

4.2 Hornformeln

Eine wichtige Formelklasse bilden die sogenannten Hornformeln, denen in der Logikprogrammierung eine zentrale Rolle zukommt, und für die ein sehr einfacher und effizienter Erfüllbarkeitstest möglich ist. Hornformeln sind spezielle KNF-Formeln, nämlich:

Definition 4.15 (Hornformel, Hornklausel). Eine *Hornklausel* ist eine Klausel, in der höchstens ein positives Literal vorkommt (aber beliebig viele negative Literale). Eine KNF-Formel heißt *Hornformel*, falls alle ihre Klauseln Hornklauseln sind. ■

Die KNF-Formeln $(x \vee \neg y) \wedge \neg z$, $(\neg x \vee \neg y) \wedge \neg z$, $(\neg x \vee y) \wedge z$ sind Hornformeln. Auch $x \wedge y$ ist eine Hornformel, da sie aus zwei Einheitsklauseln besteht. Die KNF-Formel $x \vee y$ (die nur aus einer Klausel besteht) ist dagegen keine Hornformel, da sie eine Klausel mit zwei positiven Literalen enthält. Es gibt drei Arten von Hornformeln:

- negative Klauseln, d.h. Klauseln, welche nur aus negativen Literalen bestehen. Im Kontext der Logikprogrammierung werden sie meist *Zielklauseln* genannt. Ein Sonderfall hiervon ist die leere Klausel.
- positive Einheitsklauseln, auch *Fakten* genannt.
- Hornformeln mit (genau) einem positiven und wenigstens einem negativen Literal. Diese werden auch *Regeln* genannt.

Anlehnend an die Syntax von Logikprogrammiersprachen wie PROLOG oder DATALOG sind für Hornformeln Implikationsschreibweisen “*Prämisse* \rightarrow *Folgerung*” gebräuchlich. So steht $x_1 \wedge \dots \wedge x_m \rightarrow y$ für die Hornklausel $\neg x_1 \vee \dots \vee \neg x_m \vee y$. Die Prämisse $x_1 \wedge \dots \wedge x_m$ wird der *Rumpf* und das positive Literal y der *Kopf* der Regel genannt. Positive Einheitsklauseln (Fakten) werden oftmals in der Form $true \rightarrow x$ geschrieben, Zielklauseln in der Form $x_1 \wedge \dots \wedge x_m \rightarrow false$. Beachte, dass

$$true \rightarrow x = false \vee x \equiv x \quad \text{und} \quad x_1 \wedge \dots \wedge x_m \rightarrow false \equiv \neg x_1 \vee \dots \vee \neg x_m.$$

Beispiel 4.16 (Affen-Bananen-Problem). Die vier Formeln $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ der Formelmenge \mathfrak{F} aus Beispiel 4.8 zum Affen-Bananen-Problem (siehe Seite 124) sind Hornklauseln. Die letzte Klausel $\alpha_4 = banana$ ist ein Faktum. Die anderen drei Formeln sind Regeln. In den ersten beiden Regeln

$$\begin{aligned} \alpha_1 &= down \rightarrow below \equiv \neg down \vee below \\ \alpha_2 &= down \rightarrow up \quad \equiv \neg down \vee up \end{aligned}$$

besteht der Rumpf jeweils nur aus einem Literal (nämlich *down*). Die Atome *up* bzw. *below* sind die Köpfe der beiden Regeln. Die dritte Regel

$$\alpha_3 = up \wedge below \rightarrow banana \equiv \neg up \vee \neg below \vee banana$$

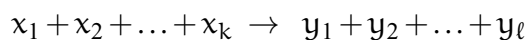
hat zwei Literale im Rumpf und die Banane im Kopf. Wie bereits in Beispiel 4.8 nachgewiesen wurde, gilt $\alpha \Vdash banana$, wobei α diejenige Hornformel ist, die sich durch Konjunktion der vier

Hornklauseln ergibt, also $\alpha = \alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge \alpha_4$. Äquivalent zur Folgerbarkeit von *banana* aus α ist die Unerfüllbarkeit der Hornformel

$$\alpha \wedge \neg \textit{banana} \equiv \alpha \wedge (\textit{banana} \rightarrow \textit{false}),$$

die sich durch die Hinzunahme der Negation der Frage als Zielklausel ergibt. ■

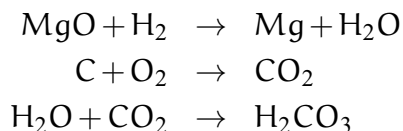
Beispiel 4.17 (“Wissensbasiertes System” für ein Chemielabor). Gegeben ist eine Datenbank, in der Informationen über die vorhandenen chemischen Stoffe und durchführbaren chemischen Reaktionen



eines Chemielabors verwaltet werden.¹⁵ Gefragt ist, ob ein gewisser chemischer Stoff z (über 0 oder mehr chemische Reaktionen) erzeugt werden kann. Jeden Stoff fassen wir als Aussagensymbol auf. Das Vorhandensein von vorhandenen Stoffe und durchführbaren Reaktionen wird durch eine Hornformel α dargestellt. Die vorhandenen Stoffe stellen wir durch Fakten dar. Jede der durchführbaren Reaktionen $x_1 + x_2 + \dots + x_k \rightarrow y_1 + y_2 + \dots + y_\ell$ zerlegen wir in ℓ Regeln

$$\begin{array}{rcl} x_1 \wedge x_2 \wedge \dots \wedge x_k & \rightarrow & y_1 \\ x_1 \wedge x_2 \wedge \dots \wedge x_k & \rightarrow & y_2 \\ & \vdots & \vdots \\ x_1 \wedge x_2 \wedge \dots \wedge x_k & \rightarrow & y_\ell \end{array}$$

welche die Herstellbarkeit von jedem der Stoffe y_i unter der Voraussetzung, dass die Stoffe x_1, x_2, \dots, x_k verfügbar sind, beschreiben. Die Frage, ob ein Stoff z in dem Labor erzeugt werden kann, ist gleichbedeutend mit der Frage, ob $\alpha \models z$ gilt (Begründung siehe unten). Zur Beantwortung der Frage ist also die Unerfüllbarkeit der Formel $\alpha \wedge \neg z$ zu prüfen. Wir illustrieren diese Aussage an einem Beispiel.



seien die durchführbaren chemischen Reaktionen, wobei die Grundstoffe MgO, H₂, O₂ und C verfügbar sind. Wir fragen, ob $z = \text{H}_2\text{CO}_3$ (Kohlensäure) herstellbar ist. Hierzu ist die Unerfüllbarkeit folgender Hornformel nachzuweisen:

$$\begin{array}{rcl} \text{MgO} \wedge \text{H}_2 \wedge \text{O}_2 \wedge \text{C} & \wedge & \\ (\text{H}_2 \wedge \text{MgO} \rightarrow \text{Mg}) & \wedge & \\ (\text{H}_2 \wedge \text{MgO} \rightarrow \text{H}_2\text{O}) & \wedge & \\ (\text{C} \wedge \text{O}_2 \rightarrow \text{CO}_2) & \wedge & \\ (\text{H}_2\text{O} \wedge \text{CO}_2 \rightarrow \text{H}_2\text{CO}_3) & \wedge & \\ (\text{H}_2\text{CO}_3 \rightarrow \textit{false}). & & \end{array}$$

¹⁵Die Durchführbarkeit einer Reaktion ist so zu verstehen, dass das Labor über die Ausstattung verfügt, um die betreffende Reaktion durchzuführen, sofern die Stoffe der linken Seite vorliegen (initial vorhanden sind oder durch vorangegangene Reaktionen erzeugt wurden).

Wir erläutern nun, warum die Herstellbarkeit von Stoff z gleichbedeutend mit $\alpha \Vdash z$ ist. Zunächst nehmen wir an, dass $\alpha \Vdash z$. Wir betrachten nun die Belegung I , welche genau denjenigen Aussagensymbolen (Stoffen) den Wahrheitswert 1 zuweist, die in dem vorliegenden Labor über 0 oder mehrere Schritte herstellbar sind. Dann gilt $\alpha^I = 1$ und somit $z^I = 1$. Also ist Stoff z herstellbar. Wir setzen nun umgekehrt die Herstellbarkeit von z voraus und zeigen, dass z eine logische Folgerung von α ist. Hierzu zeigen wir durch Induktion nach m , dass jeder Stoff, der in dem betreffenden Labor durch m Reaktionen hergestellt werden kann, eine logische Folgerung aus α ist. Im Induktionsanfang $m = 0$ betrachten wir Stoffe, die durch 0 Reaktionen hergestellt werden können. Dies sind initial vorhandene Stoffe, die als Fakten in α vorkommen. Diese Stoffe sind offenbar logische Folgerungen aus α . Nun zum Induktionsschritt $m-1 \implies m$, wobei $m \geq 1$. Sei z ein Stoff, der mittels m Reaktionen in dem betreffenden Labor hergestellt werden kann. Ferner sei

$$x_1 + \dots + x_k \rightarrow \dots + z + \dots$$

die letzte dieser m Reaktionen. Dann ist jeder der Stoffe x_j mit höchstens $m-1$ Reaktionen in dem Labor herstellbar. Nach Induktionsvoraussetzung gilt $\alpha \Vdash x_j$ für $j = 1, \dots, k$. Da $x_1 \wedge \dots \wedge x_k \rightarrow z$ eine Klausel von α ist, gilt $\alpha \Vdash z$. ■

Markierungsalgorithmus für HORN-SAT. Für Hornformeln ist das Erfüllbarkeitsproblem HORN-SAT (als Kurzform für die englische Bezeichnung “satisfiability problem for Horn formulas”) recht leicht zu beantworten. Dieses hat als Eingabe eine Hornformel α . Die Aufgabe ist zu entscheiden, ob α erfüllbar ist. Hierzu kann ein *Markierungsalgorithmus* eingesetzt werden, der sukzessive alle Aussagensymbole markiert, die unter einer erfüllenden Belegung (falls es eine solche gibt) zwingend mit 1 bewertet sein müssen. In der Initialisierungsphase werden alle Aussagensymbole, die als Fakten in α vorkommen, markiert. Der Algorithmus markiert nun solange wie möglich weitere Aussagensymbole, die als Kopf y in einer Regel $x_1 \wedge \dots \wedge x_m \rightarrow y$ erscheinen, wobei x_1, \dots, x_m bereits markiert sind, oder bricht mit der Antwort “nein” ab, sofern es eine Zielklausel $x_1 \wedge \dots \wedge x_m \rightarrow \text{false}$ gibt, für die x_1, \dots, x_m markiert sind. Siehe Algorithmus 5.

Die Terminierung des Verfahrens ergibt sich aus der Beobachtung, dass die Anzahl an Markierungsschritten, die in der Initialisierungsphase oder Markierungsschritte innerhalb der Schleife durchgeführt werden, durch die Anzahl an Aussagensymbolen, die in α vorkommen, nach oben beschränkt ist. Wir zeigen nun die Korrektheit der ausgegebenen Antworten.

Satz 4.18 (Korrektheit des Markierungsalgorithmus). *Ist α eine Hornformel, so terminiert der beschriebene Markierungsalgorithmus für HORN-SAT (Algorithmus 5) mit der korrekten Antwort “nein” im Falle der Unerfüllbarkeit von α und “ja” im Falle der Erfüllbarkeit von α .*

Ist α erfüllbar, so ist diejenige Belegung I , die genau den markierten Aussagensymbolen den Wahrheitswert 1 zuordnet, das kleinste Modell für α . D.h., es gilt $\alpha^I = 1$ und für jedes Modell J für α gilt $x^I \leq x^J$ für alle Aussagensymbole x , die in α vorkommen.

Beweis. Zunächst beweisen wir folgende Hilfsaussage, die als Schleifeninvariante angesehen werden kann:

(*) Ist J ein Modell für α , so ist $y^J = 1$ für alle markierten Aussagensymbole y .

Algorithmus 5 Markierungsalgorithmus für HORN-SAT

(* Eingabe: Hornformel α *)

(* Aufgabe: prüft, ob α erfüllbar ist *)

Markiere alle Aussagensymbole x , die in α als Fakten vorkommen.

REPEAT

IF es gibt eine Zielklausel $x_1 \wedge \dots \wedge x_m \rightarrow false$ in α , so dass x_1, \dots, x_m markiert sind
THEN gib “nein, α ist unerfüllbar” aus und halte an

FI

IF es gibt eine Regel $x_1 \wedge \dots \wedge x_m \rightarrow y$ in α , so dass x_1, \dots, x_m markiert sind
und y nicht markiert ist,

THEN markiere y

FI

UNTIL keine Veränderungen in der letzten Iteration

gib “ja, α ist erfüllbar” aus

Der Nachweis von (*) erfolgt durch Induktion nach der Anzahl an Markierungsschritten in der Initialisierungsphase oder innerhalb der REPEAT-Schleife. Der Induktionsanfang bezieht sich auf die Initialisierungsphase. In dieser werden genau die Aussagensymbole x , die als Einheitsklauseln in α vorkommen, markiert. Unter jedem Modell J für $\alpha = \dots \wedge x \wedge \dots$ müssen diese den Wahrheitswert 1 haben. Im Induktionsschritt nehmen wir nun an, dass y innerhalb der REPEAT-Schleife markiert wurde. Dann gibt es eine Regel $\kappa = x_1 \wedge \dots \wedge x_m \rightarrow y$ von α , so dass die Aussagensymbole x_1, \dots, x_m vor y markiert wurden. Die Induktionsvoraussetzung liefert $x_1^J = \dots = x_m^J = 1$ für jede erfüllende Belegung J für α . Wegen $\alpha^J = 1$ gilt $\kappa^J = 1$ und somit auch $y^J = 1$. Damit ist (*) bewiesen.

1. Wir setzen zunächst voraus, dass Algorithmus 5 “nein” ausgibt und zeigen, dass α unerfüllbar ist. Die Antwort “nein” ist nur möglich, wenn es eine Zielklausel $x_1 \wedge \dots \wedge x_n \rightarrow false$ von α gibt, so dass x_1, \dots, x_n markiert sind. Wir nehmen nun an, dass α eine erfüllende Belegung J besitzt und führen diese Annahme zu einem Widerspruch. Aufgrund von Aussage (*) müsste dann $x_1^J = \dots = x_n^J = 1$ gelten. Dann aber wäre

$$(x_1 \wedge \dots \wedge x_n \rightarrow false)^J = (\neg x_1 \vee \dots \vee \neg x_n)^J = 0$$

und somit $\alpha^J = 0$. Widerspruch zur Annahme, dass J erfüllend für α ist.

2. Wir nehmen nun an, dass der Algorithmus mit der Antwort “ja” terminiert und zeigen, dass die Belegung I , welche durch

$$x^I \stackrel{\text{def}}{=} \begin{cases} 1 & : \text{ falls } x \text{ nach der Terminierung markiert ist} \\ 0 & : \text{ sonst} \end{cases}$$

definiert ist, erfüllend für α ist. Sei κ eine Klausel von α . Wir zeigen, dass $\kappa^I = 1$ ist.

1. **Fall:** κ ist ein Faktum (positive Einheitsklausel); etwa $\kappa = x$. Dann wird x in der Initialisierungsphase markiert. Daher gilt $\kappa^I = x^I = 1$.

2. Fall: κ ist eine Regel der Form $x_1 \wedge \dots \wedge x_m \rightarrow y$.

- Falls $x_1^I = \dots = x_m^I = 1$, so ist $y^I = 1$ und somit $\kappa^I = 1$.
- Falls $x_i^I = 0$ für ein $i \in \{1, \dots, m\}$, so ist $\kappa^I = 1$.

3. Fall: κ ist eine Zielklausel der Form $x_1 \wedge \dots \wedge x_m \rightarrow false$. Der Fall $x_1^I = \dots = x_m^I = 1$ ist nicht möglich, da Algorithmus 5 sonst mit der Antwort “nein” terminiert hätte. Also gilt $x_i^I = 0$ für ein $i \in \{1, \dots, m\}$ und somit $\kappa^I = 1$.

Da alle Klauseln von κ unter I den Wahrheitswert 1 haben, ist $\alpha^I = 1$. Aus Hilfsaussage (*) folgt, dass $x^I \leq x^J$ für jede erfüllende Belegung J für α . Also ist I das kleinste Modell für α .

□

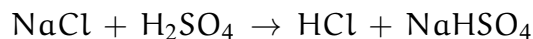
Da Algorithmus 5 für eine Hornformel mit n Aussagensymbolen höchstens n Iterationen (Markierungsschritte) ausführt, ist klar, dass eine Implementierung möglich ist, deren Laufzeit polynomiell in $|\alpha|$ beschränkt ist.

Beispiel 4.19 (Markierungsalgorithmus für HORN-SAT). Wir betrachten nochmals das Chemielabor aus Beispiel 4.17 auf Seite 140. Wie bereits erwähnt, ist die Herstellbarkeit von Kohlensäure H_2CO_3 gleichbedeutend mit der Unerfüllbarkeit nachstehender Hornformel:

$$\begin{aligned} \beta = & MgO \wedge H_2 \wedge O_2 \wedge C \wedge \\ & (H_2 \wedge MgO \rightarrow Mg) \wedge (H_2 \wedge MgO \rightarrow H_2O) \wedge \\ & (C \wedge O_2 \rightarrow CO_2) \wedge (H_2O \wedge CO_2 \rightarrow H_2CO_3) \wedge \\ & (H_2CO_3 \rightarrow false) \end{aligned}$$

Algorithmus 5 angewandt auf β beginnt mit der Markierung der vier Fakten MgO, H_2, O_2 und C (Initialisierungsphase). Dann sind die Rümpfe der ersten drei Regeln markiert, so dass nacheinander deren Köpfe (also die Aussagensymbole Mg, H_2O, CO_2) markiert werden (Markierungsschritte innerhalb der REPEAT-Schleife). Da nun auch der Rumpf der letzten Regel markiert ist, wird deren Kopf (das Aussagensymbol H_2CO_3) markiert. Damit ist der Rumpf der Zielklausel $H_2CO_3 \rightarrow false$ markiert, so dass Algorithmus mit der Antwort “nein” terminiert. Hornformel β ist also unerfüllbar und somit Stoff H_2CO_3 herstellbar.

Wir fügen nun zusätzlich das Faktum $NaCl$ und die Regeln ein, die für die Durchführbarkeit der Reaktion



stehen, die aus $NaCl$ (Natriumchlorid) und H_2SO_4 (Schwefelsäure) die beiden Stoffe HCl (Chlorwasserstoff) und $NaHSO_4$ (Natriumbisulfat) erzeugen. Die Herstellbarkeit des Stoffes $NaHSO_4$ ist nun gleichbedeutend mit der Unerfüllbarkeit folgender Hornformel:

$$\begin{aligned} \gamma = & MgO \wedge H_2 \wedge O_2 \wedge C \wedge NaCl \wedge \\ & (H_2 \wedge MgO \rightarrow Mg) \wedge (H_2 \wedge MgO \rightarrow H_2O) \wedge \\ & (C \wedge O_2 \rightarrow CO_2) \wedge (H_2O \wedge CO_2 \rightarrow H_2CO_3) \wedge \\ & (NaCl \wedge H_2SO_4 \rightarrow HCl) \wedge (NaCl \wedge H_2SO_4 \rightarrow NaHSO_4) \wedge \\ & (NaHSO_4 \rightarrow false) \end{aligned}$$

Der Markierungsalgorithmus für HORN-SAT angewandt auf γ verhält sich zunächst wie für β . In der Initialisierungsphase werden die fünf Fakten MgO , H_2 , O_2 , C und $NaCl$ markiert. Dann werden der Reihe nach die Stoffe Mg , H_2O , CO_2 und H_2CO_3 markiert. Danach hält Algorithmus 5 mit der Antwort “ja” an, da keine weiteren Markierungen und kein Abbruch möglich sind. Also ist γ erfüllbar. Unter den genannten Bedingungen ist also $NaHSO_4$ nicht herstellbar ist. Dieser Sachverhalt ist nicht verwunderlich, da der Stoff H_2SO_4 “vergessen” wurde. ■

Wir beenden diesen Abschnitt mit einigen einfachen Aussagen über Hornformeln. Zunächst stellen wir fest, dass Hornformeln *ohne* Zielklauseln niemals widersprüchlich sein können. Solche Hornformeln nennt man auch *definit*. Erst durch die Hinzunahme von Zielklauseln können unerfüllbare Formeln entstehen. Die Begründung hierfür ist sehr einfach. In definiten Hornformeln hat jede Klausel genau ein positives Literal. Betrachtet man nun die Interpretation, die alle Aussagensymbole mit dem Wahrheitswert 1 belegt, so erhält man offenbar eine erfüllende Belegung. Eine andere mögliche Begründung orientiert sich am Markierungsalgorithmus, der offenbar niemals die Antwort “nein” ausgeben kann, wenn keine Zielklauseln vorhanden sind. Die Regeln und Fakten eines Logikprogramms, die zu einer Hornformel ohne Zielklauseln zusammengefasst werden können, stellen also stets eine erfüllbare Formel dar.

Umgekehrt ist auch jede Hornformel, die keine Fakten und keine leere Klauseln enthält, erfüllbar. Die Begründung hierfür ist, dass die Belegung I, die allen Aussagensymbolen den Wert 0 zuweist, alle Klauseln wahr macht, da jede Klausel nun mindestens ein negatives Literal enthält. Auch hier ist eine alternative Argumentation mit dem Markierungsalgorithmus möglich: Wenn keine Fakten vorliegen, so wird kein Aussagensymbol markiert. Die Antwort “nein” könnte deshalb nur dann ausgegeben werden, wenn eine Zielklausel mit leerem Rumpf (also eine leere Klausel) vorliegt. Wir erhalten die Aussage des folgenden Lemmas:

Lemma 4.20 (Hinreichende Kriterien für die Erfüllbarkeit von Hornformeln).

- (a) *Jede definite Hornformel (d.h., Hornformel ohne Zielklauseln) ist erfüllbar.*
- (b) *Jede Hornformel ohne Fakten und leere Klauseln ist erfüllbar.*

Bemerkung 4.21 (Hornformeln sind nicht universell). Es gibt aussagenlogische Formeln, zu denen es keine äquivalenten Hornformeln gibt. Ein derartiges Beispiel ist die Formel $x \vee y$. Wir nehmen an, dass es eine zu $x \vee y$ äquivalente Hornformel α gibt. Dann haben $x \vee y$ und α dieselben Modelle. Da α eine erfüllbare Hornformel ist, haben α und somit auch $x \vee y$ ein *kleinstes* Modell. Dies steht jedoch im Widerspruch zur Beobachtung, dass $x \vee y$ kein kleinstes Modell besitzt, da $[x = 0, y = 0]$ kein Modell für $x \vee y$ ist und die “nächst größeren” Modelle $[x = 1, y = 0]$ und $[x = 0, y = 1]$ unvergleichbar sind. ■

Eine weitere Beobachtung ist, dass aus Hornformeln ohne Zielklauseln keine negativen Literale logisch gefolgert werden können. Diese Beobachtung ist für die Logikprogrammierung sehr bedeutsam, da sie einen vorsichtigen Umgang mit Fragen des Typs “gilt Aussage ... nicht?” erfordert.

Lemma 4.22 (Hornformeln ohne Zielklauseln lassen keine negativen Schlüsse zu!). *Ist α eine Hornformel, in der jede Klausel genau ein positives Literal hat, und x ein Aussagensymbol, so gilt $\alpha \not\models \neg x$.*

Beweis. Angenommen $\alpha \models \neg x$. Dann ist $\alpha \wedge \neg \neg x \equiv \alpha \wedge x$ unerfüllbar. Andererseits ist $\alpha \wedge x$ eine Hornformel, die keine Zielklauseln enthält, und somit erfüllbar (Teil (a) von Lemma 4.20). Widerspruch. \square

4.3 SAT-Beweiser

Neben dem Spezialfall von DNF- und Hornformeln gibt es weitere Klassen von aussagenlogischen Formeln, für die das Erfüllbarkeitsproblem algorithmisch effizient lösbar ist. Werden keine syntaktischen Einschränkungen für die Eingabeformel gemacht, so zählt das Erfüllbarkeitsproblem der Aussagenlogik (SAT für “satisfiability problem”) zu den algorithmisch schwierigen Problemen. Wir stellen hier nun einige Verfahren vor, wie man die Erfüllbarkeit für beliebige aussagenlogische Formeln und KNF-Formeln algorithmisch testen kann. Solche Verfahren werden auch SAT-Beweiser bzw. KNF-SAT-Beweiser genannt.

Naiver SAT-Beweiser. Im Folgenden sei α eine aussagenlogische Formel, deren Erfüllbarkeit zu prüfen ist, und x_1, \dots, x_n seien die in α vorkommenden Atome. Der naive Algorithmus für SAT berechnet die Wahrheitswerte α^I unter allen Belegungen I für $\{x_1, \dots, x_n\}$, und gibt “ja” zurück, genau dann, wenn $\alpha^I = 1$ für wenigstens eine der Belegungen. Diese “alle-Belegungen-ausprobieren”-Methode kann dadurch realisiert werden, indem man den *Entscheidungsbaum* für α mit einer Tiefensuche (Preorder) analysiert. Der Entscheidungsbaum ist lediglich eine anschauliche Darstellung aller Belegungen für x_1, \dots, x_n und deren Wahrheitswerten für α . Der Entscheidungsbaum für α ist ein vollständiger Binärbaum der Höhe n . Die beiden ausgehenden Kanten eines inneren Knotens der Tiefe i , wobei $0 \leq i < n$, stehen für die Belegungen $x_{i+1} = 0$ bzw. $x_{i+1} = 1$. Jedem inneren Knoten v der Tiefe i kann diejenige (Teil-)Belegung $[x_1 = b_1, \dots, x_i = b_i]$ für die ersten i Atome x_1, \dots, x_i zugeordnet werden, die sich aus den Beschriftungen der Kanten des Pfads von der Wurzel zu Knoten v ergibt. Jedem Pfad von der Wurzel zu einem Blatt (Knoten der Tiefe n) entspricht eine Belegung I für x_1, \dots, x_n . Das betreffende Blatt ist mit dem Wahrheitswert von α unter dieser Belegung I beschriftet. Ein Beispiel für einen Entscheidungsbaum ist in Abbildung 32 angegeben, wobei stets die Kante von einem inneren Knoten der Tiefe i zu dem linken Sohn für den Fall $x_{i+1} = 0$ steht und die Kante zu dem rechten Sohn für $x_{i+1} = 1$.

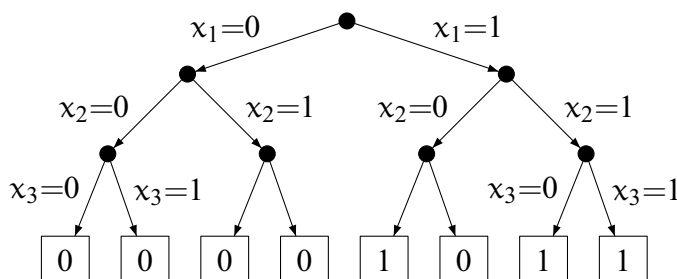


Abbildung 32: Entscheidungsbaum für $x_1 \wedge (x_2 \vee \neg x_3)$

Im Kontext des Erfüllbarkeitsproblems sind wir lediglich daran interessiert, ob der Entscheidungsbaum ein mit dem Wahrheitswert 1 beschriftetes Blatt enthält. Daher können wir die

Wahrheitswerte der Blätter bis hoch zur Wurzel propagieren, indem wir jeden inneren Knoten v mit 1 beschriften, sofern wenigstens einer seiner beiden Söhne mit 1 beschriftet ist. Andernfalls beschriften wir Knoten v mit 0. Der Beschriftung der Knoten mit den Werten 1 oder 0 kann man nun ablesen, ob der betreffende Teilbaum ein mit 1 beschriftetes Blatt enthält oder nicht. Die gegebene Formel α ist also genau dann erfüllbar, wenn die Wurzel des Entscheidungsbaums für α mit 1 beschriftet ist.

Algorithmus 6 SAT(α, i)

```

IF  $i = n$  THEN
    berechne den Wahrheitswert  $\alpha^I$  von  $\alpha$  unter  $I = [x_1 = b_1, \dots, x_n = b_n]$ 
    IF  $\alpha^I = 1$  THEN return true ELSE return false FI
ELSE
     $b_{i+1} := 0$ 
    IF SAT( $\alpha, i + 1$ ) THEN return true FI ;
     $b_{i+1} := 1$ 
    IF SAT( $\alpha, i + 1$ ) THEN return true ELSE return false FI
FI

```

Die Erzeugung und Bewertung des Entscheidungsbaums nach dem Preorder-Prinzip kann leicht mit einem *Backtracking*-Algorithmus implementiert werden, der mit n globalen Booleschen Variablen b_1, \dots, b_n arbeitet, welche für die Belegungen der Atome x_1, \dots, x_n stehen. Das Verfahren ist in Algorithmus 6 skizziert, welches initial mit SAT($\alpha, 0$) aufzurufen ist. Die Terminierung des Verfahrens ist klar, da die Rekursionstiefe durch n (Anzahl an Atomen) beschränkt ist. Falls SAT($\alpha, 0$) mit der Rückgabe *true* terminiert, so stehen die aktuellen Werte b_1, \dots, b_n für eine erfüllende Belegung für α . In diesem Fall ist α also erfüllbar. Andernfalls, also falls SAT($\alpha, 0$) mit der Rückgabe *false* terminiert, so enthält der Entscheidungsbaum kein mit 1 bewertetes Blatt. Die Eingabeformel α ist daher unerfüllbar. Damit ist die partielle Korrektheit des Verfahrens klar. Da für eine unerfüllbare Eingabeformel α alle 2^n Belegungen zu betrachten sind, ist die worst-case Laufzeit exponentiell. Aufgrund der exponentiellen Laufzeit ist nur für Formeln mit hinreichend kleiner Anzahl n an Atomen in zumutbarer Zeit eine Antwort des naiven SAT-Beweisers zu erwarten ist. Ein kleines Rechenbeispiel illustriert diese Aussage. Nimmt man an, dass α unerfüllbar ist und $n = 80$ Atome hat und dass in 1 Millisekunde die Wahrheitswerte von α unter 1.000 Belegungen berechnet werden können, so ist die benötigte Rechenzeit $2^{80}/1.000.000$ Sekunden $> 2^{34}$ Jahre.

Wir bereits erwähnt zählt SAT zu den algorithmisch schwierigen Problemen, für die es vermutlich keine effiziente Lösung gibt. Dennoch gibt es eine Reihe von ausgefeilten Algorithmen, die zwar ebenfalls exponentielle worst-case Laufzeit haben, jedoch für viele praxisrelevante Formeln recht schnell ein Ergebnis liefern. Im Folgenden stellen wir Algorithmen vor, welche als SAT-Beweiser für KNF-Formeln dienen. Der erste Algorithmus geht von einer beliebigen KNF-Formel aus und ist somit durch eine vorangeschaltete Transformation, welche eine gegebene aussagenlogische Formel in eine äquivalente KNF-Formel überführt, für alle aussagenlogischen Formeln einsetzbar. Mit einem Trick, den wir später erläutern, kann das etwaige exponentielle Längenwachstum durch die Erstellung äquivalenter KNF-Formeln verhindert werden.

KNF-SAT-Beweiser

Wir beginnen mit einer Reihe von einfachen Beobachtungen, die im Kontext von KNF-SAT-Beweisern nützlich sind. Wir gehen dabei stets von einer KNF-Formel $\alpha = \kappa_1 \wedge \dots \wedge \kappa_m$ aus, wobei $m \geq 0$ und $\kappa_1, \dots, \kappa_m$ Klauseln sind. Zunächst halten wir zwei triviale Sonderfälle fest, in denen die Unerfüllbarkeit bzw. Erfüllbarkeit von α sofort entschieden werden kann:

1. *Sonderfall: Keine Klausel.* Ist α eine KNF-Formel mit 0 Klauseln, so ist $\alpha = true$. In diesem Fall ist α erfüllbar (sogar gültig).

2. *Sonderfall: Leere Klausel.* Falls eine der Klauseln von α leer ist, so ist α von der Form

$$\alpha = \dots \wedge \square \wedge \dots = \dots \wedge false \wedge \dots$$

und somit unerfüllbar. (Wir erinnern daran, dass das Symbol \square als Schreibweise für die leere Klausel eingeführt wurde. Es entspricht der Formel *false*.)

Lemma 4.23 (Einfache Sonderfälle von KNF-SAT).

- (a) *Falls jede Klausel von α wenigstens ein positives Literal enthält, so ist α erfüllbar.*
- (b) *Falls jede Klausel von α wenigstens ein negatives Literal enthält, so ist α erfüllbar.*
- (c) *Jede KNF-Formel α ohne leere Klauseln, in welcher jedes Atom $x \in AP$ entweder nur positiv oder nur negativ vorkommt, ist erfüllbar.*

Beweis. ad (a). Liegt eine KNF-Formel vor, in der jede Klausel wenigstens ein positives Literal enthält, so ist die Belegung, die jedem Aussagensymbol x den Wahrheitswert 1 zuordnet, eine erfüllende Belegung für α . Insbesondere ist α erfüllbar.

Die Argumentation für (b) ist analog. Falls jede Klausel von α wenigstens ein negatives Literal enthält, so ist die Belegung, die jedem Aussagensymbol x den Wahrheitswert 0 zuordnet, eine erfüllende Belegung für α .

ad (c). Sei α eine KNF-Formel ohne leere Klauseln, in welcher jedes Atom $x \in AP$ entweder nur positiv oder nur negativ vorkommt. Offenbar ist die Belegung I , welche jedes Atom x , das nur positiv in α vorkommt, mit 1 bewertet und jedes andere Atom mit 0, erfüllend für α . Z.B. ist

$$(x \vee \neg y) \wedge (x \vee \neg z) \wedge (\neg y \vee \neg z)$$

eine solche KNF-Formel, da x nur positiv und y und z nur negativ vorkommen. Die Belegung $I = [x = 1, y = 0, z = 0]$ ist offenbar erfüllend. \square