Aussagenlogische Erfüllbarkeit, oder: wie man Sudokus löst

Teilnehmer:

7 Schülerinnen und Schüler	Herder-Gymnasium Käthe-Kollwitz-Gymnasium
mit tatkräftiger Unterstützung durch:	
Christoph Werner	Humboldt-Universität zu Berlin
Gruppenleiter:	
Lucas Heimberg	Humboldt-Universität zu Berlin

1. Einleitung

Das japanische Rätsel *Sudoku* (siehe Abbildung) ist jedem bekannt: Die unbeschrifteten Felder eines Quadrats aus 9×9 Feldern müssen so mit den Ziffern 1–9 beschriftet werden, dass in jeder Zeile, in jeder Spalte, und in jedem der 3×3 Felder großen Teilblöcke jede Ziffer genau einmal vorkommt.

Sudokus in Zeitungen oder Rätselheften sind meist eindeutig (d.h., es gibt nur eine Lösung) und deterministisch (d.h., in jedem Lösungsschritt gibt es ein noch leeres Feld, dessen Beschriftung durch die bereits beschrifteten Felder eindeutig bestimmt ist).

Wir entwickeln eine Methode, mit der beliebige Sudokus gelöst werden können – d.h., insbesondere Sudokus, deren Lösung „raten“ bzw. „herumprobieren“ erfordert. Zu diesem Zweck werden wir Sudokus als aussagenlogische Formeln repräsentieren und Algorithmen auf diese anwenden, die das Erfüllbarkeitsproblem für aussagenlogische Formeln entscheiden.

	3							
			1	9	5			
	9	8					6	
8				6				
4					3			1
				2				
	6					2	8	
			4	1	9			5
							7	

2. Aussagenlogik

Im Folgenden führen wir Syntax und Semantik aussagenlogischer Formeln ein. Aussagenlogische Formeln bestehen aus *Aussagensymbolen* und den *Booleschen Konstanten* **1** und **0** („wahr“ und „falsch“), die durch die *Junktoren* „nicht“, „und“ sowie „oder“ verbunden werden. Werden die Aussagensymbole einer aussagenlogischen Formel durch die *Wahrheitswerte* „wahr“ und „falsch“ interpretiert, so bestimmen die Junktoren den Wahrheitswert der gesamten Formel.

2.1. Syntax der Aussagenlogik

Wir repräsentieren Aussagensymbole durch Großbuchstaben, die wir gelegentlich zusätzlich mit Indizes versehen. Mit AS bezeichnen wir die *Menge aller Aussagensymbole*.

Die Menge AL der aussagenlogischen Formeln ist rekursiv definiert. Die Booleschen Konstanten **0** und **1** gehören zu AL. Jedes Aussagensymbol $X \in \mathrm{AS}$ ist auch eine aussagenlogische Formel. Die Junktoren werden wie folgt eingeführt:

- Ist $\varphi \in \mathrm{AL}$, dann ist auch $\neg\varphi \in \mathrm{AL}$. *(Negation)*
- Ist $\varphi \in \mathrm{AL}$ und $\psi \in \mathrm{AL}$, dann ist

  $(\varphi \wedge \psi) \in \mathrm{AL}$, *(Konjunktion)*

  $(\varphi \vee \psi) \in \mathrm{AL}$. *(Disjunktion)*

Der Lesbarkeit halber erlauben wir uns, Konjunktionen und Disjunktionen über Folgen $\varphi_1, \varphi_2, \ldots, \varphi_n$ aussagenlogischer Formeln analog zur Summenschreibweise der Arithmetik zu formulieren. D.h., wir schreiben

$$\bigwedge_{i=1}^{n} \varphi_i \quad \text{für} \quad (\ldots((\varphi_1 \wedge \varphi_2) \wedge \varphi_3) \ldots \wedge \varphi_n)$$

und

$$\bigvee_{i=1}^{n} \varphi_i \quad \text{für} \quad (\ldots((\varphi_1 \vee \varphi_2) \vee \varphi_3) \ldots \vee \varphi_n).$$

Eine *leere Konjunktion*, d.h., für $n = 0$, entspricht der aussagenlogischen Formel $\mathbf{1}$. Eine *leere Disjunktion* entspricht hingegen der aussagenlogischen Formel $\mathbf{0}$.

2.2. Semantik der Aussagenlogik

Eine *aussagenlogische Interpretation* ist eine Abbildung $\mathcal{I} \models \mathrm{AS} \to \{0, 1\}$. Die Abbildung $\mathcal{I}$ belegt bzw. interpretiert also jedes Aussagensymbol $X \in \mathrm{AS}$ mit einem der Wahrheitswerte 0 („falsch“) oder 1 („wahr“).

Der Wahrheitswert $[\![\varphi]\!]^{\mathcal{I}}$ einer aussagenlogischen Formel φ unter der Interpretation $\mathcal{I}$ ist rekursiv definiert: Es ist $[\![\mathbf{1}]\!]^{\mathcal{I}} = 1$ und $[\![\mathbf{0}]\!]^{\mathcal{I}} = 0$. Ist $\varphi = X$ für ein Aussagensymbol X, dann ist $[\![\varphi]\!]^{\mathcal{I}} = \mathcal{I}(X)$. Für aussagenlogische Formeln die mit Hilfe von Junktoren gebildet wurden gelten folgende Regeln:

- Ist $\varphi \in \mathrm{AL}$, so ist $[\![\neg\varphi]\!]^{\mathcal{I}} := \begin{cases} 1 & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = 0, \\ 0 & \text{sonst.} \end{cases}$ *(Negation)*
- Ist $\varphi \in \mathrm{AL}$ und $\psi \in \mathrm{AL}$, dann ist

$$[\![(\varphi \wedge \psi)]\!]^{\mathcal{I}} := \begin{cases} 1 & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = [\![\psi]\!]^{\mathcal{I}} = 1 \\ 0 & \text{sonst.} \end{cases} \quad \textit{(Konjunktion)}$$

$$[\![(\varphi \vee \psi)]\!]^{\mathcal{I}} := \begin{cases} 0 & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = [\![\psi]\!]^{\mathcal{I}} = 0 \\ 1 & \text{sonst.} \end{cases} \quad \textit{(Disjunktion)}$$

Wir nennen eine Interpretation $\mathcal{I}$ ein *Modell* von φ (kurz: $\mathcal{I} \models \varphi$), wenn $[\![\varphi]\!]^{\mathcal{I}} = 1$.

Eine aussagenlogische Formel φ ist *erfüllbar*, wenn sie mindestens ein Modell hat. Ansonsten nennen wir φ *unerfüllbar*.

Zwei aussagenlogische Formeln φ und ψ sind *äquivalent*, wenn sie von genau denselben Interpretationen erfüllt werden.

2.3. Das Erfüllbarkeitsproblem der Aussagenlogik

Das *Erfüllbarkeitsproblem der Aussagenlogik* hat folgende Form:

Eingabe Eine aussagenlogische Formel φ.

Frage Ist φ erfüllbar?

Ein naheliegender Algorithmus, um das Erfüllbarkeitsproblem der Aussagenlogik zu lösen besteht darin, für eine gegebene aussagenlogische Formel φ alle Interpretationen aufzulisten, die sich paarweise in der Belegung der Aussagensymbole aus φ unterscheiden, und zu überprüfen ob mindestens eine dieser Interpretationen ein Modell von φ ist. Dieser Algorithmus ist bekannt als *Wahrheitstafel-Algorithmus*. Leider ist der Algorithmus nicht effizient, da für eine aussagenlogische Formel φ mit n verschiedenen Aussagensymbolen 2^n Interpretationen untersucht werden müssen.

Aus dem bekannten *Satz von Cook und Levin (1971)* folgt, dass es wahrscheinlich keinen Algorithmus gibt, der das Erfüllbarkeitsproblem der Aussagenlogik wesentlich schneller löst als der Wahrheitstafel-Algorithmus (beispielsweise mit einem Zeitaufwand, der linear oder polynomiell mit der Anzahl der Aussagensymbole wächst):

Satz 1. *Das Erfüllbarkeitsproblem der Aussagenlogik ist NP-vollständig.*

Dies schließt jedoch nicht aus, dass es Algorithmen gibt, die das Erfüllbarkeitsproblem der Aussagenlogik für viele (wenn auch nicht für alle) aussagenlogische Formeln schneller lösen als der Wahrheitstafel-Algorithmus. Mit dem *DPLL-Algorithmus* werden wir später einen solchen Algorithmus kennenlernen.

3. Modellierung eines Sudokus

Im Folgenden werden wir beschreiben, wie aus einem gegebenen Sudoku eine aussagenlogische Formel φ konstruiert werden kann, so dass gilt:

Das Sudoku ist genau dann lösbar, wenn φ erfüllbar ist.

Da ein Aussagensymbol nur mit einem der zwei Werte 0 und 1 interpretiert werden kann, müssen wir die Beschriftung eines Feldes des Sudokus durch 9 verschiedene Aussagensymbole darstellen, die zu den möglichen Beschriftungen des Feldes korrespondieren. Für jede Wahl von $z, s, n \in \{1, \ldots, 9\}$ steht das Aussagensymbol $P_{z,s,n}$ von nun an für die Aussage:

Das Feld in Zeile z und Spalte s des Sudokus ist mit der Ziffer n beschriftet.

Ein Modell $\mathcal{I}\colon \mathrm{AS} \to \{0,1\}$ für φ repräsentiert also die Lösung, in der das Feld in Zeile z und Spalte s mit der (eindeutigen) Ziffer n beschriftet ist, für die $\mathcal{I}(P_{z,s,n}) = 1$ ist.

Wir können leicht eine aussagenlogische Formel konstruieren, die von einer Interpretation genau dann erfüllt wird, wenn die bereits vorgegebenen Felder die entsprechende Beschriftung tragen. Für das Sudoku auf Seite 52 tut dies die folgende aussagenlogische Formel (mit einer kleinen Einschränkung):

$$\begin{aligned}\varphi_P \;:=\;& P_{1,2,3} \wedge P_{2,4,1} \wedge P_{2,5,9} \wedge P_{2,6,5} \wedge P_{3,2,9} \wedge P_{3,3,8} \wedge P_{3,8,6} \\ & \wedge P_{4,1,8} \wedge P_{4,5,6} \wedge P_{5,1,4} \wedge P_{5,6,3} \wedge P_{5,6,1} \wedge P_{6,5,2} \\ & \wedge P_{7,2,6} \wedge P_{7,7,2} \wedge P_{7,8,8} \wedge P_{8,4,4} \wedge P_{8,4,1} \wedge P_{8,6,9} \wedge P_{8,9,5} \wedge P_{9,8,7}.\end{aligned}$$

Das Problem dieser Formel ist, dass sie auch „doppelte Beschriftungen“ erlaubt. D.h., sie wird auch von Interpretationen erfüllt, in denen mehr als eins der Aussagensymbole $P_{z,s,1}, \ldots, P_{z,s,9}$ für eine Wahl von $s, z \in \{1, \ldots, 9\}$ mit 1 interpretiert sind. Die folgende aussagenlogische Formel wird nur von Interpretationen erfüllt, die die Felder des Sudokus eindeutig beschriften:

$$\varphi_H \;:=\; \bigwedge_{z=1}^{9} \bigwedge_{s=1}^{9} \bigwedge_{n=1}^{8} \bigwedge_{n'=n+1}^{9} (\neg P_{z,s,n} \vee \neg P_{z,s,n'}).$$

Außerdem wollen wir nur Interpretationen als Lösungen für das Sudoku akzeptieren, in denen tatsächlich jedes Feld eine Beschriftung hat. Dies leistet die folgende aussagenlogische Formel:

$$\varphi_M \;:=\; \bigwedge_{z=1}^{9} \bigwedge_{s=1}^{9} \bigvee_{n=1}^{9} P_{z,s,n}.$$

Schließlich benötigen wir aussagenlogische Formeln, die die bekannten Regeln eines Sudokus ausdrücken.

Die folgenden Formeln fordern, dass in keiner Zeile beziehungsweise Spalte eine Ziffer mehr als einmal vorkommt:

$$\varphi_Z \;:=\; \bigwedge_{z=1}^{9} \bigwedge_{n=1}^{9} \bigwedge_{s=1}^{8} \bigwedge_{s'=s+1}^{9} \left(\neg P_{z,s,n} \vee \neg P_{z',s,n}\right) \qquad \varphi_S \;:=\; \bigwedge_{s=1}^{9} \bigwedge_{n=1}^{9} \bigwedge_{z=1}^{8} \bigwedge_{z'=z+1}^{9} \left(\neg P_{z,s,n} \vee \neg P_{z',s,n}\right)$$

Die Konjunktion $\varphi_B := \varphi_R \wedge \varphi_U$ der folgenden beiden Formeln fordert, dass in keinem 3×3 Block eine Ziffer mehr als einmal vorkommt. Die Formeln φ_R und φ_U besagen dabei jeweils, dass keine Ziffer in irgendeinem Block „weiter rechts“ beziehungsweise „weiter links“ von ihrem ersten Vorkommen im Block noch einmal vorkommt:

$$\varphi_R \;:=\; \bigwedge_{n=1}^{9} \bigwedge_{i=0}^{2} \bigwedge_{j=0}^{2} \bigwedge_{z=1}^{3} \bigwedge_{s=1}^{2} \bigwedge_{z'=1}^{3} \bigwedge_{s'=s+1}^{3} \left(\neg P_{3i+z,3j+s,n} \vee \neg P_{3i+z',3j+s',n}\right)$$

$$\varphi_U \;:=\; \bigwedge_{n=1}^{9} \bigwedge_{i=0}^{2} \bigwedge_{j=0}^{2} \bigwedge_{z=1}^{2} \bigwedge_{s=1}^{3} \bigwedge_{z'=z+1}^{3} \bigwedge_{s'=1}^{3} \left(\neg P_{3i+z,3j+s,n} \vee \neg P_{3i+z',3j+s',n}\right)$$

Mit Hilfe der gerade konstruierten Formeln können wir nun die Formel φ wie folgt definieren:

$$\varphi \;:=\; \varphi_P \wedge \varphi_H \wedge \varphi_M \wedge \varphi_Z \wedge \varphi_S \wedge \varphi_B.$$

Wie können wir feststellen, ob die Formel φ für ein gegebenes Sudoku erfüllbar ist, d.h., ob das Sudoku eine Lösung hat? Natürlich könnten wir den Wahrheitstafel-Algorithmus verwenden. Dies bedeutet jedoch einen riesigen Aufwand, da φ genau 9^3 Aussagensymbole verwendet und es somit

$$\begin{aligned} 2^{9^3} \;=\; & 28240139587082174969491088422046278633513539118515775246834019308626938303 \\ & 61198499905873920995229996970897865498283996578123296865878390947626553088 \\ & 48694610643079609148271612057263207249270352772375735947883453036573491 2 \end{aligned}$$

verschiedene Interpretationen für diese Aussagensymbole gibt!

In den folgenden Abschnitten beschäftigen wir uns mit Methoden, die (Un-)erfüllbarkeit einer aussagenlogischen Formel zu entscheiden. Obwohl im schlimmsten Fall keine dieser Methoden schneller ist als der Wahrheitstafel-Algorithmus, sind sie doch in vielen praktischen Fällen deutlich effizienter.

4. Konjunktive Normalform

Die in den folgenden Abschnitten dargestellten Methoden setzen vorraus, dass die zu untersuchende Formel bestimmte syntaktische Eigenschaften erfüllt, d.h., dass sie in konjunktiver Normalform vorliegt.

Ein *Literal* ist eine Formel der Form X oder $\neg X$ für ein Aussagensymbol $X \in \mathrm{AS}$. Eine Formel ist in *konjunktiver Normalform (KNF)*, wenn sie eine Konjunktion von Disjunkten von Literalen ist, d.h., wenn sie die Form

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} \lambda_{i,j}$$

hat, wobei $n, m_1, \ldots, m_n \geq 0$ sind und $\lambda_{i,j}$ für alle $i \in \{1, \ldots, n\}$ und $j \in \{1, \ldots, m_i\}$ jeweils ein Literal ist. Die Teilformeln

$$\delta_i \;:=\; \bigvee_{j=1}^{m_i} \lambda_{i,j}$$

werden *Klauseln* genannt.

Beispiel 1. Die aussagenlogischen Formeln φ_P, φ_H, φ_M, φ_Z, φ_S und φ_B und somit auch die aussagenlogische Formel φ aus dem vorhergehenden Abschnitt sind in konjunktiver Normalform.

Besonders wichtig für uns ist die Beobachtung, dass eine Klausel bereits durch eine Interpretation erfüllt wird, wenn diese ein einziges Literal der Klausel „wahr macht“. Umgekehrt wird eine konjunktive Normalform nur dann durch eine Interpretation erfüllt, wenn diese jede Klausel „wahr macht“.

Aus der Betrachtung der Interpretationen, die eine gegebene aussagenlogische Formel ψ (nicht) erfüllen, lässt sich leicht eine äquivalente Formel in konjunktiver Normalform konstruieren.

Satz 2. *Jede aussagenlogische Formel ist äquivalent zu einer konjunktiven Normalform.*

In den folgenden Abschnitten beschreiben wir Klauseln oft durch die Menge der in ihnen vorkommenden Literale und konjunktive Normalformen durch die Menge ihrer Klauseln. D.h., wir repräsentieren eine Klausel $\lambda_1 \vee \ldots \vee \lambda_m$ durch die Menge $\{\lambda_1, \ldots, \lambda_m\}$ und eine konjunktive Normalform $\delta_1 \wedge \ldots \wedge \delta_n$ aus den Klauseln $\delta_1, \ldots, \delta_n$ durch die *Klauselmenge* $\{\delta_1, \ldots, \delta_n\}$.

Beispiel 2. Die KNF $(\neg A \vee B) \wedge (C \vee \neg B \vee \neg A \vee D)$ entspricht der Klauselmenge $\{\{\neg A, B\}, \{C, \neg B, \neg A\}\}$.

Insbesondere entspricht eine leere Menge von Literalen der unerfüllbaren aussagenlogischen Formel $\mathbf{0}$ und eine leere Klauselmenge der allgemeingültigen Formel $\mathbf{1}$.

5. Das Resolutionsverfahren

Um herauszufinden, ob eine konjunktive Normalform (bzw. eine Klauselmenge) *unerfüllbar* ist, kann das *Resolutionsverfahren* angewendet werden. Die Grundidee des Resolutionsverfahrens ist es, in einem mehrschrittigen Verfahren zu zeigen, dass die Klauselmenge äquivalent ist zu einer Klauselmenge die eine leere Klausel $\mathbf{0}$ (bzw. $\emptyset$) enthält und somit unerfüllbar ist.

Im Folgenden bezeichnen wir mit $\overline{\lambda}$ das *Negat* eines Literals λ, d.h. die aussagenlogische Formel

$$\overline{\lambda} := \begin{cases} \neg X, & \text{wenn } \lambda = X \text{ für ein } X \in \mathrm{AS}, \\ X, & \text{wenn } \lambda = \neg X \text{ für ein } X \in \mathrm{AS}. \end{cases}$$

Definition 1. Seien δ_1 und δ_2 Klauseln. Gibt es ein Literal λ, so dass $\lambda \in \delta_1$ und $\overline{\lambda} \in \delta_2$, dann ist

$$\delta \;=\; (\delta_1 \setminus \{\lambda\}) \cup (\delta_2 \setminus \{\overline{\lambda}\})$$

eine *Resolvente* von δ_1 und δ_2.

Lemma 1. *Sei Γ eine Klauselmenge. Seien $\delta_1, \delta_2 \in \Gamma$ und sei δ eine Resolvente von δ_1 und δ_2. Dann sind die Klauselmengen Γ und $\Gamma \cup \{\delta\}$ äquivalent.*

Das Hinzufügen von Resolventen zu einer Klauselmenge ändert also nichts an ihrer (Un-)erfüllbarkeit.

Definition 2. Sei Γ eine Klauselmenge. Eine *Resolutionsableitung* einer Klausel δ aus Γ ist ein Tupel $(\delta_1, \ldots, \delta_\ell)$ von $\ell \geq 1$ Klauseln, in dem $\delta_\ell = \delta$, und für alle $i \in \{1, \ldots, \ell\}$ ist entweder

$$\delta_i \in \Gamma, \qquad \text{oder} \qquad \text{es gibt } j, k \in \{1, \ldots, i-1\}, \text{ so dass } \delta_i \text{ eine Resolvente von } \delta_j \text{ und } \delta_k \text{ ist.}$$

Eine *Resolutionswiderlegung* von Γ ist eine Resolutionsableitung der leeren Klausel $\emptyset$ aus Γ.

Beispiel 3. Sei $\Gamma = \{\{P, \neg S\}, \{S\}, \{S, R\}, \{\neg S, \neg P\}\}$. Das Tupel $(\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ mit

$\delta_1 := \{P, \neg S\}$, $\delta_2 := \{\neg P, \neg S\}$ und $\delta_4 := \{S\}$ *(aus Γ)*

$\delta_3 := \{\neg S\}$ *(Resolvente aus δ_1 und δ_2)*

$\delta_5 := \emptyset$ *(Resolvente aus δ_3 und δ_4)*

ist eine Resolutionswiderlegung von Γ.

Der folgende Satz besagt, dass das Resolutionsverfahren korrekt und vollständig ist.

Satz 3. *Eine Klauselmenge hat genau dann eine Resolutionswiderlegung, wenn sie unerfüllbar ist.*

Der folgende *Satz von Haken (1985)* zeigt jedoch, dass es unerfüllbare Klauselmengen gibt, deren Resolutionswiderlegungen zwangsläufig sehr lang werden. Die *Größe einer Klauselmenge* ist die Summe der Anzahl der Literale in ihren Klauseln.

Satz 4. *Es gibt Konstanten $c, d > 0$ und eine Folge $(\Gamma_n)_{n\geq 1}$ von* unerfüllbaren *Klauselmengen, so dass für jedes $n \geq 1$ gilt: Γ_n hat Größe $\leq n^c$, aber* jede *Resolutionswiderlegung von Γ_n hat mindestens Länge $2^{d\cdot n}$.*

6. Der DPLL-Algorithmus

Der DPLL-Algorithmus ist ein Algorithmus, der das Erfüllbarkeitsproblem der Aussagenlogik entscheidet. Er basiert – ähnlich wie der Wahrheitstafel-Algorithmus – auf einer systematischen Suche nach erfüllenden Interpretationen für eine aussagenlogische Formel φ. Der DPLL-Algorithmus ist für sehr viele aussagenlogische Formeln effizienter als der Wahrheitstafel-Algorithmus, da er Vereinfachungsheuristiken nutzt um die Suche abzukürzen. Diese Vereinfachungsheuristiken nutzen Ansätze aus dem Resolutionsverfahren.

Algorithmus DPLL(Γ):

> Wende Vereinfachungsheuristiken auf Γ an.
>
> Ist $\Gamma = \emptyset$, so antworte mit „erfüllbar“.
>
> Ist $\emptyset \in \Gamma$, so antworte mit „unerfüllbar“.
>
> Wähle ein Literal λ, das in einer Klausel von Γ vorkommt.
>
> Antwortet der rekursive Aufruf DPLL($\Gamma \cup \{\{\lambda\}\}$) mit „erfüllbar“, so antworte auch mit „erfüllbar“.
>
> Antwortet der rekursive Aufruf DPLL($\Gamma \cup \{\{\overline{\lambda}\}\}$) mit „erfüllbar“, so antworte auch mit „erfüllbar“.
>
> Antworte mit „unerfüllbar“.

Wir wissen bereits, dass die leere Klauselmenge erfüllbar ist, und dass eine leere Klausel eine Klauselmenge unerfüllbar macht.

Im Folgenden nennen wir eine *Einserklausel* eine Klausel, die nur aus genau einem Literal besteht. Für das Verständnis des DPLL-Algorithmus ist es wichtig zu sehen, dass eine Klauselmenge Γ, die eine Einserklausel $\{\lambda\}$ enthält, nur von Interpretationen erfüllt werden kann, die die Einserklausel „wahr machen“. Ist also λ ein Aussagensymbol X, dann wird Γ nur von Interpretationen $\mathcal{I}\colon \mathrm{AS} \to \{0,1\}$ mit $\mathcal{I}(X) = 1$ erfüllt. Ist $\lambda = \neg X$, so ist $\mathcal{I}(X) = 0$ in allen Modellen $\mathcal{I}\colon \mathrm{AS} \to \{0,1\}$ von Γ.

Die Erweiterung der Klauselmenge Γ um die Einserklausel $\{\lambda\}$ beziehungsweise $\{\overline{\lambda}\}$ in den rekursiven Aufrufen des DPLL-Algorithmus fixiert also eine Interpretation für das Aussagensymbol von λ.

Kann keine dieser Interpretationen zu einem Modell der Klauselmenge ergänzt werden, d.h., geben beide rekursiven Aufrufe „unerfüllbar“ aus, so führt der Algorithmus ein *Backtracking* aus. D.h., er kehrt in der Folge rekursiver Aufrufe zu dem letzten Punkt zurück, an dem der jeweilige rekursive Aufruf DPLL($\Gamma \cup \{\{\overline{\lambda}\}\}$) noch nicht beschritten wurde.

Sind keine solchen Alternativen mehr vorhanden, so kann es kein Modell für die Klauselmenge geben.

Die Vereinfachungsheuristiken *Unit Propagation* und *Pure Literal Rule* vereinfachen Γ wie folgt:

Unit Propagation *Solange Γ eine Einserklausel $\{\lambda\}$ enthält, entferne alle Klauseln aus Γ, die das Literal λ enthalten, und entferne $\overline{\lambda}$ aus allen übrigen Klauseln.*

Unit Propagation nutzt die Beobachtung, dass jedes Modell von Γ das Literal λ „wahr machen“ muss.

Pure Literal Rule *Solange Γ ein „pure“ Literal λ enthält, d.h., ein Literal dessen Negat $\overline{\lambda}$ in Γ nicht vorkommt, entferne alle Klauseln die λ enthalten aus Γ.*

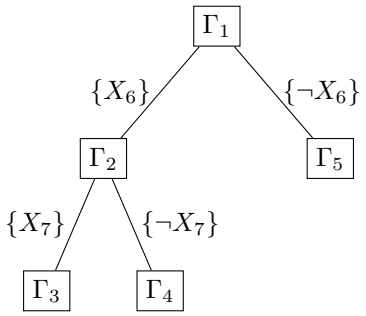
Die Pure Literal Rule basiert auf der Beobachtung, dass ein „pure“ Literal „wahr gemacht“ werden kann, ohne dass dadurch eine Klausel von Γ nicht mehr erfüllt wird.

Beispiel 4. Wir führen den DPLL-Algorithmus für die Klauselmenge

$$\begin{aligned}\Gamma \;:=\; \big\{ &\{X_1, \neg X_5, \neg X_6, X_7\}, \{\neg X_1, X_2, \neg X_5\}, \{\neg X_1, \neg X_2, \neg X_3, \neg X_5, \neg X_6\}, \{X_1, X_2, \neg X_4, X_7\},\\ &\{\neg X_4, \neg X_6, \neg X_7\}, \{X_3, \neg X_5, X_7\}, \{X_3, \neg X_4, \neg X_5\}, \{X_5, \neg X_6\}, \{X_5, X_4, \neg X_8\},\\ &\{X_1, X_3, X_5, X_6, X_7\}, \{\neg X_7, X_8\}, \{\neg X_6, \neg X_7, \neg X_8\}\big\}\end{aligned}$$

aus.

Der nebenstehende Baum stellt die rekursiven Aufrufe des DPLL-Algorithmus dar. Die Kanten sind mit den jeweils hinzugefügten Einserklauseln beschriftet. Die Knoten sind mit der Klauselmenge Γ_i, die im i-ten Schritt des Algorithmus *durch Anwendung der Vereinfachungsheuristiken entsteht*, beschriftet.



Die Klauselmenge Γ_1 ist identisch zu Γ, da die Vereinfachungsregeln nicht auf Γ angewendet werden können. Da keine der beiden Abbruchbedingungen erfüllt ist, wird das Literal X_6 (willkürlich) gewählt und der Algorithmus auf die Klauselmenge $\Gamma \cup \{\{X_6\}\}$ rekursiv angewendet.

Die Klauselmenge Γ_2 entsteht aus $\Gamma \cup \{\{X_6\}\}$ durch die Anwendung der Vereinfachungsheuristiken:

- Unit Propagation mit der Einserklausel $\{X_6\}$ liefert die Klauselmenge

$$\begin{aligned}\big\{ &\{X_1, \neg X_5, X_7\}, \{\neg X_1, X_2, \neg X_5\}, \{\neg X_1, \neg X_2, \neg X_3, \neg X_5\}, \{X_1, X_2, \neg X_4, X_7\}, \{\neg X_4, \neg X_7\},\\ &\{X_3, \neg X_5, X_7\}, \{X_3, \neg X_4, \neg X_5\}, \{X_5\}, \{X_5, X_4, \neg X_8\}, \{\neg X_7, X_8\}, \{\neg X_7, \neg X_8\}\big\}.\end{aligned}$$

- Unit Propagation mit der Einserklausel $\{X_5\}$ liefert die Klauselmenge

$$\begin{aligned}\big\{ &\{X_1, X_7\}, \{\neg X_1, X_2\}, \{\neg X_1, \neg X_2, \neg X_3\}, \{X_1, X_2, \neg X_4, X_7\}, \{\neg X_4, \neg X_7\},\\ &\{X_3, X_7\}, \{X_3, \neg X_4\}, \{\neg X_7, X_8\}, \{\neg X_7, \neg X_8\}\big\}.\end{aligned}$$

- Pure Literal Rule mit dem Literal $\neg X_4$ liefert die Klauselmenge

$$\begin{aligned}\Gamma_2 \;:=\; \big\{ &\{X_1, X_7\}, \{\neg X_1, X_2\}, \{\neg X_1, \neg X_2, \neg X_3\},\\ &\{X_3, X_7\}, \{\neg X_7, X_8\}, \{\neg X_7, \neg X_8\}\big\}.\end{aligned}$$

Die Klauselmenge Γ_2 erfüllt keine der Abbruchbedingungen, so dass der DPLL-Algorithmus rekursiv auf die Klauselmenge $\Gamma_2 \cup \{\{X_7\}\}$ angewendet wird. Diese Klauselmenge wird vereinfacht zu

$$\Gamma_3 \;:=\; \big\{\{\neg X_1, X_2\}, \{\neg X_1, \neg X_2, \neg X_3\}, \emptyset\big\}.$$

Da die leere Klausel in Γ_3 vorkommt, ist Γ_3 unerfüllbar. Das Backtracking wendet den DPLL-Algorithmus rekursiv auf die Klauselmenge $\Gamma_2 \cup \{\{\neg X_7\}\}$, die zur unerfüllbaren Klauselmenge $\{\emptyset\}$ vereinfacht wird. Das Backtracking wendet den DPLL-Algorithmus nun rekursiv auf die Klauselmenge $\Gamma_1 \cup \{\{\neg X_6\}\}$ an. Die Vereinfachungsheuristiken vereinfachen diese Klauselmenge zur leeren Klauselmenge. Da diese erfüllbar ist, wird schließlich „erfüllbar“ ausgegeben.

Es ist leicht zu sehen, dass aus der Anwendung der Vereinfachungsheuristiken im Falle einer erfüllbaren Klauselmenge auch eine erfüllende Interpretation rekonstruiert werden kann.

Beispiel 5. Die Vereinfachung der Klauselmenge $\Gamma_1 \cup \{\{\neg X_6\}\}$ zu $\emptyset$ in Beispiel 4 liefert ein Modell $\mathcal{I}\colon \mathrm{AS} \to \{0,1\}$ für Γ mit $\mathcal{I}(X_2) = \mathcal{I}(X_3) = \mathcal{I}(X_5) = 1$ und $\mathcal{I}(X_6) = \mathcal{I}(X_7) = 0$.

7. Ein Sudoku-Löser

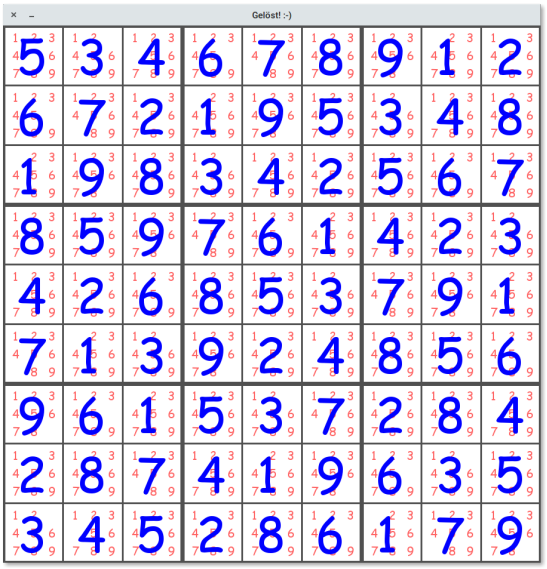
Unter Nutzung des DPLL-Algorithmus wurde ein Programm (siehe Abbildung) implementiert, dass bei Eingabe eines Sudokus eine Lösung sucht und, wenn vorhanden, ausgibt. Das Programm führt bei Eingabe eines Sudokus S folgende Schritte aus:

1. Konstruiere eine aussagenlogische Formel φ in konjunktiver Normalform, so dass gilt:

   $$\varphi \text{ ist erfüllbar} \iff S \text{ ist lösbar.}$$

2. Rufe den DPLL-Algorithmus für die zu φ korrespondierende Klauselmenge Γ auf.

3. Ist Γ unerfüllbar, so gebe aus, dass das Sudoku S keine Lösung hat. Rekonstruiere anderenfalls aus dem vom DPLL-Algorithmus gefundenen Modell von φ eine Lösung für das Sudoku S.



8. Erfüllbarkeitsäquivalenz

Im letzten Abschnitt haben wir den DPLL-Algorithmus auf Sudokus angewendet. Dies wurde uns dadurch erleichtert, dass die, die Lösungen eines Sudokus beschreibende Formel φ, von vornherein in konjunktiver Normalform vorliegt. Wie können wir vorgehen, wenn dies nicht der Fall ist?

Wir wissen bereits, dass wir zu jeder aussagenlogischen Formel eine äquivalente konjunktive Normalform finden können. Der folgende Satz zeigt uns jedoch, dass diese sehr viel größer werden können als die ursprüngliche Formel. Die *Größe einer aussagenlogischen Formel* ist dabei ihre Länge als Zeichenkette.

Satz 5. *Für jedes $n \geq 1$ sei*

$$\varphi_n \;:=\; \bigvee_{i=1}^{n} (X_i \wedge \neg Y_i).$$

Es gibt eine Zahl $c \geq 1$, so dass für jedes $n \geq 1$ gilt: Die Formel φ_n hat Größe $\leq c \cdot n$, aber jede zu φ_n äquivalente Formel in konjunktiver Normalform hat mindestens 2^n Klauseln.

Wenn es uns jedoch nur darum geht zu entscheiden, ob eine gegebene aussagenlogische Formel *erfüllbar* ist, dann benötigen wir keine zu ihr äquivalente konjunktive Normalform. Es genügt eine konjunktive Normalform, die zu ihr *erfüllbarkeitsäquivalent ist*.

Definition 3. Zwei aussagenlogische Formeln φ und ψ sind *erfüllbarkeitsäquivalent*, wenn φ genau dann erfüllbar ist, wenn auch ψ erfüllbar ist.

Im Folgenden werden wir an einem Beispiel das Tseitin-Verfahren (Grigori Tseitin, 1970) kennenlernen, mit dem zu einer beliebigen aussagenlogischen Formel eine erfüllbarkeitsäquivalente konjunktive Normalform konstruiert werden kann, die nur linear mit der ursprünglichen Formel wächst.

Beispiel 6.

1. Zuerst werden alle Teilformeln einer aussagenlogischen Formel φ, die keine Literale sind, durchnummeriert, d.h., in der Reihenfolge ihres Vorkommens in φ mit $\psi_1, \psi_2, \dots$ benannt.

$$\underbrace{(P \rightarrow \neg Q)}_{\psi_1} \vee \underbrace{\big(\underbrace{\neg \underbrace{(P \wedge Q)}_{\psi_4}}_{\psi_3} \wedge R\big)}_{\psi_2}$$

   Für jede Teilformel ψ sei X_ψ ein neues Aussagensymbol mit der Bedeutung „die Teilformel ψ ist wahr“.

2. Wir konstruieren die zu φ erfüllbarkeitsäquivalente Formel

$$\begin{aligned}\varphi' \;:=\;& X_\varphi \\ &\wedge\; \big(X_\varphi \;\leftrightarrow\; (X_{\psi_1} \vee X_{\psi_2})\big) \\ &\wedge\; \big(X_{\psi_1} \;\leftrightarrow\; (P \rightarrow \neg Q)\big) \\ &\wedge\; \big(X_{\psi_2} \;\leftrightarrow\; (X_{\psi_3} \wedge R)\big) \\ &\wedge\; \big(X_{\psi_3} \;\leftrightarrow\; \neg X_{\psi_4}\big) \\ &\wedge\; \big(X_{\psi_4} \;\leftrightarrow\; (P \wedge Q)\big).\end{aligned}$$

3. Die Teilformeln in den einzelnen Zeilen von φ' werden abschließend einzeln in äquivalente konjunktive Normalformen umgewandelt. Deren Größe ist unabhängig von der Größe von φ, da in jeder Zeile von φ' höchstens 3 verschiedene Aussagensymbole vorkommen. Wir erhalten auf diese Weise die konjunktive Normalform

$$\begin{aligned}\varphi_K \;:=\;& X_\varphi \\ &\wedge\; (\neg X_\varphi \vee X_{\psi_1} \vee X_{\psi_2}) \;\wedge\; (X_\varphi \vee \neg X_{\psi_1}) \;\wedge\; (X_\varphi \vee \neg X_{\psi_2}) \\ &\wedge\; (\neg X_{\psi_1} \vee \neg P \vee \neg Q) \;\wedge\; (P \vee X_{\psi_1}) \;\wedge\; (Q \vee X_{\psi_1}) \\ &\wedge\; (\neg X_{\psi_2} \vee X_{\psi_3}) \;\wedge\; (\neg X_{\psi_2} \vee R) \;\wedge\; (\neg X_{\psi_3} \vee \neg R \vee X_{\psi_2}) \\ &\wedge\; (\neg X_{\psi_3} \vee \neg X_{\psi_4}) \;\wedge\; (X_{\psi_4} \vee X_{\psi_3}) \\ &\wedge\; (\neg X_{\psi_4} \vee P) \;\wedge\; (\neg X_{\psi_4} \vee Q) \;\wedge\; (\neg P \vee \neg Q \vee X_{\psi_4}).\end{aligned}$$

   Offensichtlich ist φ_K äquivalent zu φ' und somit erfüllbarkeitsäquivalent zu φ. Zudem wächst die Größe von φ_K nur linear mit der Anzahl der Teilformeln von φ und ist insbesondere unabhängig von der Anzahl der Aussagensymbole in φ.

9. Danksagung

Wir bedanken uns herzlich für die großzügige Unterstützung durch Prof. Dr. Nicole Schweikardt. Die während der Sommerschule in unserer Gruppe ausgearbeiteten Inhalte basieren auf Ausschnitten ihrer Vorlesung „Logik in der Informatik“ an der Humboldt-Universität zu Berlin.