Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik

Vorlesung am 24. November 2016

INSTITUT FÜR THEORETISCHE INFORMATIK

Große Frage der Theoretischen Informatik

- Trivialerweise gilt: $\mathcal{P} \subseteq \mathcal{NP}$.
- **Frage:** Gilt $\mathcal{P} \subset \mathcal{NP}$ oder $\mathcal{P} = \mathcal{NP}$?
- Die Vermutung ist, dass $\mathcal{P} \neq \mathcal{NP}$ gilt.

- Dazu betrachten wir Probleme, die zu den schwersten Problemen in $\mathcal{NP}$ gehören.

- Dabei ist am schwersten im folgenden Sinne gemeint:
- Wenn ein solches Problem trotzdem in $\mathcal{P}$ liegt, so kann man folgern, dass alle Probleme aus $\mathcal{NP}$ in $\mathcal{P}$ liegen, d.h. $\mathcal{P} = \mathcal{NP}$.

- Diese Probleme sind also Kandidaten, um $\mathcal{P}$ und $\mathcal{NP}$ zu trennen.
- Es wird sich zeigen, dass alle diese schwersten Probleme im wesentlichen gleich schwer sind.

NP-Vollständigkeit

Eine **polynomiale Transformation** einer Sprache $L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f\colon \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:

- es existiert eine polynomiale deterministische Turing-Maschine, die f berechnet;
- für alle $x \in \Sigma_1^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$.

Wir schreiben dann $L_1 \propto L_2$ (L_1 ist polynomial transformierbar in L_2).

Eine Sprache L heißt $\mathcal{NP}$**–vollständig** , falls gilt:

- $L \in \mathcal{NP}$ und
- für alle $L' \in \mathcal{NP}$ gilt $L' \propto L$.

NP-Vollständigkeit

Wir formulieren nun die Begriffe polynomial transformierbar und $\mathcal{NP}$-vollständig für Entscheidungsprobleme.

Ein Entscheidungsproblem Π_1 ist **polynomial transformierbar** in das Entscheidungsproblem Π_2, wenn eine Funktion $f\colon D_{\Pi_1} \to D_{\Pi_2}$ existiert mit folgenden Eigenschaften:

- f ist durch einen polynomialen Algorithmus berechenbar;
- $\forall\, I \in D_{\Pi_1} : I \in J_{\Pi_1} \iff f(I) \in J_{\Pi_2}$.

Wir schreiben dann $\Pi_1 \propto \Pi_2$.

Ein Entscheidungsproblem Π heißt $\mathcal{NP}$**-vollständig**, falls gilt:

- $\Pi \in \mathcal{NP}$ und
- für alle $\Pi' \in \mathcal{NP}$ gilt $\Pi' \propto \Pi$.

Transitivität der poly. Transformation

Eine **polynomiale Transformation** einer Sprache $L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f\colon \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:

- es existiert eine polynomiale deterministische Turing-Maschine, die f berechnet;
- für alle $x \in \Sigma_1^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$.

Wir schreiben dann $L_1 \propto L_2$ (L_1 ist polynomial transformierbar in L_2).

Lemma. $\propto$ ist transitiv, d.h. aus $L_1 \propto L_2$ und $L_2 \propto L_3$ folgt $L_1 \propto L_3$.

Beweis. Die Hintereinanderausführung zweier polynomialer Transformationen ist wieder eine polynomiale Transformation.

Korollar

Korollar. Falls $L_1, L_2 \in \mathcal{NP}$, $L_1 \propto L_2$ und L_1 $\mathcal{NP}$-vollständig, dann ist auch L_2 $\mathcal{NP}$-vollständig.

Bedeutung.
Um also zu zeigen, dass ein Entscheidungsproblem Π $\mathcal{NP}$-vollständig ist, gehen wir folgendermaßen vor. Wir beweisen:

- $\Pi \in \mathcal{NP}$
- für ein bekanntes $\mathcal{NP}$-vollständiges Problem Π' gilt: $\Pi' \propto \Pi$.

Problem.

- Wir wissen noch für kein einziges Problem, dass es $\mathcal{NP}$-vollständig ist.
- Das erste $\mathcal{NP}$-vollständige Problem ist das Erfüllbarkeitsproblem SAT (satisfiability).

Das Problem SAT (satisfiability)

Sei $U = \{u_1, \ldots, u_m\}$ eine Menge von booleschen Variablen.
Es heißen $u_i, \overline{u_i}$ Literale.
Eine Wahrheitsbelegung für U ist eine Funktion $t \colon U \to \{\texttt{wahr}, \texttt{falsch}\}$.
Eine **Klausel** ist ein Boole'scher Ausdruck der Form

$$y_1 \vee \ldots \vee y_s \quad \text{mit} \quad y_i \in \underbrace{\{u_1, \ldots, u_m\} \cup \{\overline{u_1}, \ldots, \overline{u_m}\}}_{\text{Literalmenge}} \cup \{\texttt{wahr}, \texttt{falsch}\}$$

Problem SAT
Gegeben: Menge U von Variablen, Menge C von Klauseln über U.
Frage: Existiert eine Wahrheitsbelegung von U, so dass C erfüllt wird, d.h. dass alle Klauseln aus C den Wahrheitswert $\texttt{wahr}$ annehmen?

Beispiel: $U = \{u_1, u_2\}$ mit $C = \{u_1 \vee \overline{u_2}, \ \overline{u_1} \vee u_2\}$ ist Ja-Beispiel von SAT. Mit der Wahrheitsbelegung $t(u_1) = t(u_2) = \texttt{wahr}$ wird C erfüllt.

Weitere Beispiele für SAT-Instanzen

Lösbar:

$$U = \{a, b, c, d, e\}, C = \{c \vee \overline{d}, \overline{a} \vee b \vee \overline{c} \vee d \vee e, \overline{c} \vee d\}$$

Nicht lösbar:

$$U = \{a, b, c\}, C = \{a \vee b, \overline{a}, \overline{b} \vee c, \overline{c}\}$$

Der Satz von Cook (Steven Cook, 1971)

Satz:
SAT ist $\mathcal{NP}$-vollständig.

Der Satz von Cook (Steven Cook, 1971)

Satz:
SAT ist $\mathcal{NP}$-vollständig.

Beweis:

- SAT $\in \mathcal{NP}$ ist erfüllt:
  Für ein Beispiel I von SAT (mit n Klauseln und m Variablen) und einer Wahrheitsbelegung t kann in $O(m \cdot n)$ überprüft werden, ob t alle Klauseln erfüllt, d.h. ob I ein Ja–Beispiel ist.
- Wir müssen zeigen, dass für jede Sprache $L \in \mathcal{NP}$ gilt: $L \propto L_{\text{SAT}}$, wobei $L_{\text{SAT}} = L[\text{SAT}, s]$ für ein geeignetes Kodierungsschema s ist.

Beweis

Wir müssen zeigen, dass für jede Sprache $L \in \mathcal{NP}$ gilt: $L \propto L_{\text{SAT}}$, wobei $L_{\text{SAT}} = L[\text{SAT}, s]$ für ein geeignetes Kodierungsschema s ist.

- Dazu muss für alle Sprachen $L \in \mathcal{NP}$ eine polynomiale Transformation f_L angegeben werden, für die gilt, dass für alle $x \in \Sigma^*$ (Σ Alphabet zu L) gilt

$$x \in L \Longleftrightarrow f_L(x) \in L_{\text{SAT}}.$$

- Wir benutzen, dass es eine NDTM $\mathcal{M}$ zu L gibt, die L in polynomialer Laufzeit entscheidet.
- $\mathcal{M}$ sei gegeben durch $(Q, \Sigma, \sqcup, \Gamma, q_0, \delta, q_J, q_N)$ und akzeptiere die Sprache $L = L_{\mathcal{M}}$ in der Laufzeit $T_{\mathcal{M}} \leq p(n)$, wobei p ein Polynom ist. O.B.d.A. gilt $p(n) \geq n$.

Beweis

- Sei x eine Eingabe mit $n := |x|$
- Bei einer akzeptierenden Berechnung von $\mathcal{M}$ für $x \in \Sigma^*$ ist die Anzahl der Berechnungsschritte beschränkt durch $p(n)$.
- An einer so beschränkten Berechnung können höchstens die Zellen $-p(n)$ bis $p(n) + 1$ des Bandes beteiligt sein.

Der Zustand der deterministischen Stufe ist zu jedem Zeitpunkt eindeutig festgelegt durch:

- den jeweiligen Bandinhalt dieser $-p(n)$ bis $p(n) + 1$ Plätze,
- den Zustand der endlichen Kontrolle
- und der Position des Lese-/Schreibkopfs.

Im folgenden beschreiben wir eine Berechnung vollständig durch Variablen einer Instanz von SAT.

Beweis: Konstruktion der Variablen

Bezeichne

- die Zustände aus Q durch $q_0, q_1 = q_J, q_2 = q_N, q_3, \ldots, q_r$
- die Symbole aus Γ durch $s_0 = \sqcup, s_1, \ldots, s_\ell$ mit $|\Gamma| = \ell + 1$.

Es gibt drei Typen von Variablen in der zugehörigen SAT-Instanz

Variable	Gültigkeitsbereich	Bedeutung
$Q[i, k]$	$0 \leq i \leq p(n)$ $0 \leq k \leq r$	zum Zeitpunkt i der Über–prüfungsphase ist $\mathcal{M}$ in Zustand q_k
$H[i, j]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n) + 1$	zum Zeitpunkt i der Über–prüfungsphase ist der Lese–/ Schreibkopf an Position j des Bandes
$S[i, j, k]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n) + 1$ $0 \leq k \leq \ell$	zum Zeitpunkt i der Über–prüfungsphase ist der Bandinhalt an Position j das Symbol s_k

Beweis: Konstruktion der Variablen

Variable	Gültigkeitsbereich	Bedeutung
$Q[i,k]$	$0 \leq i \leq p(n)$ $0 \leq k \leq r$	zum Zeitpunkt i der Über–prüfungsphase ist $\mathcal{M}$ in Zustand q_k
$H[i,j]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n)+1$	zum Zeitpunkt i der Über–prüfungsphase ist der Lese–/ Schreibkopf an Position j des Bandes
$S[i,j,k]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n)+1$ $0 \leq k \leq \ell$	zum Zeitpunkt i der Über–prüfungsphase ist der Bandinhalt an Position j das Symbol s_k

- Eine Berechnung von $\mathcal{M}$ induziert in kanonischer Weise eine Wahrheitsbelegung dieser Variablen.
- Wir benutzen folgende Konvention:
- Falls $\mathcal{M}$ vor dem Zeitpunkt $p(n)$ stoppt, bleibt $\mathcal{M}$ in allen folgenden Zuständen in demselben Zustand und der Bandinhalt unverändert.

Beweis: Konstruktion der Variablen

Variable	Gültigkeitsbereich	Bedeutung
$Q[i,k]$	$0 \leq i \leq p(n)$ $0 \leq k \leq r$	zum Zeitpunkt i der Über–prüfungsphase ist $\mathcal{M}$ in Zustand q_k
$H[i,j]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n)+1$	zum Zeitpunkt i der Über–prüfungsphase ist der Lese–/ Schreibkopf an Position j des Bandes
$S[i,j,k]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n)+1$ $0 \leq k \leq \ell$	zum Zeitpunkt i der Über–prüfungsphase ist der Bandinhalt an Position j das Symbol s_k

Der Bandinhalt zum Zeitpunkt 0 der Überprüfungsphase sei

- Eingabe x auf Platz 1 bis n
- Orakel w auf Platz -1 bis $-|w|$
- ansonsten Blanks.

Beweis

- Eine beliebige Wahrheitsbelegung muss nicht notwendigerweise eine Berechnung induzieren (zum Beispiel $Q[i,k] = Q[i,\ell]$ für $k \neq \ell$).

Also konstruiere Transformation f_L die Klauseln einführt, so dass äquivalent ist:

- Für Eingabe x gibt es eine akzeptierende Berechnung, deren Überprüfungsphase höchstens $p(n)$ Zeit benötigt, und deren Orakel höchstens Länge $p(n)$ hat.
- Es gibt eine erfüllende Belegung für die SAT-Instanz $f_L(x)$.

Beweis

Also konstruiere Transformation f_L die Klauseln einführt, so dass äquivalent ist:

- Für Eingabe x gibt es eine akzeptierende Berechnung, deren Überprüfungsphase höchstens $p(n)$ Zeit benötigt, und deren Orakel höchstens Länge $p(n)$ hat.
- Es gibt eine erfüllende Belegung für die SAT-Instanz $f_L(x)$.

Damit können wir dann schließen:

$x \in L \Leftrightarrow$ es existiert eine akzeptierende Berechnung von $\mathcal{M}$ bei Eingabe x

$\Leftrightarrow$ es existiert eine akzeptierende Berechnung von $\mathcal{M}$ bei Eingabe x mit höchstens $p(n)$ Schritten in der Überprüfungsphase und einem Orakel w der Länge $|w| = p(n)$

$\Leftrightarrow$ es existiert eine erfüllende Wahrheitsbelegung für die Klauselmenge $f_L(x)$

Beweis

Karlsruher Institut für Technologie

Konvention:

- Die Bewegungsrichtung des Kopfes sei $d \in \{-1, 0, 1\}$.

Beweis: Konstruktion der Klauseln - Übersicht

Klausel-gruppe	Einschränkung / Bedeutung
G_1	Zum Zeitpunkt i ist $\mathcal{M}$ in genau einem Zustand.
G_2	Zum Zeitpunkt i hat der Lese-/Schreibkopf genau eine Position.
G_3	Zum Zeitpunkt i enthält jede Bandstelle genau ein Symbol aus Γ.
G_4	Festlegung der Anfangskonfiguration zum Zeitpunkt 0: $\mathcal{M}$ ist im Zustand q_0, der Lese-/Schreibkopf steht an Position 1 des Bandes; in den Zellen 1 bis n steht das Wort $x = s_{k_1} \dots s_{k_n}$
G_5	Bis zum Zeitpunkt $p(n)$ hat $\mathcal{M}$ den Zustand q_J erreicht.
G_6	Zu jedem Zeitpunkt i folgt die Konfiguration von $\mathcal{M}$ zum Zeitpunkt $i + 1$ aus einer einzigen Anwendung von δ aus der Konfiguration von $\mathcal{M}$ zum Zeitpunkt i.

Klauselgruppe 1:

Zum Zeitpunkt *i* ist $\mathcal{M}$ in genau einem Zustand.

Konstruktion:

- Zu jedem Zeitpunkt *i* ist $\mathcal{M}$ in mindestens einem Zustand

$$Q[i,0] \vee \ldots \vee Q[i,r] \quad \text{für } 0 \leq i \leq p(n)$$

- Zu jedem Zeitpunkt i ist $\mathcal{M}$ in nicht mehr als einem Zustand

$$\overline{Q[i,j]} \vee \overline{Q[i,j']} \quad \text{für } 0 \leq i \leq p(n),\ 0 \leq j < j' \leq r$$

Klauselgruppe 2:

Zum Zeitpunkt *i* hat der Lese-/Schreibkopf genau eine Position

Konstruktion:

- Zum Zeitpunkt *i* hat der Lese-/Schreibkopf mindestens eine Position

$$H[i, -p(n)] \vee \ldots \vee H[i, p(n)+1] \quad \text{für } 0 \leq i \leq p(n)$$

- Zum Zeitpunkt *i* hat der Lese-/Schreibkopf höchstens eine Position

$$\overline{H[i,j]} \vee \overline{H[i,j']} \quad \text{für } 0 \leq i \leq p(n) und - p(n) \leq j < j' \leq p(n) + 1$$

Klauselgruppe 3:

Zum Zeitpunkt *i* enthält jede Bandstelle genau ein Symbol

Konstruktion:

- Zum Zeitpunkt *i* enthält jede Bandstelle mindestens ein Symbol

$$S[i,j,0] \vee S[i,j,1] \vee \ldots \vee S[i,j,\ell] \quad \text{für} \begin{cases} 0 \leq i \leq p(n) \\ -p(n) \leq j \leq p(n)+1 \end{cases}$$

- Zum Zeitpunkt *i* enthält jede Bandstelle höchstens ein Symbol

$$\overline{S[i,j,k]} \vee \overline{S[i,j,k']} \quad \text{für} \begin{cases} 0 \leq i \leq p(n) \\ -p(n) \leq j \leq p(n)+1 \\ 0 \leq k < k' \leq \ell \end{cases}$$

Klauselgruppe 4:

Festlegung der Anfangskonfiguration zum Zeitpunkt 0

Konstruktion:

- $\mathcal{M}$ ist im Zustand q_0

$$Q[0,0]$$

- der Lese-/Schreibkopf steht an Position 1 des Bandes

$$H[0,1]$$

- in den Zellen 1 bis n steht das Wort $x = s_{k_1} \dots s_{k_n}$

$$\begin{cases} S[0,0,0], S[0,1,k_1], \dots, S[0,n,k_n] & \text{, für Eingabe } x = s_{k_1} \dots s_{k_n} \\ S[0,n+1,0], \dots, S[0,p(n)+1,0] & \text{, alle anderen Positionen} \end{cases}$$

Klauselgruppe 5:

Bis zum Zeitpunkt $p(n)$ hat $\mathcal{M}$ den Zustand q_J erreicht.

Konstruktion:

$$Q[p(n), 1]$$

Klauselgruppe 6:

Zu jedem Zeitpunkt i folgt die Konfiguration von $\mathcal{M}$ zum Zeitpunkt $i+1$ aus einer einzigen Anwendung von δ aus der Konfiguration von $\mathcal{M}$ zum Zeitpunkt i.

Wir unterteilen Klauselgruppe G_6 in zwei Teilgruppen $G_{6,1}$, $G_{6,2}$.

- $G_{6,1}$: Falls $\mathcal{M}$ zum Zeitpunkt i an der Position j das Symbol s_k hat und der Lese-/Schreibkopf nicht an der Position j steht, dann hat $\mathcal{M}$ auch zum Zeitpunkt $i+1$ an Position j das Symbol s_k für $0 \leq i < p(n)$.
- $G_{6,2}$: Der Wechsel von einer Konfiguration zur nächsten entspricht tatsächlich δ.

Klauselgruppe 6,1:

Falls $\mathcal{M}$ zum Zeitpunkt i an der Position j das Symbol s_k hat und der Lese-/Schreibkopf nicht an der Position j steht, dann hat $\mathcal{M}$ auch zum Zeitpunkt $i+1$ an Position j das Symbol s_k für $0 \leq i < p(n)$.

Konstruktion:

$$\left(\left(S[i,j,k] \wedge \overline{H[i,j]}\right) \Longrightarrow S[i+1,j,k]\right)$$

Dies ergibt die Klausel

$$\left(\overline{S[i,j,k]} \vee H[i,j] \vee S[i+1,j,k]\right)$$

Klauselgruppe 6,2:

Der Wechsel von einer Konfiguration zur nächsten entspricht tatsächlich δ.

- Sei $\delta(q_k, s_m) = (q_\kappa, s_\mu, d)$.
- Œ sei q_k aus $Q \backslash \{q_J, q_N\}$ sonst gilt $q_\kappa = q_k$, $s_\mu = s_m$ und $d = 0$.

$$
\begin{aligned}
& (H[i,j] \wedge Q[i,k] \wedge S[i,j,m]) \Rightarrow H[i+1, j+d] \\
\text{und } & (H[i,j] \wedge Q[i,k] \wedge S[i,j,m]) \Rightarrow Q[i+1, \kappa] \\
\text{und } & (H[i,j] \wedge Q[i,k] \wedge S[i,j,m]) \Rightarrow S[i+1, j, \mu]
\end{aligned}
$$

Dies ergibt folgende Klauseln

$$
\begin{aligned}
& \overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,m]} \vee H[i+1, j+d] \\
& \overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,m]} \vee Q[i+1, \kappa] \\
& \overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,m]} \vee S[i+1, j, \mu]
\end{aligned}
$$

$$\text{für } 0 \leq i < p(n),\ -p(n) \leq j \leq p(n) + 1,\ 0 \leq k \leq r,\ 0 \leq m \leq \ell$$

Konstruktion der Klauseln - Zwischenergebnis

- Durch f_L wird nun ein Input $(\mathcal{M}, x)$ auf die Klauselmenge

$$G := G_1 \wedge G_2 \wedge \ldots \wedge G_6$$

  abgebildet.
- Wenn $x \in L$, dann ist G erfüllbar.
- Eine erfüllende Wahrheitsbelegung der Variablen aus G induziert eine akzeptierende Berechnung von $\mathcal{M}$ für die Eingabe $x \in$ L.

Polynomialität der Transformation

Karlsruher Institut für Technologie

Wir schätzen die Anzahl der Literale in den Klauselgruppen ab.

Polynomialität - Klauselgruppe 1:

Zum Zeitpunkt i ist $\mathcal{M}$ in genau einem Zustand.

Konstruktion:

- Zu jedem Zeitpunkt i ist $\mathcal{M}$ in mindestens einem Zustand

$$Q[i,0] \vee \ldots \vee Q[i,r] \quad \text{für } 0 \leq i \leq p(n)$$

- Zu jedem Zeitpunkt i ist $\mathcal{M}$ in nicht mehr als einem Zustand

$$\overline{Q[i,j]} \vee \overline{Q[i,j']} \quad \text{für } 0 \leq i \leq p(n),\ 0 \leq j < j' \leq r$$

Abschätzung:

$$(p(n)+1)(r+1) + (p(n)+1)\frac{1}{2}(r(r+1))$$

Polynomialität - Klauselgruppe 2:

Zum Zeitpunkt *i* hat der Lese-/Schreibkopf genau eine Position

Konstruktion:

- Zum Zeitpunkt *i* hat der Lese-/Schreibkopf mindestens eine Position

$$H[i, -p(n)] \vee \ldots \vee H[i, p(n)+1] \quad \text{für } 0 \leq i \leq p(n)$$

- Zum Zeitpunkt *i* hat der Lese-/Schreibkopf höchstens eine Position

$$\overline{H[i,j]} \vee \overline{H[i,j']} \quad \text{für } 0 \leq i \leq p(n) und - p(n) \leq j < j' \leq p(n)+1$$

Abschätzung:

$$(p(n)+1)(2p(n)+1) + (p(n)+1)\frac{1}{2}(2p(n) \cdot (2p(n)+1))$$

Polynomialität - Klauselgruppe 3:

Zum Zeitpunkt *i* enthält jede Bandstelle genau ein Symbol

Konstruktion:

- Zum Zeitpunkt *i* enthält jede Bandstelle mindestens ein Symbol

$$S[i,j,0] \vee S[i,j,1] \vee \ldots \vee S[i,j,\ell] \quad \text{für} \begin{cases} 0 \leq i \leq p(n) \\ -p(n) \leq j \leq p(n)+1 \end{cases}$$

- Zum Zeitpunkt *i* enthält jede Bandstelle höchstens ein Symbol

$$\overline{S[i,j,k]} \vee \overline{S[i,j,k']} \quad \text{für} \begin{cases} 0 \leq i \leq p(n) \\ -p(n) \leq j \leq p(n)+1 \\ 0 \leq k < k' \leq \ell \end{cases}$$

Abschätzung:

$$(p(n)+1)(2p(n)+1)(\ell+1) + (p(n)+1)(2p(n)+1)\frac{1}{2}(\ell(\ell+1))$$

Polynomialität - Klauselgruppe 4:

Festlegung der Anfangskonfiguration zum Zeitpunkt 0

- $\mathcal{M}$ ist im Zustand q_0

$$Q[0,0]$$

- der Lese-/Schreibkopf steht an Position 1 des Bandes

$$H[0,1]$$

- in den Zellen 1 bis n steht das Wort $x = s_{k_1} \dots s_{k_n}$

$$\begin{cases} S[0,0,0], S[0,1,k_1], \dots, S[0,n,k_n] & \text{, für Eingabe } x = s_{k_1} \dots s_{k_n} \\ S[0,n+1,0], \dots, S[0,p(n)+1,0] & \text{, alle anderen Positionen} \end{cases}$$

Abschätzung:

$$2 + (n+1) + (p(n) + 2 - (n+1)) = p(n) + 4$$

Polynomialität - Klauselgruppe 5:

Bis zum Zeitpunkt $p(n)$ hat $\mathcal{M}$ den Zustand q_J erreicht.

Konstruktion:

$$Q[p(n), 1]$$

Abschätzung:

$$1$$

Polynomialität - Klauselgruppe 6,1:

Falls $\mathcal{M}$ zum Zeitpunkt i an der Position j das Symbol s_k hat und der Lese-/Schreibkopf nicht an der Position j steht, dann hat $\mathcal{M}$ auch zum Zeitpunkt $i+1$ an Position j das Symbol s_k für $0 \leq i < p(n)$.

Konstruktion:

$$\left(\left(S[i,j,k] \wedge \overline{H[i,j]}\right) \Longrightarrow S[i+1,j,k]\right)$$

Dies ergibt die Klausel

$$\left(\overline{S[i,j,k]} \vee H[i,j] \vee S[i+1,j,k]\right)$$

Abschätzung:

$$p(n)(\ell+1)(2p(n)+2) \cdot 3$$

Polynomialität - Klauselgruppe 6,2:

Der Wechsel von einer Konfiguration zur nächsten entspricht tatsächlich δ.

- Sei $\delta(q_k, s_m) = (q_\kappa, s_\mu, d)$.
- Œ sei q_k aus $Q \backslash \{q_J, q_N\}$ sonst gilt $q_\kappa = q_k$, $s_\mu = s_m$ und $d = 0$.

$$\overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,m]} \vee H[i+1, j+d]$$
$$\overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,m]} \vee Q[i+1, \kappa]$$
$$\overline{H[i,j]} \vee \overline{Q[i,k]} \vee \overline{S[i,j,m]} \vee S[i+1, j, \mu]$$

$$\text{für } 0 \leq i < p(n),\ -p(n) \leq j \leq p(n)+1,\ 0 \leq k \leq r,\ 0 \leq m \leq \ell$$

Abschätzung:

$$p(n)(p(n)+2)(r+1)(\ell+1) \cdot 3 \cdot 4$$

Polynomialität der Transformation

Wir schätzen die Anzahl der Literale in den Klauselgruppen ab.

- G_1: $(p(n)+1)(r+1)+(p(n)+1)\frac{1}{2}(r(r+1))$
- G_2: $(p(n)+1)(2p(n)+1)+(p(n)+1)\frac{1}{2}(2p(n)\cdot(2p(n)+1))$
- G_3:
  $(p(n)+1)(2p(n)+1)(\ell+1)+(p(n)+1)(2p(n)+1)\frac{1}{2}(\ell(\ell+1))$
- G_4: $2+(n+1)+(p(n)+2-(n+1))=p(n)+4$
- G_5: 1
- G_6: $\underbrace{p(n)(\ell+1)(2p(n)+2)\cdot 3}_{G_{6,1}}+\underbrace{p(n)(p(n)+2)(r+1)(\ell+1)\cdot 3\cdot 4}_{G_{6,2}}$

Polynomialität der Transformation

Wir schätzen die Anzahl der Literale in den Klauselgruppen ab.

- G_1: $(p(n)+1)(r+1)+(p(n)+1)\frac{1}{2}(r(r+1))$
- G_2: $(p(n)+1)(2p(n)+1)+(p(n)+1)\frac{1}{2}(2p(n)\cdot(2p(n)+1))$
- G_3:
  $(p(n)+1)(2p(n)+1)(\ell+1)+(p(n)+1)(2p(n)+1)\frac{1}{2}(\ell(\ell+1))$
- G_4: $2+(n+1)+(p(n)+2-(n+1))=p(n)+4$
- G_5: 1
- G_6: $\underbrace{p(n)(\ell+1)(2p(n)+2)\cdot 3}_{G_{6,1}}+\underbrace{p(n)(p(n)+2)(r+1)(\ell+1)\cdot 3\cdot 4}_{G_{6,2}}$

- r und ℓ sind Konstanten, die durch $\mathcal{M}$ (und damit durch L) induziert werden
- $p(n)$ ist ein Polynom in n

Polynomialität der Transformation

Wir schätzen die Anzahl der Literale in den Klauselgruppen ab.

- G_1: $(p(n)+1)(r+1)+(p(n)+1)\frac{1}{2}(r(r+1))$
- G_2: $(p(n)+1)(2p(n)+1)+(p(n)+1)\frac{1}{2}(2p(n)\cdot(2p(n)+1))$
- G_3:
  $(p(n)+1)(2p(n)+1)(\ell+1)+(p(n)+1)(2p(n)+1)\frac{1}{2}(\ell(\ell+1))$
- G_4: $2+(n+1)+(p(n)+2-(n+1))=p(n)+4$
- G_5: 1
- G_6: $\underbrace{p(n)(\ell+1)(2p(n)+2)\cdot 3}_{G_{6,1}}+\underbrace{p(n)(p(n)+2)(r+1)(\ell+1)\cdot 3\cdot 4}_{G_{6,2}}$

- Also sind alle Größen polynomial in n.
- Die angegebene Funktion f_L ist damit eine polynomiale Transformation von L nach L_{SAT}.