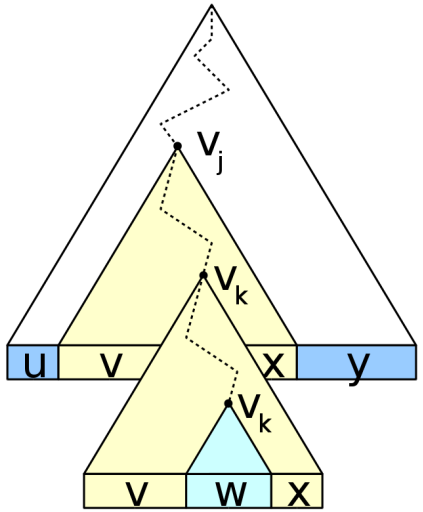


Gegeben: Wort $x \in L$ mit $|x| \geq n$

Ableitungsbaum T für x mit Höhe $h \geq N$



Erzeugen von uv^0wx^0y

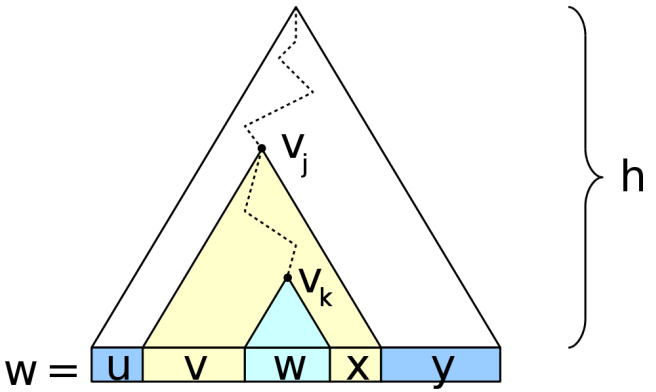


Erzeugen von uv^2wx^2y

https://de.wikipedia.org/wiki/Pumping-Lemma

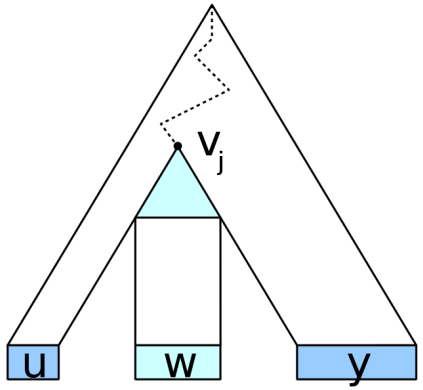
Pumping-Lemma

- Annahme: Grammatik $G = (V,\Sigma,P,S)$ ist in Chomsky-Normalform gegeben
- Einsetzen $j := 2^{|V|}$ und wählen eines beliebigen Worts $\omega \in L(G)$ mit $|\omega| \geq j$, sofern es existiert.
- Da Chomsky-Normalform gegeben ist, besitzt der zugehöri Syntaxbaum im Innern die Form eines Binärbaums mit mindestens $2^{|V|}$ Blättern.
- Zusammen mit dem Startsymbol findet man auf dem Pfad mindestens $|V| + 1$ Nonterminale, so dass eines davon mehrfach auftauchen muss.
- Bezeichnen wir dieses Nonterminal mit A, so lässt sich der Syntaxbaum in einer Form darstellen, wie in der Abbildung skizziert.
- Man erhält Zerlegung des abgeleiteten Wortes in fünf Segmente.

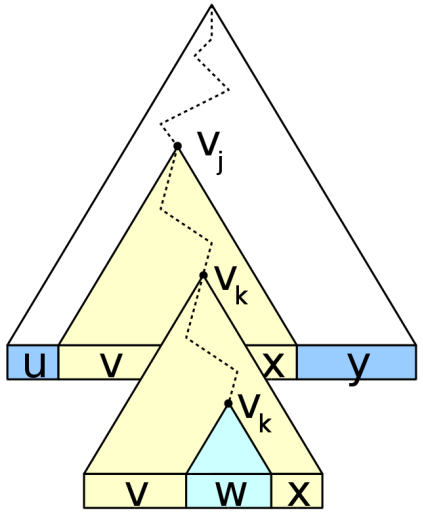


Gegeben: Wort $x \in L$ mit $|x| \geq n$

Ableitungsbaum T für x mit Höhe $h \geq N$



Erzeugen von uv^0wx^0y



Erzeugen von uv^2wx^2y

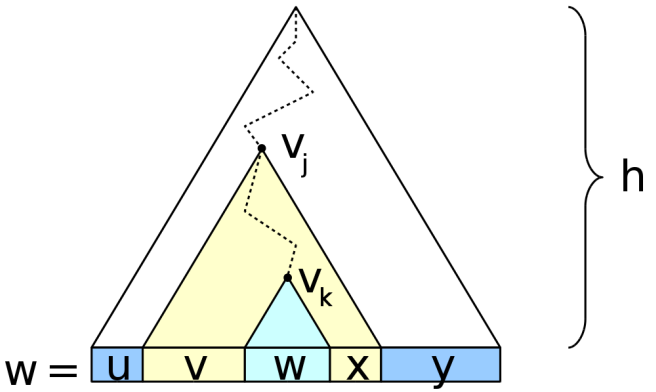
https://de.wikipedia.org/wiki/Pumping-Lemma

Pumping-Lemma

- Da G Chomsky-Normalform ist, kann aus A nur dann ein A erzeugt werden, wenn mindestens ein Ableitungsschritt der Form $A \rightarrow BC$ durchlaufen wurde.
- Es muss mindestens eines der Segmente v oder x ein Zeichen enthalten, d. h., es gilt die Beziehung: $|vx| \geq 1$.
- Über die Mindestlänge der Segmente u und y kann man keine Aussage machen; beide können leeres Wort werden
- Annahme, dass die zwei Vorkommen der Nonterminale A so gewählt wurden, dass alle weiteren Vorkommen von A, falls diese überhaupt existieren, näher an der Wurzel liegen und alle anderen Nonterminale, die näher an den Blättern liegen, paarweise verschieden sind.
- Hierdurch ist das ausgewählte A maximal $|V|$ Schritte von den Blättern entfernt und der aufgespannte Syntaxbaum kann höchstens $2^{|V|} = j$ Blätter enthalten.
- Damit gilt die Beziehung $|vwx| \leq j$.

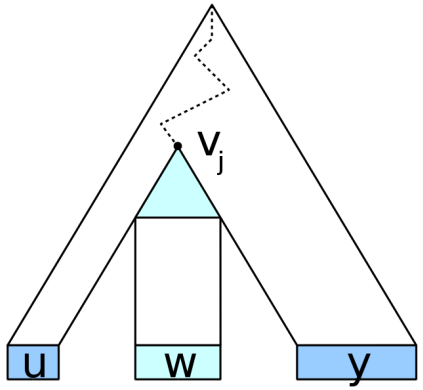
Pumping-Lemma

- Wie im Falle des Pumping-Lemmas für reguläre Sprachen erlaubt das Doppelvorkommen von *A*, das ableitbare Wort „aufzupumpen“, indem die Ableitungssequenz von *A* nach *A* wiederholt wird.
- Hierdurch können die Worte uv^2wx^2y, uv^3wx^3y, ... produziert und eine entsprechende Überlegung gezeigt werden, dass auch das Wort uv^0wx^0y ableitbar ist.

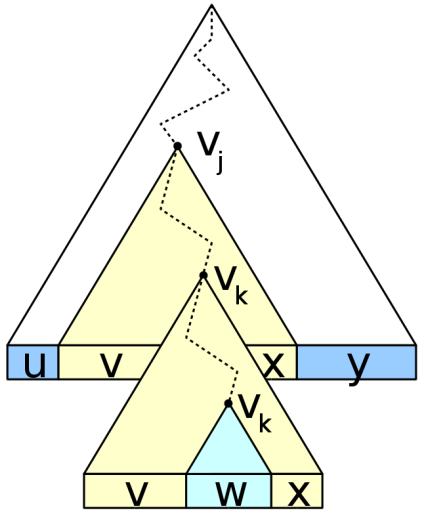


Gegeben: Wort $x \in L$ mit $|x| \geq n$

Ableitungsbaum T für x mit Höhe $h \geq N$



Erzeugen von uv^0wx^0y



Erzeugen von uv^2wx^2y

https://de.wikipedia.org/wiki/Pumping-Lemma

Pumping-Lemma für kontextfreie Sprachen

- Fasst man die erarbeiteten Ergebnisse zusammen, so erhält man in direkter Weise das Pumping-Lemma für kontextfreie Sprachen
- Für jede kontextfreie Sprache L existiert ein $j \in \mathbb{N}$, so dass sich alle Wörter $\omega \in L$ mit $|\omega| \geq j$ in der folgenden Form darstellen lassen:

$$\omega = uvwxy \qquad \text{mit } |vx| \geq 1 \text{und } |vwx| \leq j$$

- Mit ω ist auch das Wort uv^iwx^iy für alle $i \in \mathbb{N}$ in L enthalten.

Pumping-Lemma

- Pumping-Lemma versetzt in die Lage, die Sprache $L_{C1} = \{a^n b^n c^n \mid n \in \mathrm{N}+\}$ als nicht kontextfrei zu erkennen.
- Für den Beweis nimmt man an, L_{C1} sei kontextfrei.
- Dann garantiert das Pumping-Lemma, dass ein $j \in \mathrm{N}$ existiert, so dass sich jedes Wort $\omega = a^j b^j c^j$ mit $|\omega| \geq j$ in der Form $uvwxy$ darstellen lässt mit $|vx| \geq 1$ und $|vwx| \leq j$.
- Wählt man $i = j$, so kann das Segment vwx aufgrund seiner Längenbeschränkung nicht gleichzeitig a's und c's enthalten.
- Entfernt man die Segmente v und x aus ω, so entsteht mit uwy ein Wort, das eine ungleiche Anzahl von a's, b's und c's enthält.
- Nach dem Pumping-Lemma muss das Wort $uwy = uv^0wx^0y$ jedoch in L_{C1} enthalten sein, im Widerspruch zur Definition dieser Sprache.

Pumping-Lemma

- Pumping-Lemma hat sich als Hilfsmittel erwiesen, um die Sprache L_{C1} als nicht kontextfrei zu ermitteln.
- Dass sich das Pumping-Lemma nicht immer als geeignetes Mittel erweist, zeigt die Beispielsprache

  $L_f := \{b^k c^l d^m \mid k,l,m \in \mathbb{N}^+\} \cup \{a^m b^n c^n d^n \mid m,n \in \mathbb{N}^+\}$

- Obwohl L_f nicht kontextfrei ist, erfüllt die Sprache alle innerhalb des Pumping-Lemmas getroffenen Aussagen.
- Hierzu setzt man $j = 4$ und wählt für alle Wörter $\omega \in L_f$ mit $|\omega| \geq j$ eine Zerlegung.

Pumping-Lemma - Negativbeispiel

Mit der getroffenen Wahl von u,v,w,x und y gelten die folgenden Beziehungen:

- $|vx| \geq 1$
- $|vwx| \leq j$
- $uv^iwx^iy \in L$ für alle $i \in \mathbb{N}$

- Damit ist es unmöglich, die Sprache L_f mithilfe des Pumping-Lemmas von den kontextfreien Sprachen zu unterscheiden.
- Der Grund für das Versagen geht auf die Eigenschaft des Pumping-Lemmas zurück, keine Aussage über die Startposition der Teilwörter u, v, w, x und y zu machen.
- Dadurch ist es nicht möglich, den kritischen Teilabschnitt vwx in Wörtern der Form $a^mb^nc^nd^n$ komplett mit a's zu füllen und damit zu vermeiden, dass die Struktur des nicht kontextfreien Teilworts $b^nc^nd^n$ durch das Aufpumpen von v und x zerstört wird.

Ogdens Lemma für kontextfreie Sprachen

- Um die Sprache dennoch als nicht kontextfrei zu identifizieren, braucht es ein anderes Verfahren.
- Ein solches ist das *Ogdens Lemma*– eine Verallgemeinerung des Pumping-Lemmas, das die „pumpbaren“ Symbole freier wählen lässt:

Ogdens Lemma für kontextfreie Sprachen:

Für jede kontextfreie Sprache L existiert ein $j \in \mathbb{N}$, so dass alle Wörter $\omega \in L$ mit $|\omega| \geq j$ die folgende Eigenschaft erfüllen:

- Markiert man mindestens j Zeichen in ω, so lässt sich das Wort in der Form $\omega = uvwxy$ schreiben, so dass
  - mindestens ein Zeichen in vx markiert ist,
  - höchstens j Zeichen in vwx markiert sind,
  - und für alle $i \in \mathbb{N}$ gilt: $uv^iwx^iy \in L$

Ogdens Lemma für kontextfreie Sprachen

- Mithilfe von Ogdens Lemma kann man zeigen, dass L_f keine kontextfreie Sprache sein kann.
- Hierzu betrachten man das Wort $\omega = ab^jc^jd^j$, wobei j die Konstante aus Ogdens Lemma ist.
- Markiert man die Symbolsequenz bc^jd, so muss es für ω eine Zerlegung uvwxy geben, so dass vx mindestens einen und vwx höchstens j der markierten Buchstaben enthält.
- Damit kann die Sequenz vwx und damit auch die Sequenz vx maximal zwei verschiedene Buchstaben aus der Menge {b,c,d} enthalten.
- Da aber vx mindestens einen Buchstaben aus der Menge {b,c,d} enthält, kommt die Anzahl an b's, c's und d's im Wort uv^2wx^2y aus dem Gleichgewicht, so dass es kein Element von L sein kann.
- Nach Ogdens Lemma müsste uv^2wx^2y aber in L enthalten sein, falls L tatsächlich kontextfrei wäre.

Entscheidungsprobleme

Entscheidungsprobleme

- Wortproblem für kontextfreie Sprachen ist entscheidbar und lässt sich mit dem Mittel der dynamischen Programmierung effizient lösen.
- Anwenden lässt sich dynamische Programmierung immer dann, ***wenn sich ein Problem in kleinere Teile zerlegen*** lässt und die optimale Lösung des Gesamtproblems aus den optimalen Lösungen der Teilprobleme berechnet werden kann.
- Von der klassischen Rekursion unterscheidet sich die dynamische Programmierung durch den Einsatz einer Tabelle, in der sämtliche Zwischenergebnisse gespeichert werden.
- Durch das Vorhalten der bereits berechneten Zwischenlösungen konsumieren die Algorithmen mehr Speicherplatz als ihre rein rekursiv programmierten Pendants, so dass sich die dynamische Programmierung immer dann anbietet, wenn die Laufzeit und nicht der Speicherbedarf eines Algorithmus im Vordergrund steht.

Prominente Algorithmen

- Bekannte Algorithmen, die nach dem Prinzip der dynamischen Programmierung arbeiten, sind der Viterbi-Algorithmus und der Floyd-Warshall-Algorithmus.
- Die Wissenschaftler John **Cocke**, Daniel **Younger** und Tadao **Kasami** erkannten Ende der Sechzigerjahre unabhängig voneinander, dass sich das Wortproblem kontextfreier Sprachen mit dem Mittel der dynamischen Programmierung lösen lässt.
- **Ausgangspunkt** für den nach den Anfangsbuchstaben seiner Entdecker benannten **CYK-Algorithmus ist eine Grammatik G in Chomsky-Normalform**.

CYK-Algorithmus (Cocke, Younger, Kasami)

CYK-Algorithmus

- CYK-Algorithmus (Cocke, Younger, Kasami) ist ein effizienter Algorithmus für das Wortproblem kontextfreier Sprachen, wobei diese in Form von CNF Grammatiken gegeben sein müssen.
- **Grundidee:** Wort x = a der Länge 1 kann nur durch einmalige Anwendung einer Regel $A \rightarrow a$ abgeleitet werden, Wort x der Länge $n > 1$ nur durch Anwendung einer Regel $A \rightarrow BC$, wobei B Anfangsstück von x ableitet, und C Endstück ist.

CYK-Algorithmus

Im Kern basiert der CYK-Algorithmus auf der Beobachtung:

- Lässt sich aus einem Nonterminal A ein Wort ω ableiten, das aus einem einzelnen Terminalzeichen σ besteht, so muss die Regel $A \to \sigma$ existieren.
- Andernfalls würden die Produktionen mindestens ein weiteres Nonterminal und damit auch ein weiteres Terminalzeichen produzieren.
- Für den Fall $|\omega| = 1$ lässt sich das Wortproblem ohne Mühe entscheiden.
- Besteht das Wort ω aus mehreren Terminalzeichen $\sigma_1,\ldots,\sigma_n$ mit $n \geq 2$, so kann es aus einem Nonterminal A nur durch eine voran gegangene Anwendung einer Regel $A \to BC$ entstanden sein.
- Kann man nachweisen, dass ein gewisses k mit $1 \leq k < n$ existiert, so dass sich die Anfangssequenz $\sigma_1 \ldots \sigma_k$ aus B und die Endsequenz $\sigma_{k+1} \ldots \sigma_n$ aus C ableiten lässt, dann lässt sich das Gesamtwort ω aus A ableiten.

CYK-Algorithmus

- Damit gelingt es, das Problem für Wörter ω der Länge n auf die Lösung für Wörter der Länge k bzw. n − k zurückzuführen.
- Auch wenn man den Wert von k nicht kennen und alle Möglichkeiten in Betracht ziehen muss, ist sichergestellt, dass die Teilwörter eine kleinere Länge besitzen als ω selbst.
- Damit sind alle Voraussetzungen für die Anwendbarkeit der dynamischen Programmierung erfüllt.
- Um das Wortproblem für $\omega = \sigma_1,\ldots,\sigma_n$ zu entscheiden, verwaltet der CYK-Algorithmus intern ein zweidimensionales Array cyk der Größe n × n.
- Sobald der Algorithmus terminiert, enthält das Feld cyk[i][j] alle Nonterminale, aus denen sich das Teilwort $\sigma_i,\ldots,\sigma_{i+j-1}$ ableiten lässt.

CYK-Algorithmus

Offensichtlich gelten die folgenden Eigenschaften:

- Für i + j > n+1 kann cyk[i][j] niemals Nonterminale enthalten.
- Effektiv benötigt der CYK-Algorithmus damit nur etwas mehr als die Hälfte der Array-Felder.
- Alle anderen Felder brauchen nicht berücksichtigt zu werden.
- Das Wort ω lässt sich genau dann ableiten, wenn das Feld cyk[1][n] das Startsymbol S enthält.
- Damit reduziert sich das Wortproblem auf einen simplen Inklusionstest, sobald das Array cyk komplett aufgebaut ist.

CYK-Algorithmus Pseudocode

```
/ / Eingabe : Grammatik G = (V,Σ,P,S)
// Wort ω=σ1,...,σn
// Ausgabe: true , wenn ω ∈ L(G), false wenn ω ∉ L(G)

boolean cyk (G,ω)

{  // Berechne die erste Zeile ...

        for (i = 1; i ≤ n; i++) {

        cyk[i][1] ={A|(A→σi)∈P};
        }

// Berechne alle restlichen Zeilen ...

        for (j = 2; j ≤ n; j++) {

          for (i = 1; i ≤ n+1−j; i++) {

                  c y k [ i ] [ j ] = ∅;
                  for (k=1;k<j;k++){

                    cyk[i][j]=cyk[i][j] ∪ {A|(A→BC)∈P, B ∈ cyk[i][k], C ∈ cyk[i+k][j-k]};
                  }
          }
         }

 return S∈cyk[1][n];
}
```

CYK-Algorithmus

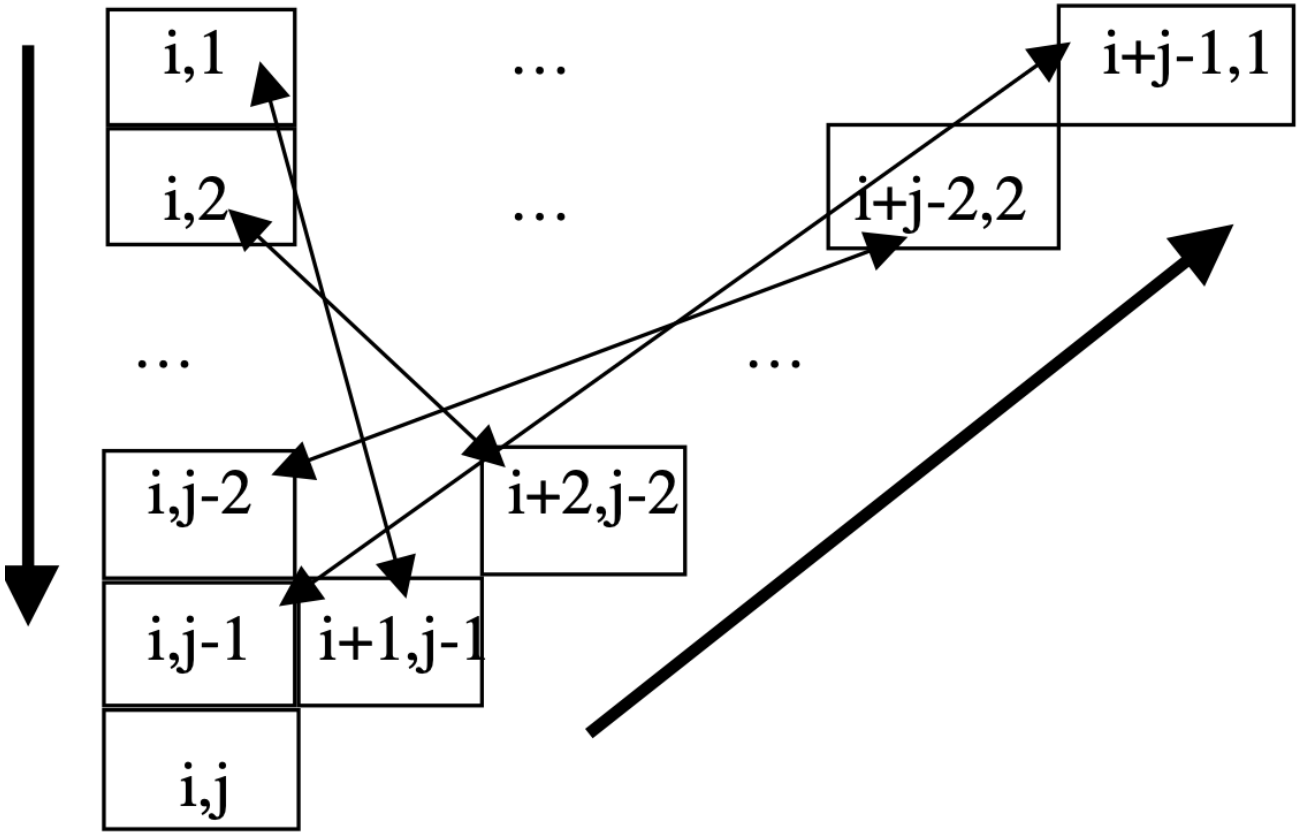
- Hauptarbeit des CYK-Algorithmus besteht im Ausfüllen der Array-Felder cyk[i][j].
- In zwei verschachtelten Schleifen iteriert der Algorithmus in gewöhnlicher Leserichtung von links nach rechts und von oben nach unten über die Array-Felder cyk[i][j].
- Für jedes Feld wird geprüft, ob eine Regel $A \rightarrow BC$ und ein k existieren, so dass sich das Teilwort $\sigma_i,\ldots,\sigma_{i+k-1}$ aus B und das Teilwort $\sigma_{i+k},\ldots,\sigma_{i+j-1}$ aus C erzeugen lässt.
- Dies ist genau dann der Fall, wenn das Nonterminal B in cyk[i,k] und das Nonterminal C in cyk[i+k,j−k] enthalten ist.
- Fällt der Test erfolgreich aus, so wird dem Element cyk[i][j] das Nonterminal A hinzugefügt.

CYK-Algorithmus

- Im Kern verbirgt sich hinter Implementierung nichts anderes als ein rekursiver Algorithmus.
- Effizienzgewinn resultiert aus der Eigenschaft, die Zwischenergebnisse tabellarisch zu speichern und den rekursiven Aufruf durch einen simplen Tabellenzugriff zu ersetzen.
- Während der rein rekursiv arbeitende Algorithmus eine exponentielle Laufzeit benötigen würde, macht es die dynamische Programmierung an dieser Stelle möglich, das Wortproblem in kubischer Laufzeit zu entscheiden.
- Erkauft wird Effizienzgewinn durch einen erhöhten Logistikbedarf, der den benötigten Speicherplatz des CYK-Algorithmus quadratisch mit der Länge des Eingabeworts wachsen lässt.

CYK-Algorithmus Tabelle

Um die Tabelleneintrag i,j zu bestimmen, müssen Paare von Einträgen verglichen werden



CYK-Algorithmus

Beispiel:

S → AB

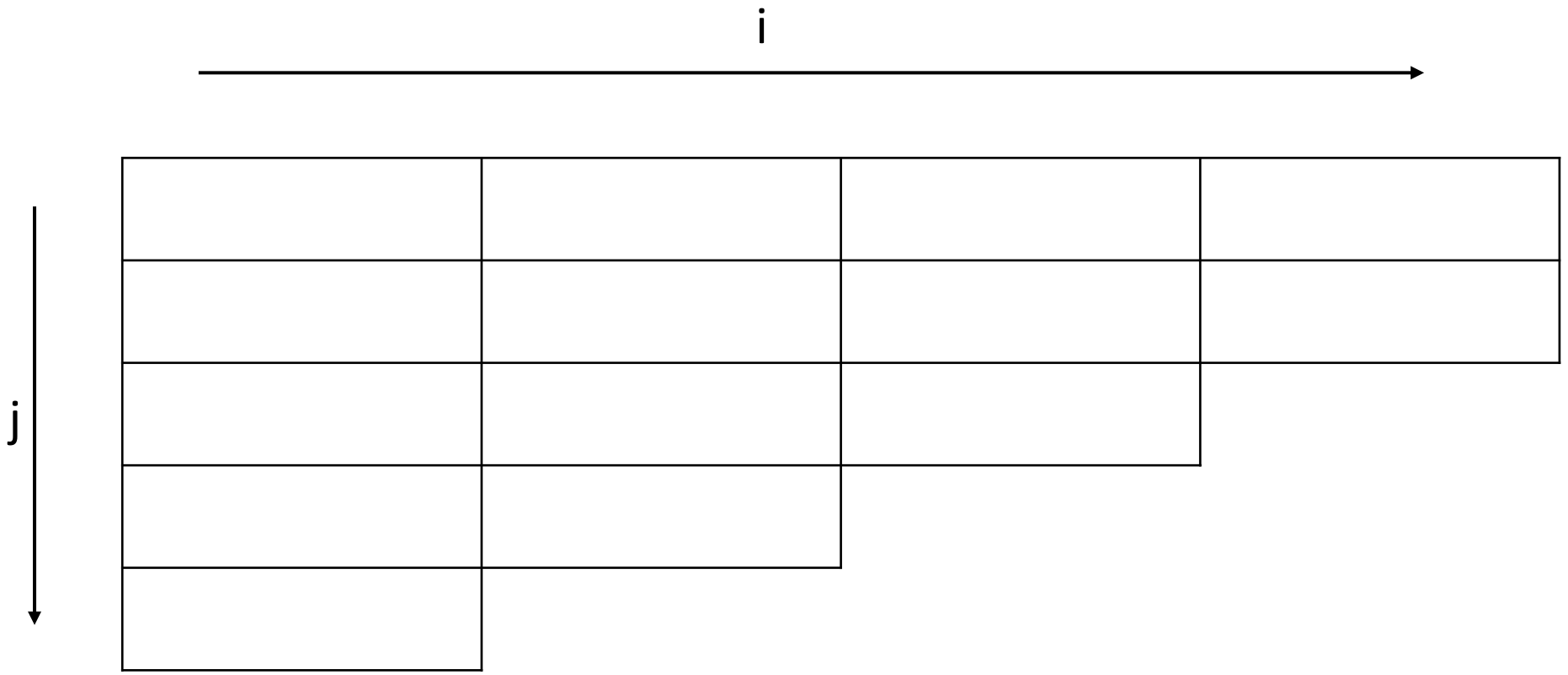
S → AC

C → SB

A → a

B → b

zu testen: aabb



Übung: L = {$a^n b^m c^m$ | m,n > 0}

Beispielwort: abbcc

S → AD

A → AA

D → BC

D → BE

E → DC

A → a

B → b

C → c