Prädikatenlogik zweiter Stufe und der Satz von Fagin

Alexander Kövesdi

Einleitung

Die gewöhnliche Beschreibung von NP ist als Klasse aller Entscheidungsprobleme, die von einer nichtdeterministischen Turingmaschine in Polynomialzeit gelöst werden können. In dieser Seminararbeit werden wir die existentielle Prädikatenlogik 2. Stufe (SO∃) einführen und mit dem Satz von Fagin zu einer neuen Beschreibung von der Klasse NP kommen. Als nächstes werden wir einige Entscheidungsprobleme betrachten, die bezüglich Reduktionen 1. Stufe NP-vollständig sind und zuletzt folgt aus einem Korollar, dass Boolesche Abfragen 2. Stufe genau diejenigen sind, die in der Polynomialzeithierarchie berechenbar sind.

1 Prädikatenlogik zweiter Stufe

Prädikatenlogik zweiter Stufe besteht aus der Prädikatenlogik erster Stufe und zusätzlich aus quantifizierbaren Relationsvariablen. Die Formel $(\forall A^r)\varphi$ bedeutet, dass φ für alle r-stelligen Relationen A gilt.

Beispiel 1.1. *Die Aussage „es gibt einen Gott und der hat alle positiven Eigenschaften“ kann man mit Prädikatenlogik zweiter Stufe wie folgt beschreiben:*

$$\exists x(G(x) \wedge \forall E(P(E) \rightarrow E(x)))$$

Jede Formel zweiter Stufe kann in eine äquivalente Formel umgewandelt werden, in der alle Quantoren am Anfang stehen. Bestehen diese nur aus Existenzquantoren, so hat man eine existentielle Formel zweiter Stufe. Wir werden die Menge der existentiellen Booleschen Abfragen zweiter Stufe mit $(SO\exists)$ bezeichnen.
Es folgen einige Beispiele zu $(SO\exists)$ Abfragen:

Beispiel 1.2. *Man beachte, dass R, Y und B einstellige Relationsvariablen sind:*

$$\Phi_{3-color} \equiv (\exists R^1)(\exists Y^1)(\exists B^1)(\forall x)\Big[\big(R(x) \vee Y(x) \vee B(x)\big) \wedge (\forall y)\Big(E(x,y) \rightarrow \neg\big(R(x) \wedge R(y)\big) \wedge \neg\big(Y(x) \wedge Y(y)\big) \wedge \neg\big(B(x) \wedge B(y)\big)\Big)\Big]$$

Ein Graph G erfüllt $\Phi_{3-color}$ genau dann, wenn G 3-färbbar ist. Das bedeutet, dass man mit 3 Farben jedem Knoten eine Farbe zuordnen kann, sodass keine zwei benachbarten Knoten die selbe Farbe haben.

Beispiel 1.3.

$$\Phi_{SAT} \equiv (\exists S)(\forall x)(\exists y)((P(x,y) \wedge S(y)) \vee (N(x,y) \wedge \neg S(y)))$$

SAT ist die Menge aller Boolesche Formeln in konjunktiver Normalform, die erfüllbar sind. Φ_{SAT} sagt aus, dass eine Menge S von Variablen existiert, die als Interpretation aufgefasst die Eingabeformel erfüllt. Die Relation $P(c,v)$ bedeutet, dass die Variable v in Klausel c positiv auftritt und $N(c,v)$ beschreibt negatives Auftreten.

Die obigen Beispiele sind NP-vollständige Probleme. Wir werden sehen, dass diese bezüglich Reduktionen erster Stufe vollständig bleiben.

2 Satz von Fagin

Der Satz von Fagin beschreibt die Komplexitätsklasse NP, ohne ein Berechnungsmodell wie Turingmaschinen zu benötigen. NP wird durch die Menge der existentiellen Boolesche Abfragen zweiter Stufe beschrieben.
Wir fangen zunächst mit der einfachen Richtung des Beweises an:

Proposition 2.1. $SO\exists \subseteq NP$

Beweis. Gegeben ist ein existentieller Satz zweiter Stufe
$\Phi \equiv (\exists R_1^{r_1}) \ldots (\exists R_k^{r_k})\psi$. Sei τ das Vokabular von Φ. Unsere Aufgabe ist es, eine nichtdeterministische Turingmaschine N zu konstruieren, so dass für alle $\mathcal{A} \in STRUCT[\tau]$,

$$(\mathcal{A} \models \Phi) \iff (N(bin(\mathcal{A}) \downarrow) \tag{1}$$

Sei $\mathcal{A}$ ein Input zu N und $n = ||\mathcal{A}||$. N schreibt nichtdeterministisch einen Binärstring der Länge n^{r_1} auf, um R_1 zu repräsentieren. Es folgt ähnlich für R_2 bis R_k. Mit nichtdeterministisch einen Binärstring schreiben, meinen wir, dass bei jedem Schritt, N auf nichtdeterministischer Art entscheidet ob

sie eine 0 oder eine 1 schreiben soll. Nach einer polynomiellen Anzahl von Schritten erhalten wir die erweiterte Struktur $\mathcal{A}' = (\mathcal{A}, R_1, R_2, \ldots, R_k)$. N akzeptiert genau dann, wenn $\mathcal{A}' \models \psi$. Wegen Satz 3.1 aus [1] gilt $FO \subseteq L \subseteq NP$. Wenn wir in L überprüfen können, ob $\mathcal{A}' \models \psi$, dann auch in NP. Zu beachten ist, dass $\mathcal{A}$ genau dann von N akzeptiert wird, wenn es eine Wahl von Relationen R_1 bis R_k gibt, so dass $(\mathcal{A}, R_1, R_2, \ldots, R_k) \models \psi$. Damit gilt (1). □

Satz 2.2 (Satz von Fagin)**.** *$NP = SO\exists$. Weiters bleibt diese Gleichheit wahr, wenn die erste Stufe der SO-Formel auf Allaussagen beschränkt wird.*

Beweis. Sei N eine nichtdeterministische Turingmaschine, welche die Zeit $n^k - 1$ für Eingaben $bin(\mathcal{A})$ mit $n = ||\mathcal{A}||$ verwendet. Wir schreiben einen Satz zweiter Stufe,

$$\Phi = (\exists C_1^{2k} \ldots C_g^{2k} \Delta^k)\varphi$$

der aussagt: „Es existiert eine akzeptierte Berechnung $\overline{C}$, Δ von N.“ Genauer: Der Satz 1. Stufe φ wird die Eigenschaft haben, dass $(\mathcal{A}, \overline{C}, \Delta) \models \varphi$ genau dann, wenn $\overline{C}, \Delta$ als Berechnung von N auf Eingabe $\mathcal{A}$ akzeptiert wird.
Wir beschreiben nun, wie man die Berechnung von N codiert. $\overline{C}$ besteht aus einer Matrix $\overline{C}(\bar{s}, \bar{t})$ von n^{2k} Bandfeldern mit Speicher $\bar{s}$ und Zeit $\bar{t}$, die zwischen 0 und $n^k - 1$ variieren. Wir verwenden k-Tupel von Variablen $\bar{t} = t_1, \ldots, t_k$ und $\bar{s} = s_1, \ldots, s_k$ mit Bereichen von 0 bis $n-1$. Für jedes Paar $\bar{s}$, $\bar{t}$, codiert $\overline{C}(\bar{s}, \bar{t})$ das Bandsymbol σ, welches in Feld $\bar{s}$ um die Zeit $\bar{t}$ auftaucht, falls der Kopf von n nicht auf diesem Feld ist. Ist der Kopf vorhanden, dann codiert $\overline{C}(\bar{s}, \bar{t})$ das Paar $\langle q, \sigma \rangle$, welches aus dem Zustand q von N um die Zeit $\bar{t}$ und dem Bandsymbol σ besteht. Sei $\Gamma = \{\gamma_0, \ldots, \gamma_g\} = (Q \times \Sigma) \cup \Sigma$ eine Liste von allen möglichen Inhalten eines Rechenfeldes. C_i sei eine $2k$-stellige Relationsvariable mit $0 \leq i \leq g$. Intuitiv bedeutet $C_i(\bar{s}, \bar{t})$, dass das Rechenfeld $\bar{s}$ zur Zeit $\bar{t}$ das Symbol γ_i enthält.
Die nichtdeterministische Turingmaschine wird bei jedem Schritt eine von höchstens zwei möglichen Entscheidungen treffen.[1] Wir codieren diese Entscheidungen in die k-stellige Relation Δ. $\Delta(\bar{t})$ ist wahr, wenn der Schritt $\bar{t} + 1$ der Berechnung die Entscheidung „1“ trifft. Andernfalls wählt sie „0“. Eine Berechnung von N findet man in Abbildung 1.
Wir werden nun den Satz erster Stufe $\varphi(\overline{C}, \Delta)$ schreiben, der aus vier Teilen besteht:

$$\varphi \equiv \alpha \wedge \beta \wedge \eta \wedge \zeta$$

[1] Eine nichtdeterministische Turingmaschine kann pro Schritt höchstens eine von einer beschränkten Anzahl an Entscheidungen treffen. Wir reduzieren dies zu einer binären Anzahl an Entscheidungen pro Schritt. Die Maschine wird dabei um einen kleinen konstanten Faktor verlangsamt, vereinfacht jedoch die Analyse.

α besagt, dass Zeile 0 der Berechnung die Eingabe $bin(\mathcal{A})$ korrekt codiert.
β sagt, dass der Fall wo $C_i(\bar{s}, \bar{t})$ und $C_j(\bar{s}, \bar{t})$ für $i \neq j$ beide richtig sind, nie gelten kann.
η besagt, dass für alle $\bar{t}$, die Zeile $\bar{t} + 1$ von $\overline{C}$, aus der Zeile $\bar{t}$ mittels $\Delta(\bar{t})$ folgt.
ζ sagt, dass die letzte Zeile der Berechnung den akzeptierenden Endzustand enthält.
Wir können ζ explizit aufschreiben. Man kann annehmen, dass wenn N akzeptiert, das Band gelöscht wird, sich komplett nach links bewegt und einen akzeptierenden Endzustand q_f erreicht. Sei γ_{17} aus Γ, das dem Paar $\langle q_f, 1\rangle$ entspricht. Dann ist $\zeta = C_{17}(\overline{0}, \overline{max})$.
Satz α muss behaupten, dass unsere Eingabe Länge $I_\tau(n)$ für ein n hat und dass $\mathcal{A}$ richtig codiert wurde als $bin(\mathcal{A})$ (vgl. 2.3 in [1]). Zum Beispiel nehme man an, dass τ das einstellige Relationssymbol R_1 inkludiert. Weiters gehe man davon aus, dass γ_0, γ_1 jeweils ‚0' und ‚1' seien. Dann enthält α folgende Klauseln, die bedeuten, dass die Zelle $0 \ldots 0 s_k$ die 1 enthält, falls $R_1(s_k)$ gilt, und 0 falls nicht.

$$\begin{aligned} &\cdots \wedge \Big(\bar{t} = 0 = s_1 = \cdots = s_{k-1} \wedge s_k \neq 0 \wedge R_1(s_k) \rightarrow C_1(\bar{s}, \bar{t})\Big) \\ &\quad \wedge \Big(\bar{t} = 0 = s_1 = \cdots = s_{k-1} \wedge s_k \neq 0 \wedge \neg R_1(s_k) \rightarrow C_0(\bar{s}, \bar{t})\Big) \wedge \cdots \end{aligned}$$

η soll besagen, dass der Inhalt des Bandfeldes $(\bar{s}, \bar{t}+1)$, mittels $\Delta(\bar{t})$, aus den Inhalten der Felder $(\bar{s}-1, \bar{t})$, $(\bar{s}, \bar{t})$ und $(\bar{s}+1, \bar{t})$ folgt. Definiere $\langle a_{-1}, a_0, a_1, \delta\rangle \xrightarrow{N} b$ als den Übergang der Tripel a_{-1}, a_0, a_1 zu dem Feld b mittels δ von N.

$$\begin{aligned} \eta_1 \equiv (\forall \bar{t}.\bar{t} \neq \overline{max})(\forall \bar{s}.\bar{0} < \bar{s} < \overline{max}) \bigwedge_{\langle a_{-1}, a_0, a_1, \delta\rangle \xrightarrow{N} b} \Big(\neg^\delta \Delta(\bar{t}) \vee \\ \neg C_{a_{-1}}(\bar{s}-1, \bar{t}) \vee \neg C_{a_0}(\bar{s}, \bar{t}) \vee \neg C_{a_1}(\bar{s}+1, \bar{t}) \vee C_b(\bar{s}, \bar{t}+1)\Big) \end{aligned}$$

$\neg^\delta$ ist $\neg$, falls $\delta = 1$ und das leere Symbol, falls $\delta = 0$.
Sei nun $\eta \equiv \eta_0 \wedge \eta_1 \wedge \eta_2$, wobei η_0 und η_2 dieselbe Information mit $\bar{s} = \overline{0}$ bzw. $\bar{s} = \overline{max}$ kodieren.

□

In 2.2 ist der first-order Teil der Aussage Φ eine Allaussage in konjunktiver Normalform. Ist N eine deterministische Turingmaschine, so benötigen wir Δ nicht, also ist der first-order Teil von Φ eine Horn-Formel. Wir erhalten dadurch folgendes Korollar.

Korollar 2.3. $P \subseteq SO\exists\text{-}Horn$

Durch den Beweis von 2.2 sehen wir, dass die nichtdeterministische Zeit n^k in $SO\exists$ mit Stelligkeit $2k$ enthalten ist. Lynch verbessert dies auf Stelligkeit k.

	Space 0	1	p	$n-1$	n		n^k-1	Δ
Time 0	$\langle q_0, w_0\rangle$	w_1	$\cdots$	w_{n-1}	$\sqcup$	$\cdots$	$\sqcup$	δ_0
1	w_0	$\langle q_1, w_1\rangle$	$\cdots$	w_{n-1}	$\sqcup$	$\cdots$	$\sqcup$	δ_1
	$\vdots$	$\vdots$	$\vdots$			$\vdots$		$\vdots$
t			$a_{-1}a_0a_1$					δ_t
$t+1$			b					δ_{t+1}
	$\vdots$	$\vdots$	$\vdots$			$\vdots$		$\vdots$
n^k-1	$\langle q_f, 1\rangle$	$\cdots$	$\cdots$			$\cdots$		

Abbildung 1: Eine NP-Berechnung zur Eingabe $w_0w_1\cdots w_{n-1}$. $\sqcup$ bezeichnet ein leeres Feld.

Satz 2.4 (Satz von Lynch)**.** *Für $k \geq 1$, $NTIME[n^k] \subseteq SO\exists(Stelligkeit\ k)$*

Beweis. Wir modifizieren den Beweis von 1 so, dass man nur eine beschränkte Anzahl an bits pro Schritt errät anstatt das ganze Band. Die folgenden Relationen müssen erraten werden.

1. $Q_i(\bar{t})$ ... der Zustand um Bewegung $\bar{t}$ ist q_i,
2. $S_i(\bar{t})$ ... das Symbol, dass um Bewegung $\bar{t}$ geschrieben wird ist σ_i,
3. $D(\bar{t})$ ... der Lesekopf bewegt sich eins nach rechts nach Bewegung $\bar{t}$; sonst bewegt es sich eins nach links.

Wir müssen eine Aussage 1. Stufe schreiben, welche besagt, dass $\bar{Q}, \bar{S}, D$ eine korrekte Berechnung kodiert, die von N akzeptiert wird. Die einzige Schwierigkeit dabei ist, dass wir nach jeder Bewegung $\bar{t}$ das Symbol $\rho_{\bar{t}}$, welches von N gelesen wird, ermitteln müssen. $\rho_{\bar{t}}$ entspricht σ_i, wenn $S_i(\bar{t'})$ gilt. Dabei ist $\bar{t'}$ das letzte Mal vor $\bar{t}$, wo der Lesekopf in seiner aktuellen Position war (oder sie ist das korrespondierende Eingabesymbol, falls es das erste Mal ist, dass der Lesekopf an dieser Position am Band ist).
Um $\rho_{\bar{t}}$ auszudrücken, müssen wir die Funktion $\bar{s} = p(\bar{t})$ ausdrücken, welches bedeutet, dass um Zeit $\bar{t}$ der Lesekopf sich in Position $\bar{s}$ befindet. Da wir auf Relationen mit Stelligkeit k beschränkt sind, können wir die $k\ log\ n$ bits pro Zeit die benötigt werden um die Funktion p zu spezifizieren, nicht erraten. Die Lösung zu diesem Problem ist, die Position vom Lesekopf alle $log\ n$ Schritte einmal existenziell zu quantifizieren. Wir können dies erreichen, indem wir k Bits pro Schritt in den Relationen $P_i(\bar{t}), i = 1, 2, \ldots, k$ quantifizieren. Wenn wir $log\ n$ davon zusammenfügen, von der Zeit $r\ log\ n$ bis $(r+1)log\ n - 1$, erhalten wir insgesamt $k\ log\ n$ Bits, welche die Position vom Lesekopf um Zeit $r\ log\ n$ kodieren.
Die Idee ist ähnlich wie der Beweis vom Bit Sum Lemma 1.18 aus [1]. BIT erlaubt uns jede first-order Variable zum Speichern von $log\ n$ Bits zu verwenden. Weiters ist $BSUM(x, y)$ von erster Stufe (Lemma 1.18 aus [1]).

$BSUM(x, y)$ bedeutet, dass die Anzahl an 1-ern in der binary expansion von x gleich y ist Daher können wir behaupten, dass die Relationen $\bar{P}$ mit den Lesekopf Bewegungen die von D gegeben werden, konsistent sind und wir somit die Lesekopf Position bei $log\ n$ Schritt-Intervallen korrekt kodieren. Mit einer erneuten Anwendung von $BSUM$ können wir letztendlich die Lesekopf Position um jede Zeit $\bar{t}$ ermitteln. □

Die umgekehrte Richtung vom Satz von Lynch bleibt ein offenes Problem.

Offenes Problem 2.5. *Ist $SO\exists(Stelligkeit\ k) = NTIME[n^k]$?*

Die Subtilität bei 2.5 ist, dass der first-order Teil einer $SO\exists(Stelligkeit\ k)$ Aussage mehr als k Allquantoren besitzen kann. Daher könnte ein wichtiger Schritt bei der Beantwortung von Problem 2.5 die Antwort zur folgenden Frage sein:

Offenes Problem 2.6. *Existiert ein festes k, sodass $FO \subseteq DTIME[n^k]$? Existiert ein festes k, sodass $FO \subseteq NTIME[n^k]$?*

Grandjean stellt eine enge Beziehung zwischen nichtdeterministischer Zeit n^k und der Klasse $(SO\exists, fun, k\forall)$ dar. $(SO\exists, fun, k\forall)$ ist die Klasse von Eigenschaften, die durch existentielle Ausdrücke 2. Stufe ausdrückbar sind, welche Funktionsvariablen unkludieren und nur k Allquantoren 1. Stufe enthalten.

Fakt 2.7. *Für $k \geq 2$, $NTIME[n^k] \subseteq (SO\exists, fun, k\forall) = (SO\exists, fun, k\forall, Stelligkeit\ k) \subseteq NTIME[n^k(log\ n)^2]$.*

Unter Berücksichtigung von nichtdeterministischen Random Access Maschinen ($NRAM$) statt Turingmaschinen, erhält Grandjean eine exakte Schranke.

Fakt 2.8. *Für $k \geq 1$,*

$$NRAM\text{-}TIME[n^k] = (SO\exists, fun, k\forall, Stelligkeit\ k)$$

3 NP-vollständige Probleme

Satz 3.1. *SAT ist NP-vollständig bezüglich Reduktionen 1. Stufe.*

Beweis. Folgt mit dem Satz von Fagin. Gegeben sei eine beliebige Boolesche Abfrage $B \in NP$. Wir wissen $B = MOD[\Phi]$, wobei $\Phi = (\exists S_1^{a_1} \cdots S_g^{a_g} \Delta^k)(\forall x_1 \cdots x_t)\psi(\bar{x})$ mit ψ quantorenfrei. Wir können annehmen, dass $\psi(\bar{x}) = \bigwedge_{j=1}^{r} C_j(\bar{x})$ in konjunktiver Normalform ist.

Für jede beliebige Eingabestruktur $\mathcal{A}$ mit $n = ||\mathcal{A}||$, definieren wir die Boolesche Formel $\gamma(\mathcal{A})$ wie folgt: $\gamma(\mathcal{A})$ hat Boolesche Variablen: $S_i(e_1, \ldots, e_{a_i})$ und $D(e_1, \ldots, e_k)$, $i = 1, \ldots, g$, $e_1, \ldots, e_{a_i} \in |\mathcal{A}|$. Die Klauseln von $\gamma(\mathcal{A})$ sind $C_j(\bar{e})$, $j = 1, \ldots, r$. In jedem $C_j(\bar{e})$ können Numerische- oder Eingabeprädikate $\gamma(\bar{e})$ auftauchen. Diese sollen durch **wahr** oder **falsch** ersetzt werden, je nachdem ob sie in $\mathcal{A}$ wahr oder falsch sind.
Es wird von der Konstruktion klar, dass

$$\mathcal{A} \in B \iff \mathcal{A} \models \Phi \iff \gamma(\mathcal{A}) \in SAT.$$

Ferner gilt, dass die Abbildung von $\mathcal{A}$ zu $\gamma(\mathcal{A})$ eine $t+1$-stellige Boolesche Abfrage 1. Stufe ist. □

Da wir wissen, dass SAT mittels Reduktionen 1. Stufe NP-vollständig ist, können wir SAT auf andere SO∃ Boolsche Abfragen reduzieren. Dies ist möglich genau dann, wenn die anderen Probleme auch mittels Reduktionen 1. Stufe NP-vollständig sind.

Proposition 3.2. *Sei* 3-SAT *die Teilmenge von* SAT*, in der jede Klausel höchstens aus drei Literalen besteht. Dann ist* 3-SAT *bezüglich Reduktionen 1. Stufe* NP*-vollständig.*

Beweis. Wir zeigen $SAT \leq_{fo}$ 3-SAT. Es folgt ein Beispiel zur Idee hinter der Reduktion. Sei $C = (\ell_1 \vee \ell_2 \vee \cdots \vee \ell_7)$ eine Klausel mit mehr als drei Literalen. Beachte, dass $C \in SAT$ genau dann, wenn $C' \in$ 3-SAT, wobei C' die folgende Klausel ist, in der neue Variablen $d_1, \ldots, d_4$ eingeführt werden.

$$\begin{aligned} C' \equiv (\ell_1 \vee \ell_2 \vee d_1) \ \wedge \ (\overline{d_1} \vee \ell_3 \vee d_2) \ \wedge \ (\overline{d_2} \vee \ell_4 \vee d_3) \wedge \\ (\overline{d_3} \vee \ell_5 \vee d_4) \ \wedge \ (\overline{d_4} \vee \ell_6 \vee \ell_7) \end{aligned}$$

Die Reduktion 1. Stufe von SAT auf 3-SAT geht wie folgt vor. Sei $\mathcal{A} \in STRUC[\langle P^2, N^2 \rangle]$ eine Instanz von SAT mit $n = ||\mathcal{A}||$. Jede Klausel c von $\mathcal{A}$ wird wie folgt von $2n$ Klauseln ersetzt:

$$\begin{aligned} c' \equiv ([x_1]^c \vee d_1) \ \wedge \ (\overline{d_1} \vee [x_2]^c \vee d_2) \ \wedge \ (\overline{d_2} \vee [x_3]^c \vee d_3) \wedge \ \cdots \ \wedge \\ (\overline{d_n} \vee \overline{[x_1]^c} \vee d_{n+1})(\overline{d_{n+1}} \vee \overline{[x_2]^c} \vee d_{n+2}) \ \wedge \ \cdots \ \wedge \ (\overline{d_{2n-1}} \vee [\overline{x_n}^c]) \end{aligned}$$

Hier bedeutet $[\ell]^c$ das Literal ℓ, falls es in c auftritt und sonst **false**. Nun ist leicht zu erkennen, dass c' erfüllbar ist genau dann, wenn c erfüllbar ist und dass c' in einer 1. Stufe definierbar ist von c. □

Proposition 3.3. *3-COLOR ist NP-vollständig bezüglich Reduktionen 1. Stufe*

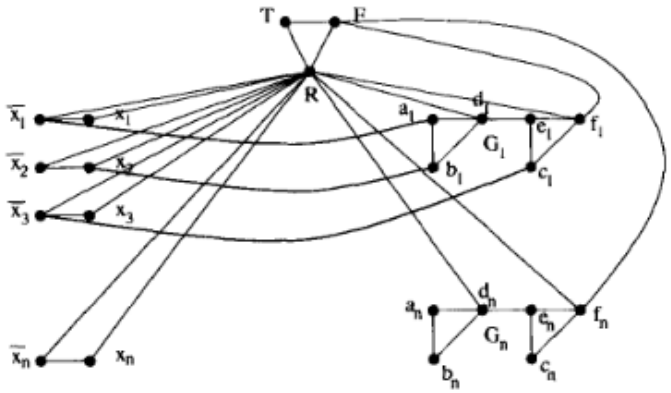

Abbildung 2: 3-SAT $\leq_{fo}$ 3-COLOR; G_1 codiert Klausel $C_1 = (\overline{x_1} \vee x_2 \vee \overline{x_3})$

Beweis. Wir werden zeigen, dass 3-SAT $\leq_{fo}$ 3-COLOR. Gegeben ist also eine Instanz $\mathcal{A}$ von 3-SAT und wir müssen ein Graph $f(\mathcal{A})$ erstellen, welches 3-färbbar ist $\iff$ $\mathcal{A} \in$ 3-SAT. Sei $n = ||\mathcal{A}||$, also ist $\mathcal{A}$ eine Boolesche Formel mit höchstens n Variablen und n Klauseln.
Die Konstruktion von $f(\mathcal{A})$ wird in Abbildung 2 gezeigt. Betrachte das Dreieck mit den Eckpunkten T, F, R. Jede 3-Färbung vom Graph muss diese Eckpunkte in verschiedenen Farben färben. Wir können ohne Bechränkung der Allgemeinheit annehmen, dass die Farben die für die Färbung von T,F,R benutzt werden jeweils wahr, falsch und rot sind.
Der Graph $f(\mathcal{A})$ besitzt eine Leiter, in der jede Sprosse die Variable x_i und ihre Negation $\bar{x_i}$ ist. Jedes dieser Leitersprossen ist mit R verbunden, also wird jede gültige Färbung von den x_i, $\bar{x_i}$ das eine als wahr und das andere als falsch färben.
Für jede Klausel $C_i = \ell_1 \vee \ell_2 \vee \ell_3$ enthält $f(\mathcal{A})$ G_i, welches sechs Eckpunkte besitzt. G_i hat drei Eingänge a_i, b_i, c_i die jeweils mit den Literalen ℓ_1, ℓ_2, ℓ_3 verbunden sind, und einen Ausgang f_i. Siehe Abbildung 2, wo G_1 der Klausel $C_1 = \bar{x_1} \vee x_2 \vee \bar{x_3}$ entspricht.
Das Dreieck a_1, b_1, d_1 dient als „oder“, in dem Sinne dass d_1 wahr gefärbt werden kann genau dann, wenn mindestens eines der $\bar{x_1}$, x_2 als wahr gefärbt ist. In ähnlicher Weise kann Ausgang f_1 als wahr gefärbt werden genau dann, wenn zumindest eins von d_1 und $\bar{x_3}$ als wahr gefärbt ist. Da f_i sowohl mit F als auch mit R verbunden ist, kann f_i nur wahr als Farbe haben. Es folgt: Eine drei-Färbung der Literalen kann erweitert werden um G_i zu färben genau dann, wenn die entsprechende Wahrheitszuweisung C_i wahr macht. Somit gilt $f(\mathcal{A}) \in$ 3-COLOR $\iff$ $\mathcal{A} \in$ 3-SAT.
Die Details der Reduktion 1. Stufe sind leicht auszufüllen. $f(\mathcal{A})$ besteht aus einem Dreieck, einer Leiter mit n Sprossen und n Kopien des Geräts. Die einzige Abhängigkeit von Eingabe $\mathcal{A}$ - im Gegensatz zu seiner Größe - ist dass es eine Kante von Literal ℓ zu Eingabe j von Gerät G_i gibt, genau

dann, wenn ℓ das j-te Literal ist, welches in C_i vorkommt. $\square$

4 Die Polynomialzeithierarchie (PH)

Definition 4.1 (Polynomialzeithierarchie mittels Orakeln)**.** *Sei $\Sigma_0^p = P$ Stufe 0 der Polynomialzeithierarchie. Wir definieren induktiv:*

$$\Sigma_{i+1}^p = \left\{ L(M^A) \mid M \text{ ist NP Orakel-TM}, A \in \Sigma_i^p \right\}$$

Äquivalent dazu ist Σ_{i+1}^p die Menge der Boolesche Abfragen die in nichtdeterministischer polynomieller Zeit Turing-reduzierbar sind zu einer Menge in Σ_i^p,

$$\Sigma_{i+1}^p = \left\{ B \mid B \leq_{np}^t A, \text{für ein } A \in \Sigma_i^p \right\}$$

Definiere Π_i^p als co-Σ_i^p, $\Pi_i^p = \left\{ \bar{A} \mid \ A \in \Sigma_i^p \right\}$. Schließlich, $PH = \bigcup_{k=1}^{\infty} \Sigma_k^p$.

Die Beziehung zwischen Booleschen Abfragen 2. Stufe und den Stufen der Polynomialhierarchie wird durch folgenden Satz gegeben:

Satz 4.2. *Sei $S \subseteq STRUC[\tau]$ eine Boolesche Abfrage und sei $k \geq 1$. Dann sind folgende Aussagen äquivalent,*

1. *$S = MOD[\Phi]$, für ein $\Phi \in \Sigma_k^{SO}$. (Hier is Σ_k^{SO} die Menge alle Ausdrücke 2. Stufe mit Quantifizierer-Präfix 2. Stufe $(\exists \bar{R}_1)(\forall \bar{R}_2) \ldots (Q_k \bar{R}_k)$.)*
2. *$S = \left\{ x \mid (\exists y_1.|y_1| \leq |x|^c)(\forall y_2.|y_2| \leq |x|^c) \cdots (Q_k y_k.|y_k| \leq |x|^c) R(x, \bar{y}) \right\}$ wobei R ein deterministisches Polynomialzeitprädikat auf $k+1$ Tupeln von Binärstrings ist und c eine Konstante.*
3. *$S \in ATIME\text{-}ALT[n^{O[1]}, k]$.*
4. *$S \in \Sigma_k^p$.*

Korollar 4.3. *Eine Boolesche Abfrage ist genau dann in der Polynomialzeithierarchie, wenn sie mittels 2. Stufe ausdrückbar ist,*

$$PH = SO.$$

Korollar 4.4. *Die folgenden Aussagen sind äquivalent:*

1. *$P = NP$*
2. *Für endliche und geordnete Strukturen gilt $FO(LFP) = SO$*

Beweis. Ist $FO(LFP) = SO$, so ist $P \subseteq NP \subseteq PH = P$.
Ist $P = NP$, dann gilt $PH = NP$, also $FO(LFP) = SO$. $\square$

Korollar 4.5. *PH ist gleich die Menge aller Boolsche Abfragen die von ein CRAM erkennbar sind durch Verwendung von exponentiell vielen Prozessoren und konstanter Zeit,*

$$PH = \bigcup_{k=1}^{\infty} CRAM\text{-}PROC[1, 2^{n^k}]$$

Beweis. Die Inklusion $SO \subseteq CRAM\text{-}PROC[1, 2^{n^{O[1]}}]$ folgt wie im Beweis von Lemma 5.4 in [1]. Eine Prozessornummer ist nun groß genug um allen Relationsvariablen und Variablen 1. Stufe Werte zu geben. Also kann CRAM wie in Lemma 5.4 in [1] jeden Quantor 1. oder 2. Stufe in drei Schritte evaluieren.
Die Inklusion $CRAM\text{-}PROC[1, 2^{n^{O[1]}}]$ folgt wie im Beweis von Lemma 5.3 in [1]. Der einzige Unterschied ist, dass wir nun Variablen 2. Stufe benutzen um die Prozessornummer zu spezifizieren. □

Korollar 4.5 kann erweitert werden zu,

Korollar 4.6. *Für alle konstruierbaren $t(n)$,*

$$SO[t(n)] = CRAM\text{-}PROC[t(n), 2^{n^{O[1]}}].$$

Literatur

[1] Neil Immerman. *Descriptive Complexity.* Springer New York, 1999