Info III Tutorium

Thomas Pajor

ITI Sanders

12. Dezember 2006

Punkteverteilung

Übungsblatt 6

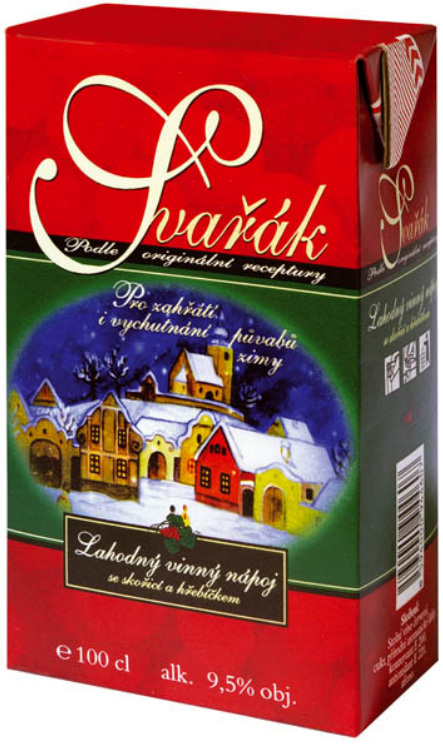
- Aufgabe 1a – 3P
- Aufgabe 1b – 5P
- Aufgabe 2a – 2P
- Aufgabe 2b – 2P
- Aufgabe 2c – 8P
- Aufgabe 2d – 2P

$\Rightarrow$ 22 Punkte insgesamt

Glüüüüüüüüüühwein

Nächste Woche ist Glühwein–Tut!

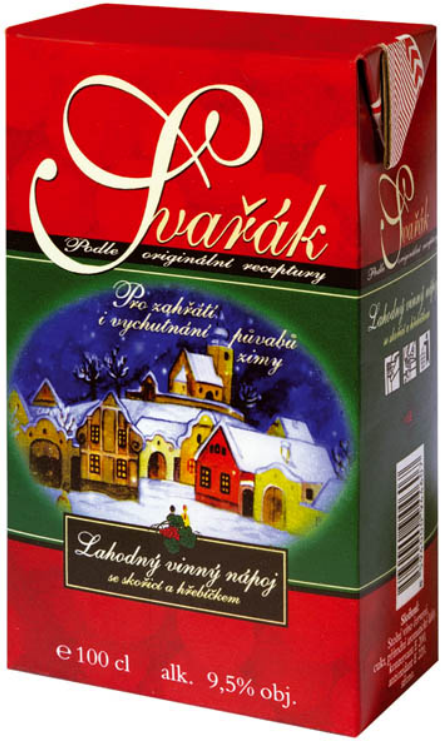
- Ich bringe den Glühwein!
- Bringt Tassen bitte selber mit
- Wenn ihr wollt, könnt ihr auch Plätzchen und Gebäck mitbringen, damit es etwas gemütlicher wird



Glüüüüüüüüüühwein

Nächste Woche ist Glühwein–Tut!

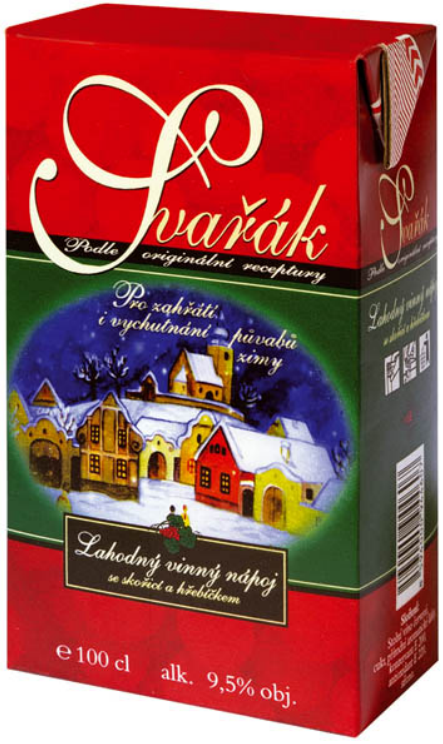
- Ich bringe den Glühwein!
- Bringt Tassen bitte selber mit
- Wenn ihr wollt, könnt ihr auch Plätzchen und Gebäck mitbringen, damit es etwas gemütlicher wird



Glüüüüüüüüüühwein

Nächste Woche ist Glühwein–Tut!

- Ich bringe den Glühwein!
- Bringt Tassen bitte selber mit
- Wenn ihr wollt, könnt ihr auch Plätzchen und Gebäck mitbringen, damit es etwas gemütlicher wird



Aufgabe 1

Aufgabe

Gegeben sei folgende kontextfreie Grammatik
$G := (V := \{S\}, \Sigma := \{a, b\}, P, S)$ mit

$$P := \{S \to SS, \quad (1)$$
$$S \to aSb, \quad (2)$$
$$S \to ab\} \quad (3)$$

(a) Wandeln Sie G in Chomsky–Normalform G'.

(b) Prüfen Sie mit dem Algorithmus von Cocke–Younger–Kasami ob die Wörter $w_1 := ababaabb$ und $w_2 := ababb$ in $L(G)$ enthalten sind und geben Sie ggf. eine Ableitungsfolge $S \overset{*}{\Rightarrow} w$ an.

Chomsky–Normalform (Wiederholung)

Definition

Eine kontextfreie Grammatik G ist in *Chomsky–Normalform*, falls alle Produktionen folgende Form haben:

$$A \to BC \qquad \text{oder} \qquad A \to a$$

wobei $A, B, C \in V$ und $a \in \Sigma$.

Zur Herleitung der Chomsky–Normalform aus einer beliebigen kontextfreien Grammatik gibt es ein systematisches Verfahren.

Verfahren zur Herleitung der CNF

Schritt 1: Elimination von Terminalzeichen auf der rechten Seite

- Füge für jedes $x_i \in \Sigma$ eine neue Variable X_i in V hinzu.
- Ersetze in allen Produktionen jedes Vorkommen von x_i durch X_i.
- Füge für jedes $x_i \in \Sigma$ eine neue Produktion $X_i \longrightarrow x_i$ hinzu.

Verfahren zur Herleitung der CNF

Schritt 1: Elimination von Terminalzeichen auf der rechten Seite

- Füge für jedes $x_i \in \Sigma$ eine neue Variable X_i in V hinzu.
- Ersetze in allen Produktionen jedes Vorkommen von x_i durch X_i.
- Füge für jedes $x_i \in \Sigma$ eine neue Produktion $X_i \longrightarrow x_i$ hinzu.

Verfahren zur Herleitung der CNF

Schritt 1: Elimination von Terminalzeichen auf der rechten Seite

- Füge für jedes $x_i \in \Sigma$ eine neue Variable X_i in V hinzu.
- Ersetze in allen Produktionen jedes Vorkommen von x_i durch X_i.
- Füge für jedes $x_i \in \Sigma$ eine neue Produktion $X_i \longrightarrow x_i$ hinzu.

Verfahren zur Herleitung der CNF

Schritt 2: Rechte Seiten verkürzen

Betrachte jede Regel der Form

$$A \to B_1 \dots B_m$$

mit $m > 2$ und $A, B_i \in V$. Führe $m-2$ neue Variablen $C_1, \dots, C_{m-2}$ ein und ersetze die obige Regel durch folgende Regeln:

$$\begin{aligned} A &\to B_1 C_1 \\ C_i &\to B_{i+1} C_{i+1} \qquad \text{für } 1 \leq i \leq m-3 \\ C_{m-2} &\to B_{m-1} B_m \end{aligned}$$

Verfahren zur Herleitung der CNF

Schritt 2: Rechte Seiten verkürzen

Betrachte jede Regel der Form

$$A \to B_1 \dots B_m$$

mit $m > 2$ und $A, B_i \in V$. Führe $m - 2$ neue Variablen $C_1, \dots, C_{m-2}$ ein und ersetze die obige Regel durch folgende Regeln:

$$\begin{aligned} A &\to B_1 C_1 \\ C_i &\to B_{i+1} C_{i+1} \qquad \text{für } 1 \leq i \leq m-3 \\ C_{m-2} &\to B_{m-1} B_m \end{aligned}$$

Verfahren zur Herleitung der CNF

Schritt 3: Zyklen bei Kettenregeln löschen

Verfolge alle Kettenregeln bis sich ein Kreis bildet, also betrachte

$$A_1 \to A_2 \to \ldots \to A_r \to A_1$$

Ersetze in allen Produktionen die Variablen $A_2, \ldots, A_r$ durch A_1 und lösche die Produktion $A_1 \to A_1$.

Verfahren zur Herleitung der CNF

Schritt 3: Zyklen bei Kettenregeln löschen

Verfolge alle Kettenregeln bis sich ein Kreis bildet, also betrachte

$$A_1 \to A_2 \to \ldots \to A_r \to A_1$$

Ersetze in allen Produktionen die Variablen $A_2, \ldots, A_r$ durch A_1 und lösche die Produktion $A_1 \to A_1$.

Verfahren zur Herleitung der CNF

Schritt 4: Sonstige Kettenproduktionen löschen

Wir nummerieren alle Variablen durch, also $V = \{A_1, \ldots, A_n\}$ wobei aus der Existenz einer Produktion $A_i \to A_j$ folgen soll, dass $i < j$ (Topologisches Sortieren).

Für jedes $k = n-1, \ldots, 1$ lösche jede Regel der Form $A_k \to A_l$ wobei $l > k$, und füge zu jeder Regel $A_l \to X$ (X beliebig) eine Regel $A_k \to X$ hinzu.

Verfahren zur Herleitung der CNF

Schritt 4: Sonstige Kettenproduktionen löschen

Wir nummerieren alle Variablen durch, also $V = \{A_1, \ldots, A_n\}$ wobei aus der Existenz einer Produktion $A_i \to A_j$ folgen soll, dass $i < j$ (Topologisches Sortieren).
Für jedes $k = n-1, \ldots, 1$ lösche jede Regel der Form $A_k \to A_l$ wobei $l > k$, und füge zu jeder Regel $A_l \to X$ (X beliebig) eine Regel $A_k \to X$ hinzu.

Verfahren zur Herleitung der CNF

Schritt 4: Sonstige Kettenproduktionen löschen

Wir nummerieren alle Variablen durch, also $V = \{A_1, \ldots, A_n\}$ wobei aus der Existenz einer Produktion $A_i \to A_j$ folgen soll, dass $i < j$ (Topologisches Sortieren).
Für jedes $k = n-1, \ldots, 1$ lösche jede Regel der Form $A_k \to A_l$ wobei $l > k$, und füge zu jeder Regel $A_l \to X$ (X beliebig) eine Regel $A_k \to X$ hinzu.

Cocke–Younger–Kasami Algorithmus

Eingabe : Eine Grammatik $G = (V, \Sigma, P, S)$ in CNF und $w \in \Sigma^*$
Ausgabe : $w \in L(G)$?

$T \leftarrow n \times n$ Tabelle, initialisiert mit $\emptyset$

für $j \leftarrow 1 \dots n$ **tue**
  $T[1, j] \leftarrow \{A \mid A \rightarrow w_j \in P\}$

für $i \leftarrow 1 \dots n$ **tue**
  für $j \leftarrow 1 \dots (n - i + 1)$ **tue**
    für $k \leftarrow 1 \dots (i - 1)$ **tue**
      $T[i, j] \leftarrow T[i, j] \cup \{A \mid A \rightarrow BC \in P$
      und $B \in T[k, j]$
      und $C \in T[i - k, j + k]\}$

wenn $S \in T[n, 1]$ **dann**
  return „w *ist in* $L(G)$“

Cocke–Younger–Kasami Algorithmus

Eingabe : Eine Grammatik $G = (V, \Sigma, P, S)$ in CNF und $w \in \Sigma^*$
Ausgabe : $w \in L(G)$?

$T \leftarrow n \times n$ Tabelle, initialisiert mit $\emptyset$

für $j \leftarrow 1 \dots n$ **tue**
  $T[1, j] \leftarrow \{A \mid A \rightarrow w_j \in P\}$

für $i \leftarrow 1 \dots n$ **tue**
  für $j \leftarrow 1 \dots (n - i + 1)$ **tue**
    für $k \leftarrow 1 \dots (i - 1)$ **tue**
      $T[i, j] \leftarrow T[i, j] \cup \{A \mid A \rightarrow BC \in P$
      und $B \in T[k, j]$
      und $C \in T[i - k, j + k]\}$

wenn $S \in T[n, 1]$ **dann**
  return „w *ist in* $L(G)$“

Cocke–Younger–Kasami Algorithmus

Eingabe : Eine Grammatik $G = (V, \Sigma, P, S)$ in CNF und $w \in \Sigma^*$
Ausgabe : $w \in L(G)$?

$T \leftarrow n \times n$ Tabelle, initialisiert mit $\emptyset$

für $j \leftarrow 1 \dots n$ **tue**
  $T[1, j] \leftarrow \{A \mid A \rightarrow w_j \in P\}$

für $i \leftarrow 1 \dots n$ **tue**
  für $j \leftarrow 1 \dots (n - i + 1)$ **tue**
    für $k \leftarrow 1 \dots (i - 1)$ **tue**
      $T[i, j] \leftarrow T[i, j] \cup \{A \mid A \rightarrow BC \in P$
      und $B \in T[k, j]$
      und $C \in T[i - k, j + k]\}$

wenn $S \in T[n, 1]$ **dann**
  return „w *ist in* $L(G)$“

Cocke–Younger–Kasami Algorithmus

Eingabe : Eine Grammatik $G = (V, \Sigma, P, S)$ in CNF und $w \in \Sigma^*$
Ausgabe : $w \in L(G)$?

$T \leftarrow n \times n$ Tabelle, initialisiert mit $\emptyset$

für $j \leftarrow 1 \ldots n$ **tue**
  $T[1, j] \leftarrow \{A \mid A \rightarrow w_j \in P\}$

für $i \leftarrow 1 \ldots n$ **tue**
  für $j \leftarrow 1 \ldots (n - i + 1)$ **tue**
    für $k \leftarrow 1 \ldots (i - 1)$ **tue**
      $T[i, j] \leftarrow T[i, j] \cup \{A \mid A \rightarrow BC \in P$
        und $B \in T[k, j]$
        und $C \in T[i - k, j + k]\}$

wenn $S \in T[n, 1]$ **dann**
  return „w *ist in* $L(G)$“

Cocke–Younger–Kasami Algorithmus

Eingabe : Eine Grammatik $G = (V, \Sigma, P, S)$ in CNF und $w \in \Sigma^*$
Ausgabe : $w \in L(G)$?

$T \leftarrow n \times n$ Tabelle, initialisiert mit $\emptyset$

für $j \leftarrow 1 \ldots n$ **tue**
  $T[1, j] \leftarrow \{A \mid A \rightarrow w_j \in P\}$

für $i \leftarrow 1 \ldots n$ **tue**
  für $j \leftarrow 1 \ldots (n - i + 1)$ **tue**
    für $k \leftarrow 1 \ldots (i - 1)$ **tue**
      $T[i, j] \leftarrow T[i, j] \cup \{A \mid A \rightarrow BC \in P$
        und $B \in T[k, j]$
        und $C \in T[i - k, j + k]\}$

wenn $S \in T[n, 1]$ **dann**
  return „w *ist in* $L(G)$“

Cocke–Younger–Kasami Algorithmus

Eingabe : Eine Grammatik $G = (V, \Sigma, P, S)$ in CNF und $w \in \Sigma^*$
Ausgabe : $w \in L(G)$?

$T \leftarrow n \times n$ Tabelle, initialisiert mit $\emptyset$

für $j \leftarrow 1 \ldots n$ **tue**
  $T[1, j] \leftarrow \{A \mid A \rightarrow w_j \in P\}$

für $i \leftarrow 1 \ldots n$ **tue**
  für $j \leftarrow 1 \ldots (n - i + 1)$ **tue**
    für $k \leftarrow 1 \ldots (i - 1)$ **tue**
      $T[i, j] \leftarrow T[i, j] \cup \{A \mid A \rightarrow BC \in P$ und $B \in T[k, j]$ und $C \in T[i - k, j + k]\}$

wenn $S \in T[n, 1]$ **dann**
  return „w *ist in* $L(G)$“

Cocke–Younger–Kasami Algorithmus

Eingabe : Eine Grammatik $G = (V, \Sigma, P, S)$ in CNF und $w \in \Sigma^*$
Ausgabe : $w \in L(G)$?

$T \leftarrow n \times n$ Tabelle, initialisiert mit $\emptyset$

für $j \leftarrow 1 \ldots n$ **tue**
  $T[1, j] \leftarrow \{A \mid A \rightarrow w_j \in P\}$

für $i \leftarrow 1 \ldots n$ **tue**
  für $j \leftarrow 1 \ldots (n - i + 1)$ **tue**
    für $k \leftarrow 1 \ldots (i - 1)$ **tue**
      $T[i, j] \leftarrow T[i, j] \cup \{A \mid A \rightarrow BC \in P$ und $B \in T[k, j]$ und $C \in T[i - k, j + k]\}$

wenn $S \in T[n, 1]$ **dann**
  return „w *ist in* $L(G)$“

Aufgabe 2

Aufgabe

Zeigen oder widerlegen Sie: Die Sprache

$$L := \{ww \mid w \in \{0,1\}^*\}$$

ist kontextfrei.

Aufgabe 3

Aufgabe

Konstruieren Sie eine (deterministische) Turingmaschine $\mathcal{M}$ die zu einer auf dem Band stehenden Binärzahl $w \in \{0,1\}^*$ (binär) eins addiert. Dabei sei die Eingabe w frei von führenden Nullen.