6 Beweiskomplexität

Ist eine Formel F erfüllbar, so lässt sich diese Tatsache leicht durch Angabe einer erfüllenden Belegung α bezeugen, man sagt α ist ein *Zertifikat* dafür dass F erfüllbar ist. Die Beweiskomplexität befasst sich mit der Frage nach Zertifikaten für die Unerfüllbarkeit von Formeln.

Ein solches Zertifikat für die Unerfüllbarkeit von F kann gegeben sein durch einen Beweis für $\neg F$ in einem Beweisalkül für die Aussagenlogik, oder auch durch die Wertetabelle von F. Wir verlangen nur, dass es effizient überprüfbar sein soll, ob ein vorgelegtes angebliches Zertifikat tatsächlich eines ist.

Definition. *Ein Beweissystem ist eine zweistellige Relation* R*, die in polynomieller Zeit entscheidbar ist, mit der Eigenschaft: es gibt ein* p *mit* $R(p, F)$ *gdw.* F *unerfullbar ist.*

Ist R ein Beweissystem, dann nennen wir ein p mit $R(p, F)$ einen R-Beweis für F. Die zentrale Frage der Beweiskomplexität ist: wie groß müssen Beweise für eine unerfüllbare Formel F sein, in Abhängigkeit von $|F|$?

Ein Beweissystem R ist *polynomiell beschränkt*, wenn es eine Polynom $q()$ gibt, so dass für jede unerfüllbare Formel F einen Beweis p mit $R(p, F)$ und $|p| \leq q(|F|)$ gibt.

Der folgende Beobachtung von Cook und Reckhow stellt einen Zusammenhang zwischen der Beweiskomplexität und zentralen Fragen der Komplexitätstheorie her.

Satz 27. *Es gibt ein polynomiell beschränktes Beweissystem genau dann, wenn NP = co-NP ist.*

Beweis. Die Idee des Beweises ist sehr einfach: falls R ein polynomiell beschränktes Beweissystem ist, kann eine nichtdeterministische Maschine für eine Formel F einen (polynomiell langen) Beweis in polynomieller Zeit raten und überprüfen, also ist das Komplement von SAT in NP, und somit NP = co-NP.

Umgekehrt gibt es unter der Annahme NP = co-NP eine nichtdeterministische Maschine M, die in polynomieller Zeit unerfüllbare Formeln erkennt. Daraus erhält man ein Beweissystem, in dem ein Beweis für F eine akzeptierende Berechnung der Maschine M bei Eingabe F ist. $\square$

Diese Beobachtung führt zu einem Forschungsprogramm (*Cook's program*), in dem versucht wird, für immer stärkere Beweissysteme untere Schranken zu zeigen, die beweisen, dass diese nicht polynomiell beschränkt sind. Dier

Hoffnung dabei ist, dass irgendwann hinreichend Einsichten gewonnen und Techniken entwickelt werden, um zu zeigen, dass es kein polynomiell beschränktes Beweissystem geben kann, womit NP $\neq$ co-NP und somit P $\neq$ NP folgen würde.

Neben dieser komplexitätstheoretischen Motivation ist die Untersuchung von Beweissystemen auch für das Studium der Laufzeit von SAT-Algorithmen von Interesse.

Ist A ein vollständiger SAT-Algorithmus, dann können wir aus diesem ein Beweissystem R_A definieren, in dem ein R_A Beweis für F ein Protokoll eines Laufes von A bei Eingabe F ist, der am Ende "UNSAT" ausgibt. Wegen der Korrektheit von A ist dies ein Beweissystem im obigen Sinne, da ein Protokoll in polynomieller Zeit überprüft werden kann, wenn es hinreichend detailliert ist.

Untere Schranken an die Länge von R_A-Beweisen implizieren nun offensichtlich ebensolche unteren Schranken an die Laufzeit von A. Die Beweissysteme R_A wirken auf den ersten Blick künstlich, sind aber in vielen relevanten Fällen äquivalent zu natürlichen, wohluntersuchten Beweissystemen.

6.1 Resolution und DLL

Ein besonders wichtiges und viel untersuchtes Beweissystem ist der Resolutionskalkül. Dieser ist wegen seiner engen Beziehung zu DLL-Algorithmen für uns von besonderem Interesse.

Ein Resolutionsherleitung einer Klausel C aus einer Formel F in KNF ist eine Folge

$$C_1, \ldots, C_s$$

von Klauseln mit den Eigenschaften:

- $C_s = C$
- für alle $1 \leq i \leq s$ gilt eine der folgenden Aussagen:
  - C_i ist eine Klausel in F
  - es gibt $j < j' < i$, so dass C_i Resolvente von C_j und $C_{j'}$ ist.

Ein Resolutionsbeweis für F ist eine Herleitung der leeren Klausel 0 aus F. Die Größe einer Resolutionsherleitung ist die Anzahl s der Klauseln darin.

Wie auf Seite 18 gezeigt wurde, folgt C aus F, falls C aus F herleitbar ist. Insbesondere ist F unerfüllbar, falls F einen Resolutionsbeweis hat. Die Umkehrung, dass jede unerfüllbare Formel einen Resolutionsbeweis hat, folgt aus der unten gezeigten stärkeren Aussage, somit ist Resolution ein Beweissystem im obigen Sinn.

Proposition 28. *Wenn ein DLL-Algorithmus bei Eingabe einer unerfüllbaren KNF-Formel* F r *rekursive Aufrufe durchführt, dann hat* F *einen Resolutionsbeweis der Größe höchstens* r.

Beweis. Wir zeigen die folgende allgemeinere Behauptung durch Induktion nach r: Falls der Aufruf $\mathrm{DLL}(F, \alpha)$ zu r rekursiven Aufrufen führt, so gibt es eine Klausel C mit $C\alpha = 0$, derart dass C eien baumförmige Resolutionsherleitung aus F der Größe höchstens r hat.

Der Induktionsanfang ist der Fall $r = 1$, also wenn $\mathrm{DLL}(F, \alpha)$ sofort terminiert. Da F unerfüllbar ist, ist $F\alpha = 0$, also gibt es eine Klausel C in F mit $C\alpha = 0$. Offenbar ist C aus F in einem Schritt herleitbar.

Sei also $r > 1$, dann führt der Aufruf $\mathrm{DLL}(F, \alpha)$ zu zwei rekursiven Aufrufen $\mathrm{DLL}(F, \alpha \cup [x \leftarrow i])$ für eine Variable x und $i = 0, 1$, die jeweils r_i rekursive Aufrufe ausführen, wobei $r = r_0 + r_1 + 1$ ist.

Nach Induktionshypothese gibt es für $i = 0, 1$ jeweils eine Klausel C_i mit $C_i\alpha[x \leftarrow i] = 0$, die in r_i Schritten aus F herleitbar ist.

Falls nun C_0 das Literal x nicht enthält, dann ist $C\alpha = 0$, also setzen wir $C = C_0$. Sonst setzen wir $C = C_1$, falls $\bar{x}$ in C_1 nicht vorkommt.

Andernfalls erhalten wir C als Resolvente von C_0 und C_1 nach der Variablen x. Wegen der Korrektheit der Resolution gilt $C\alpha = 0$, und wir haben C in $r_0 + r_1 + 1 = r$ Schritten aus F hergeleitet.

Der äußerste Aufruf des DLL-Algorithmus ist nun $\mathrm{DLL}(F, \emptyset)$, also muss für die Klausel C dort gelten $C\emptyset = 0$, also $C = 0$, somit erhalten wir einen Resolutionsbeweis für F der Größe höchstens r. □

Wegen der Vollständigkeit des DLL-Algorithmus folgt insbesondere, dass jede unerfüllbare Formel in KNF einen Resolutionsbeweis hat.

Die obige Proposition gilt auch für DLL-Verfahren mit Klausellernen und zufälligen Restarts (s. Kapitel 3), und somit ist Resolution äquivalent zum Beweissystem P_A für die in den Kapiteln 2 und 3 vorgestellten Algorithmen A.

6.2 Untere Schranken für Resolution

Wir wollen nun zeigen, dass Resolution nicht polynomiell beschränkt ist. Dazu müß eine Familie von Formeln F_n angegeben werden, die selbst nicht zu groß werden, also $|F_n| \leq n^{O(1)}$, derart dass aber jeder Resolutionsbeweis für F_n exponentiell in n groß ist.

Es gibt viele Beispiele für Familien solcher Formeln, das im Folgenden ist das historisch erste solche, und ist durch das Schubfachprinzip motiviert.

Die Formel PHP_n für $n \geq 2$ ist eine Formel in KNF in den $n(n-1)$ Variablen $x_{i,j}$ für $1 \leq i \leq n$ und $1 \leq j \leq n-1$. Das Dirichlet'sche Schubfachprinzip besagt, dass n Dinge nicht in $n-1$ Fächer gepackt werden können, ohne dass in einem Fach zwei Dinge liegen. Eine Belegung der Variable $x_{i,j}$ mit 1 interpretieren wir so, dass das Ding i im Fach j liegt.

Die Formel PHP_n besteht aus den poitiven Klauseln

$$P_i \;:=\; x_{i,1} \vee \ldots \vee x_{i,n-1}$$

für $1 \leq i \leq n$, die ausdrückt, dass das Ding i in einem der Fächer liegen muss, sowie den negativen Klauseln

$$H_{i,i',j} \;:=\; \bar{x}_{i,j} \vee \bar{x}_{i',j}$$

für $1 \leq i < i' \leq n$ und $1 \leq j \leq n-1$, die sagt, dass die Dinge i und i' nicht im selben Fach j liegen.

Wegen des Schubfachprinzips ist PHP_n offensichtlich unerfüllbar, und $|\mathrm{PHP}_n| \leq O(n^3)$. Wir zeigen die folgende untere Schranke.

Satz 29. *Für hinreichend große n enthält jeder Resolutionsbeweis für PHP_n mindestens $2^{n/20}$ Klauseln.*

Ein Matching ρ ist eine Menge von Paarren

$$\big\{(i_1, j_1), \ldots, (i_k, j_k)\big\} \subset \{1, \ldots, n\} \times \{1, \ldots, n-1\}$$

derart dass die sowohl i_ν als auch die j_ν für $1 \leq \nu \leq k$ untereinander paarweise verschieden sind. Die Größe $|\rho|$ von ρ ist k.

Ein Matching induziert eine Bewertung der Variablen von PHP_n wie folgt:

$$\rho(x_{i,j}) = \begin{cases} 1 & \text{falls } (i,j) \in \rho \\ 0 & \text{falls} (i,j') \in \rho \text{ mit } j \neq j' \\ & \text{oder } (i',j) \in \rho \text{ mit } i \neq i' \\ \text{undefiniert} & \text{sonst.} \end{cases}$$

Wir unterscheiden im Folggenden nicht zwischen einem Matching und der dadurch induzierten Bewertung.

Eine *kritische* Bewertung ist ein Matching ρ mit $|\rho| = n-1$, eine solche Bewertung ist immer total. Eine kritische Bewertung erfüllt alle negativen Klauseln $H_{i,i',j}$, und alle positiven Klauseln P_i bis auf eine.

Für eine Formel F und Klausel C bedeute $F \models_{kr} C$, dass für jede kritische Bewertung ρ mit $\rho \models F$ auch $\rho \models C$ gilt. Falls C aus F folgt, dann gilt auch $F \models_{kr} C$, aber nicht umgekehrt, denn z.B. gilt $P_1, \ldots, P_n \models_{kr} 0$.

Wir bezeichnen im Folgenden mit PA_n die Menge $\{P_1, \ldots, P_n\}$ der positiven Klauseln in PHP_n.

Sei F eine Menge von positiven Klauseln. Ein monotoner Beweis fur F ist eine Folge

$$C_1, \ldots, C_s$$

von positiven Klauseln mit den Eigenschaften:

- $C_s = 0$
- für alle $1 \leq i \leq s$ gilt eine der folgenden Aussagen:
  - C_i ist eine Klausel in F
  - $\rho \models C_i$ für alle kritischen Bewertungen ρ
  - es gibt $j < j' < i$ mit $C_j \wedge C_{j'} \models_{kr} C_i$.

Lemma 30. *Wenn* PHP_n *einen Resolutionsbeweis der Länge* s *hat, dann hat* PA_n *einen monotonen Beweis der Länge* s.

Beweis. Für eine Klausel C sei C^+ die Menge der positiven und C^- die Menge der negativen Literale in C. Die positive KLausel C^m ist definiert durch

$$C^M := \bigvee_{x_{i,j} \in C^+} x_{i,j} \vee \bigvee_{\bar{x}_{i,j} \in C^-} \bigvee_{i' \neq i} x_{i',j} \, .$$

Offensichtlich sind C uund C^m äquivalent bzgl. kritischer Bewertungen, d.h. $C \models_{kr} C^m$ und $C^m \models_{kr} C$.

Nun gilt: ist $R = C_1, \ldots, C_s$ ein Resolutionsbeweis für PHP_n der Grösse s, dann ist $R^m = C_1^m, \ldots, C_s^m$ ein monotoner Beweis für PA_n. Offensichtlich is $C_s^m = 0$, da $C_s = 0$. Wir verifizieren die Eigenschaften eines monotonen Beweises für $1 \leq i \leq s$:

- Ist C_i eine positive Klausel P_j in PHP_n, dann ist $C_i^m = C_i$ in PA_n.
- Ist C_i eine negative Klausel $H_{j,j',k}$ in PHP_n, dann gilt $\rho \models C_i$ für jede kritische Bewertung ρ, also auch $\rho \models C_i^m$.
- Ist C_i Resolvente von C_j und $C_{j'}$, dann gilt $C_j, C_{j'} \models C_i$. Sei ρ kritisch mit $\rho \models C_j^m \wedge C_{j'}^m$. Dann gilt auch $\rho \models C_j \wedge C_{j'}$, also auch $\rho \models C_i$, und somit $\rho \models C_i^m$. Also gilt insgesamt $C_j^m, C_{j'}^m \models_{kr} C_i$.

Damit ist R^m ein monotoner Beweis für PA_n der Gröse s. $\square$

Damit reicht es aus, eine untere Schranke an die Größe von monotonen Beweisen für PA_n zu zeigen. Diese untere Schranke folgt aus zwei Lemmas: einerseits werden wir zeigen, dass jeder monotone Beweis für PA_n eine große Klausel, d.h. eine mit vielen Variablen, enthalten muss. Andererseits lässt sich aus jedem kurzen Beweis einer gewinnen, in dem alle Klauseln klein sind. Dieser Schritt wird durch Anwendung einer durch ein Matching gegebenen Bewertung erreicht:

Sei $R = C_1, \dots C_s$ ein monotoner Beweis für PA_n, und ρ ein Matching der Größe $|\rho| = k$. Dann ist $R\rho := C_1\rho, \dots, C_s\rho$ ein monotoner Beweis für $PA_n\rho$, und durch Umbenennung der Variablen erhalten wir daraus einen monotonen Beweis für PA_{n-k}.

Lemma 31. *Ist R ein monotoner Beweis für PA_n der Größe $|R| < 2^{n/20}$, dann gibt es ein Matching ρ mit $\rho \leq 0.329\,n$ derart dass $R\rho$ keine Klausel C mit mindestens $n^2/10$ Variablen enthält.*

Beweis. Eine Klausel C in R heiße breit, wenn sie mindestens $n^2/10$ Variablen enthält, insbesondere sind dies mehr als $1/10$ aller Variablen. Daher ist die Wahrscheinlichkeit, dass eine zufällig gewählte Variable in einer festen breiten Klausel vorkommt, mindestens $1/10$. Dies heißt umgekehrt, dass es eine Variable gibt, die in mindestens $1/10$ der breiten KLauseln vorkommt.

Wir bauen das matching ρ greedy induktiv auf wie folgt:

$$\begin{aligned}
\rho_0 &:= \emptyset \\
\rho_{k+1} &:= \rho_k \cup \{(i,j)\} \qquad && \text{für diejenige Variable } x_{i,j}\text{, die in den meisten} \\
& && \text{breiten Klauseln in } P\rho_k \text{ vorkommt}
\end{aligned}$$

Sei s_k die Anzahl der breiten Klauseln in $R\rho_k$, dann gilt $s_0 \leq s$ und $s_{k+1} \leq \frac{9}{10}s_k$. Also ist $s_r = 0$ für $r := \lceil \log_{10/9} s \rceil$, und wir setzen $\rho := \rho_r$. Für die Größe $|\rho|$ erhalten wir

$$|\rho| = r = \log_{10/9} s \leq ((\log_{10/9} 2)/20)n < 0.329n$$

und damit folgt die Behauptung. □

Lemma 32. *Ist R monotoner Beweis für PA_n, dann enthält R eine Klausel mit mindestens $2n^2/9$ Variablen.*

Beweis. Wir definieren ein Maß μ auf den Klauseln in R durch

$$\mu(C) := \min\{\, |F| \,;\, F \subseteq PA_n \text{ und } F \models_{kr} C \,\}$$

Offenbar gilt $\mu(P_i) = 1$, $\mu(H^m_{i,i',j}) = 0$ und $\mu(0) = n$, außerdem gilt für Klauseln C_1, C_2, D mit $C_1, C_2 \models_{kr} D$ dass $\mu(D) \leq \mu(C_1) + \mu(C_2)$. Daher gibt es eine Klausel C in R mit $n/3 < \mu(C) \leq 2n/3$.

Sei $F \subset PA_n$ minimal mit $F \models_{kr} C$, also $n/3 < |F| \leq 2n/3$. Wir zeigen, dass C mindestens $|F|(n - |F|) \geq 2n^2/9$ Variablen enthält.

Sei $1 \leq i \leq n$ mit $P_i \in F$, und sei ρ eine kritische Bewertung mit $\rho \not\models P_i$ und $\rho \not\models C$. Ein solches ρ existiert, da F minimal mit $F \models_{kr} C$ ist.

Sei weiter $1 \leq j \leq n-1$ mit $P_j \notin F$, und sei k_j derart dass $(j, k_j) \in \rho$. Definiere

$$\rho_j := (\rho \setminus \{(j, k_j)\}) \cup \{(i, k_j)\}$$

dann gilt $\rho_j \models F$, also $\rho_j \models C$, da $F \models_{kr} C$. Dies kann aber nur dadurch verursacht sein, dass C die Variable x_{i,k_j} enthält, da dies die einzige Variable ist, die von ρ_j, aber nicht von ρ auf 1 gesetzt wird.

Da i und j mit $P_i \in F$ und $P_j \notin F$ beliebig waren, gilt dies für jedes soche Paar i, j, und somit enthält C mindestens $|F|(n - |F|)$ Variablen. □

Damit können wir nun den Satz beweisen:

Beweis des Satzes 29. Sei R ein monotoner Beweis für PA_n, und sei R von der Größe $|R| < 2^{n/20}$. Nach Lemma 31 gibt es ein Matching ρ mit $|\rho| < 0.329\,n$ derart dass $R\rho$ keine Klausel C mit mindestens $n^2/10$ Variablen enthält.

Andererseits ist $R\rho$ ein monotoner Beweis für $PA_{n'}$, wobei $n' \leq 0.671\,n$, daher enthält R_ρ nach Lemma 32 eine Klausel, die mindestens

$$2(0.671\,n)^2/9 > 2(0.45n^2)/9 = n^2/10$$

Variablen enthält, im Widerspruch zum Obigen. Daher ist $|R| \geq 2^{n/20}$. □

6.3 Interpolation für Resolution

Seien A und B Formeln in KNF mit $V(A) \subseteq \{\vec{x}, \vec{y}\}$ und $V(B) \subseteq \{\vec{x}, \vec{z}\}$ für disjunkte Variablenlisten $\vec{x}, \vec{y}, \vec{z}$. Wenn $A \wedge B$ unerfüllbar ist, dann gilt für jede Bewertung α der gemeinsamen Variablen $\vec{x}$, also mit $\operatorname{dom} \alpha = \{\vec{x}\}$, dass entweder $A\alpha$ oder $B\alpha$ unerfüllbar ist. Wäre nämlich $\beta \models A\alpha$ und $\gamma \models B\alpha$, dann auch $\alpha \cup \beta \cup \gamma \models A \wedge B$.

Eine *Interpolante* C für A und B ist eine Formel C mit $V(C) \subseteq \{\vec{x}\}$, die für jedes α entscheidet, welcher der zwei Fälle eintritt, also mit der Eigenschaft:

$$\begin{array}{lcl} C\alpha = 0 & \Rightarrow & A\alpha \text{ unerfüllbar} \\ C\alpha = 1 & \Rightarrow & B\alpha \text{ unerfüllbar} \end{array}$$

Satz 33. *Falls für alle Formeln A, B wie oben eine Interpolante C existiert mit*

- *Formelgröße* $|C| \leq |A \wedge B|^{O(1)}$, *dann ist* $NP \cap$ *co-*$NP \subseteq NC^1$.
- *Schaltkreisgröße* $|C| \leq |A \wedge B|^{O(1)}$, *dann ist* $NP \cap$ *co-*$NP \subseteq P/poly$.

Allerdings zeigt der folgende Satz, dass sich die Schaltkreisgröße durch die Größe eines Resolutionsbeweises für $A \wedge B$ beschränken lässt.

Satz 34. *Seien* A, B *wie oben. Falls* $A \wedge B$ *einen Resolutionsbeweis* R *hat, dann gibt es eine Interpolante* C *der der Schaltkreisgröße* $O(s)$. *Ferner kann* C *aus* R *in linearer Zeit berechnet werden.*
Außerdem gilt: falls die Variablen $\vec{x}$ *in* A *nicht negativ vorkommen, dann ist* C *ein monotoner Schaltkreis.*

Beweis. Zunächst wird für jedes α der Beweis $R = C_1, \ldots, C_s$ in zwei Schritten in einen Beweis für entweder $A\alpha$ oder $B\alpha$ transformiert. In einem dritten Schritt wird dann der Schaltkreis konstruiert.
Im ersten Schritt wird jede Klausel C in R durch eine andere Klausel C' ersetzt, und diese in rot oder blau gefärbt, so dass folgende Invarianten gelten:

- blaue Klauseln enthalten keine der Variablen $\vec{z}$,
- rote Klauseln enthalten keine der Variablen $\vec{y}$.

Für eine Klausel C aus A wird $C' = C$ gesetzt und diese blau gefärbt. Für C aus B wird ebenso $C' = C$ gesetzt, und diese rot gefärbt. Für eine Klausel C, die durch Resolution aus D_1 und D_2 entsteht, werden drei Fälle unterschieden, je nach der resolvierten Variable.
Fall 1: die resolvierte Variable ist ein y:

- Falls eines der D_i' für $i = 1, 2$ rot ist, setze $C' = D_i'$ (und somit ist C' auch rot).
- Sonst sind beide D_i' blau. Falls eines der D_i' die Variable y nicht enthält, so setze $C' = D_i'$ (welches somit blau ist).
- Andernfalls ist C' die Resolvente von D_1' und D_2' (nach der Variablen y) und wird blau gefärbt.

Fall 2: die resolvierte Variable ist ein z. Dieser Fall ist dual zu Fall 1:

- Falls eines der D_i' für $i = 1, 2$ blau ist, setze $C' = D_i'$ (und somit ist C' auch blau).

- Sonst sind beide D'_i rot. Falls eines der D'_i die Variable z nicht enthält, so setze $C' = D'_i$ (welches somit rot ist).
- Andernfalls ist C' die Resolvente von D'_1 und D'_2 (nach der Variablen z) und wird rot gefärbt.

Fall 3: die resolvierte Variable ist ein x. O.B.d.A. enthalte D_1 das Literal x und D_2 das Literal $\bar{x}$. Dann setze

$$C' := \begin{cases} D'_1 & \text{falls } \alpha(x) = 0 \\ D'_2 & \text{falls } \alpha(x) = 1 \end{cases}$$

und damit erhält C' auch die Farbe des jeweiligen D'_i.

Das Ergebnis ist eine Folge $R' = C'_1, \ldots, C'_s$ von Klauseln. Durch eine einfache Induktion zeigt man die folgenden Eigenschaften für alle $i \leq s$:

- Jedes Literal in den Variablen $\vec{y}$, $\vec{z}$, dass in C'_i vorkommt, kommt auch schon in C_i vor.
- Jedes Literal in den Variablen $\vec{x}$, dass in C'_i vorkommt, aber nicht in C_i, wird von α mit 0 bewertet.

Im zweiten Schritt wird nun darauf die bewertung α angewendet, wir betrachten also $R'\alpha = C'_1\alpha, \ldots, C'_s\alpha$. Wegen der obigen Eigenschaften ist dann für jedes i die Klausel $C'_i\alpha$ eine Teilklausel von $C_i\alpha$, und insbesondere ist $C'_s\alpha = 0$. Ist also C'_s blau gefärbt, so bilden die blauen Klauseln einen Resolutionsbeweis für $A\alpha$, andernfalls bilden die roten Klauseln einen Resolutionsbeweis für $B\alpha$.

Daher ist klar, dass der zu konstruierende Schaltkreis nur entscheiden muss, ob die Klausel C'_s rot oder blau ist. Wir bauen im Schritt 3 den Schaltkreis so auf, dass wir für jede Klausel C in R ein Gatter c konstruieren, dass den Wert 0 hat, falls C' blau ist, und den Wert 1, falls C' rot ist. Dabei werden die selben Fälle unterschieden wie im Schritt 1.

Ist C in A, so ist C' blau, also setze $c = 0$. Ebenso wird für C in B die Konstante $c = 1$ gesetzt.

Im Fall 1 ist C' rot, wenn eines der D'_i rot ist, also setzen wir $c := d_1 \vee d_2$. Im Fall 2 dagegen ist C' nur dann rot, wenn beide D'_i rot sind, also setzen wir $c := d_1 \wedge d_2$.

Im Fall 3 dagegen hat C' die Farbe von D'_1, wenn $\alpha(x) = 0$ ist, und die Farbe von D'_2 sonst. Daher setzen wir

$$c \;:=\; \text{if } x \text{ then } d_2 \text{ else } d_1 \;=\; (x \wedge d_2) \vee (\bar{x} \wedge d_1) \;.$$

Offensichtlich ist der Schaltkreis durch die angegebene Konstruktion in linearer Zeit aus R konstruierbar.

Falls nun in A die Variablen $\vec{x}$ nicht vorkommen, ist die Konstruktion wie folgt zu modifizieren: die erste der beiden Invarianten im Schritt 1 wird verstärkt zu:

- blaue Klauseln enthalten keine der Variablen $\vec{z}$, und keine der Variablen $\vec{x}$ negativ.

Diese Invariante gilt nach Voraussetzung für die Klauseln in A, und es ist leicht zu sehen, dass sie durch die Konstruktion in den Fällen 1 und 2 erhalten bleibt.

Fall 3 wird wie folgt modifiziert: Falls D_2' blau ist, setze $C' = D_2'$, andernfalls wird wie vorher verfahren. Im Schaltkreis können wir dann $c = (x \vee d_1) \wedge d_2$ setzen.

Dieser Wert unterscheidet sich von dem in der ursprünglichen Konstruktion ($c = (x \wedge d_2) \vee (\bar{x} \wedge d_1)$) nur für die Eingabe $x = 0$, $d_1 = 1$ und $d_2 = 0$, die genau der obigen Situation entspricht: D_2' ist blau und D_1' rot, und da $x = 0$ ist, würde die alte Konstruktion $C' = D_1'$ setzen, womit C' rot und daher $c = 1$ wäre. Jetzt hingegen ist $c = 0$, da $C' = D_2'$ blau ist. □

Als Anwendung wollen wir nun nochmals eine untere Schranke an die Größe von Resolutionsbeweisen zeigen. Wir betrachten Schaltkreise, die Probleme auf ungerichteten Graphen lösen, in den Variablen $x_{i,j}$ für $1 \leq i < j \leq n$. Jede Bewertung α dieser Variablen $\vec{x}$ definiert einen Graphen $G(\alpha)$ auf n Knoten, wobei zwischen i und j genau dann eine Kante ist, wenn $\alpha(x_{i,j}) = 1$ ist.

Ein bekannter Satz aus der Komplexitätstheorie zeigt, dass monotone Schaltkreise, die das NP-vollständige Problem der Erkennung von Cliquen in Graphen lösen, exponentiell groß sein müssen.

Satz 35. *Ist C ein monotoner Schaltkreis, so dass für ein $k \leq n^{1/4}$ gilt:*

- *falls $G(\alpha)$ eine k-Clique enthält, ist $C\alpha = 1$,*
- *falls $G(\alpha)$ $(k-1)$-färbbar ist, dann ist $C\alpha = 0$,*

dann ist $|C| \geq n^{\Omega(\sqrt{k})}$.

Man beachte, dass die untere Schranke für große Werte von k, etwa $k = \Omega(n)$, tatsächlich exponentiell ist.

Es werden nun Formeln A und B konstruiert, für die jeder interpolierende Schaltkreis die Vorraussetzung dieses Satzes erfüllt, also k-Cliquen von $(k-1)$-färbbaren Graphen unterscheidet.

Die Formel A enthält neben den Variablen $\vec{x}$ Variablen $y_{i,j}$ für $1 \le i \le k$ und $1 \le j \le n$. Eine Bewertung dieser Variablen codiert eine Teilmenge von k Knoten: $y_{i,j} = 1$ wenn das ite Element der Menge j ist. A besteht aus den Klauseln:

$$
\begin{array}{ll}
y_{i,1} \vee \ldots \vee y_{i,n} & \text{für } 1 \le i \le k \\
\bar{y}_{i,j} \vee \bar{y}_{i',j} & \text{für } 1 \le i < i' \le k \text{ und } 1 \le j \le n \\
\bar{y}_{i,j} \vee \bar{y}_{i',j'} \vee x_{j,j'} & \text{für } 1 \le i, i' \le k\,, i \neq i' \text{ und } 1 \le j < j' \le n
\end{array}
$$

Die erste Gruppe von Klauseln garantiert, dass jeder Position $1 \le i \le k$ ein Knoten zugeordnet wird, die zweite Gruppe erzwingt, dass die Knoten in der Menge paarweise verschieden sind. Die dritte Gruppe schließlich drückt aus, dass je zwei Knoten in der Menge im Graphen benachbart sind, dass also die Menge eine Clique bildet. Offenbar kommen die Variablen $\vec{x}$ in A nur positiv vor.

Die Formel B enthält neben den Variablen $\vec{x}$ Variablen $z_{i,j}$ für $1 \le i \le n$ und $1 \le j \le k-1$. Eine Bewertung dieser Variablen codiert eine Färbung des Graphen mit $k-1$ Farben. B besteht aus den Klauseln:

$$
\begin{array}{ll}
z_{i,1} \vee \ldots \vee z_{i,k-1} & \text{für } 1 \le i \le n \\
\bar{z}_{i,j} \vee \bar{z}_{i',j} \vee \bar{x}_{i,i'} & \text{für } 1 \le i < i' \le n \text{ und } 1 \le j \le k-1
\end{array}
$$

Die erste Gruppe von Klauseln garantiert, dass jedem Knoten $1 \le i \le n$ eine Farbe zugeordnet wird. Die zweite Gruppe drückt aus, dass benachbarte Knoten verschiedene Farben haben.

Satz 36. *Jeder Resolutionsbeweis für die oben konstruierte Formel* $A \wedge B$ *hat mindestens die Größe* $n^{\Omega(\sqrt{k})}$.

Beweis. Sei R ein Resolutionsbeweis für $A \wedge B$ der Größe s. Nach Satz 34 existiert ein monotoner interpolierenddder Schaltkreis C der Größe $O(s)$.

Der Schaltkreis C hat die Eigenschaft, dass für jede Bewertung α der Variabeln $\vec{x}$ gilt:

- falls $G(\alpha)$ eine k-Clique enthält so ist $A\alpha$ erfüllbar, also muss $C\alpha = 1$ sein,
- falls $G(\alpha)$ $(k-1)$-färbbar ist, so ist $B\alpha$ erfüllbar, also muss $C\alpha = 0$ sein.

Also muss nach Satz 36 $s \ge n^{\Omega(\sqrt{k})}$ sein. ☐