Teil III: Berechenbarkeit

§11. Turingmaschinen

§12. Zusammenhang zwischen Turingmaschinen und Grammatiken

§13. LOOP-Programme und WHILE-Programme

§14. Primitiv-rekursive Funktionen und μ-rekursive Funktionen

§15. Entscheidbare und Unentscheidbare Probleme

§16. Weitere unentscheidbare Probleme

§17. Semi-Entscheidbarkeit und Rekursive Aufzählbarkeit

Teil IV: Komplexität

§18. Komplexitätsklassen

§19. NP-vollständige Probleme

§20. Jenseits von NP

Einschub Aussagenlogik

Logik wird in der Informatik, Mathematik und Philosophie studiert

Es gibt viele logische Formalismen, z.B.:
Aussagenlogik, Prädikatenlogik erster und höherer Stufe, μ-Kalkül, usw.

Verschiedenste Anwendungen in der Informatik, zum Beispiel:

- Aussagenlogik im Schaltkreisentwurf und Chip Design
- Logik erster Stufe als Theoretisches Fundament von SQL
- μ-Kalkül als Spezifikationssprache in der Verifikation

Wir betrachten hier Aussagenlogik als Element der Komplexitätstheorie

Aussagenlogik behandelt logische Verknüpfung von Aussagen mittels Junktoren wie und, oder, nicht

Jeder Aussage ist ein Wahrheitswert (wahr/falsch) zugeordnet

Betrachtet wird nun der Wahrheitswert zusammengesetzter Aussagen:

x oder y wahr gdw. x wahr oder y wahr

x könnte z.B. stehen für Die Erde ist ein Planet und y für Bremen liegt am Ganges. Davon wird abstrahiert.

Die Ausdrucksstärke von Aussagenlogik ist äußerst begrenzt

Es ergeben sich jedoch interessante algorithmische Probleme, unter anderem aus Sicht der Komplexitätstheorie

Wir fixieren eine abzählbar unendliche Menge VAR von Aussagenvariablen.
Diese Variablen bezeichnen wir mit x, y, z

Definition B.1 (Syntax Aussagenlogik)

Die aussagenlogischen Formeln sind induktiv wie folgt definiert:

- jede Aussagenvariable ist eine aussagenlogische Formel
- wenn φ und ψ aussagenlogische Formeln, dann auch:
  - $\neg\varphi$ (Negation)
  - $(\varphi \wedge \psi)$ (Konjunktion) und
  - $(\varphi \vee \psi)$ (Disjunktion).

Wir lassen die Klammern weg, wenn das Resultat eindeutig ist.

Dabei bindet $\neg$ stärker als $\wedge$ und $\vee$

Zum Beispiel:

- $\neg x \wedge y$ steht für $(\neg x) \wedge y$, nicht etwa für $\neg(x \wedge y)$
- $x_1 \wedge y_1 \vee x_2 \wedge y_2$ ist ohne Klammern nicht erlaubt, da wir zwischen $\wedge$ und $\vee$ keine Präzedenz verinbart haben.

Definition B.2 (Semantik Aussagenlogik)

Eine (aussagenlogische) Belegung ist Abbildung $V : \mathsf{VAR} \to \{0, 1\}$.

Erweiterung einer Belegung V auf zusammengesetzte Formeln:

- $V(\neg\varphi) = 1 - V(\varphi)$
- $V(\varphi \wedge \psi) = \begin{cases} 1 & \text{falls } V(\varphi) = 1 \text{ und } V(\psi) = 1 \\ 0 & \text{sonst} \end{cases}$
- $V(\varphi \vee \psi) = \begin{cases} 1 & \text{falls } V(\varphi) = 1 \text{ oder } V(\psi) = 1 \\ 0 & \text{sonst} \end{cases}$

Wenn $V(\varphi) = 1$, dann erfüllt die Belegung V die Formel φ.
Wir sagen auch: V is ein Modell von φ.

Die Semantik der Junktoren übersichtlich als Wahrheitstafeln:

$V(\varphi)$	$V(\neg\varphi)$
0	1
1	0

$V(\varphi)$	$V(\psi)$	$V(\varphi \wedge \psi)$
0	0	0
0	1	0
1	0	0
1	1	1

$V(\varphi)$	$V(\psi)$	$V(\varphi \vee \psi)$
0	0	0
0	1	1
1	0	1
1	1	1

Es gibt natürlich noch weitere Junktoren, zum Beispiel:

$V(\varphi)$	$V(\psi)$	$V(\varphi \rightarrow \psi)$
0	0	1
0	1	1
1	0	0
1	1	1

Implikation

$V(\varphi)$	$V(\psi)$	$V(\varphi \leftrightarrow \psi)$
0	0	1
0	1	0
1	0	0
1	1	1

Biimplikation

Definition B.3 (Logische Äquivalenz)

Zwei Formeln φ und ψ sind äquivalent, geschrieben $\varphi \equiv \psi$,
wenn für alle Belegungen V gilt: $V(\varphi) = V(\psi)$.

Zwei äquivalente Formeln sind also austauschbar ohne Bedeutungsänderung

Erweiterung der Aussagenlogik um Implikation und Biimplikation erhöht nicht die Ausdrucksstärke:

- $\varphi \to \psi$ ist äquivalent zu $\neg\varphi \vee \psi$
- $\varphi \leftrightarrow \psi$ ist äquivalent zu $(\varphi \to \psi) \wedge (\psi \to \varphi)$

Man kann sogar zeigen: Jeder mögliche Junktor (beliebiger Stelligkeit) kann mittels $\neg$, $\wedge$, $\vee$ ausgedrückt werden.

Wir können "$\to$" und "$\leftrightarrow$" also o.B.d.A. verwenden.

Dabei binden $\neg$, $\wedge$, $\vee$ stärker als $\rightarrow$ und $\leftrightarrow$

$x \wedge y \rightarrow z$ steht also für $(x \wedge y) \rightarrow z$, nicht etwa für $x \wedge (y \rightarrow z)$

Einige weitere nützliche Äquivalenzen:

φ	$\equiv$	$\neg\neg\varphi$	Doppelte Negation
$\neg(\varphi \wedge \psi)$	$\equiv$	$\neg\varphi \vee \neg\psi$	De Morgansches Gesetz
$\neg(\varphi \vee \psi)$	$\equiv$	$\neg\varphi \wedge \neg\psi$	De Morgansches Gesetz
$\varphi \wedge \varphi$	$\equiv$	φ	Idempotenz von Konjunktion
$\varphi \vee \varphi$	$\equiv$	φ	Idempotenz von Disjunktion
$\varphi \wedge \psi$	$\equiv$	$\psi \wedge \varphi$	Kommutativität von Konjunktion
$\varphi \vee \psi$	$\equiv$	$\psi \vee \varphi$	Kommutativität von Disjunktion
$(\varphi \wedge \psi) \wedge \vartheta$	$\equiv$	$\varphi \wedge (\psi \wedge \vartheta)$	Assoziativität von Konjunktion
$(\varphi \vee \psi) \vee \vartheta$	$\equiv$	$\varphi \vee (\psi \vee \vartheta)$	Assoziativität von Disjunktion
$\varphi \wedge (\psi \vee \vartheta)$	$\equiv$	$(\varphi \wedge \psi) \vee (\varphi \wedge \vartheta)$	Distributivgesetz
$\varphi \vee (\psi \wedge \vartheta)$	$\equiv$	$(\varphi \vee \psi) \wedge (\varphi \vee \vartheta)$	Distributivgesetz

Definition B.4 (KNF / DNF)

Ein Literal ist Formel der Form x oder $\neg x$ mit x Aussagenvariable.

Eine aussagenlogische Formel ist in

- konjunktiver Normalform (KNF) wenn sie eine Konjunktion von Disjunktionen von Literalen ist:

  $$(\ell_{1,1} \vee \cdots \vee \ell_{1,m_1}) \wedge \cdots \wedge (\ell_{1,n} \vee \cdots \vee \ell_{1,m_n})$$

  wobei $\ell_{i,j}$ Literale sind;

- disjunktiver Normalform (DNF) wenn sie eine Disjunktion von Konjunktionen von Literalen ist:

  $$(\ell_{1,1} \wedge \cdots \wedge \ell_{1,m_1}) \vee \cdots \vee (\ell_{1,n} \wedge \cdots \wedge \ell_{1,m_n})$$

  wobei $\ell_{i,j}$ Literale sind.

Eine Formel der Form $\ell_{1,1} \vee \cdots \vee \ell_{1,m_1}$ nennen wir Klausel.

Eine Formel in KNF ist also eine Konjunktion von Klauseln.

Satz B.5

Jede aussagenlogische Formel ist äquivalent zu einer Formel in DNF und zu einer Formel in KNF.

Beweis:

Sei φ Formel mit Variablen $x_1, \ldots, x_n$

Zu jeder Belegung V der Variablen in φ definiere Formel

$$\vartheta_V = \ell_1 \wedge \cdots \wedge \ell_n$$

wobei $\ell_i = x_i$ wenn $V(x_i) = 1$ und $\ell_i = \neg x_i$ wenn $V(x_i) = 0$.

Seien $V_1, \ldots, V_k$ alle Belegungen der Variablen in φ, so dass $V_i(\varphi) = 1$. Setze

$$\psi := \vartheta_{V_1} \vee \cdots \vee \vartheta_{V_k}.$$

Offensichtlich ist ψ in DNF. Zudem ist ψ äquivalent zu φ.

Bleibt zu zeigen: φ ist äquivalent zu Formel in KNF.

Umwandlung in folgenden Schritten:

1. φ äquivalent zu $\neg\neg\varphi$;

2. in $\neg\neg\varphi$ kann man Teilformel $\neg\varphi$ in DNF wandeln,
   also ist φ äquivalent zu Formel der Form
   $$\neg(\ (\ell_{1,1} \wedge \cdots \wedge \ell_{1,m_1}) \vee \cdots \vee (\ell_{1,n} \wedge \cdots \wedge \ell_{1,m_n})\ ).$$

3. Anwendung der de Morganschen Gesetze ergibt zunächst
   $$\neg(\ell_{1,1} \wedge \cdots \wedge \ell_{1,m_1}) \wedge \cdots \wedge \neg(\ell_{1,n} \wedge \cdots \wedge \ell_{1,m_n})$$

4. nochmaliges Anwenden von de Morgan liefert die gewünschte KNF
   $$(\neg\ell_{1,1} \vee \cdots \vee \neg\ell_{1,m_1}) \wedge \cdots \wedge (\neg\ell_{1,n} \vee \cdots \vee \neg\ell_{1,m_n})$$

Beachte:

Umwandlung in DNF und KNF kann Formel exponentiell vergrössern

(2^n viele Disjunkte / Konjunkte, mit n Anzahl Variablen)

Das läßt sich auch durch bessere Konstruktion nicht verhindern:

Man kann z.B. zeigen, dass für die n-stellige Paritätsfunktion gilt:

- sie kann mit Formel der Größe $\mathcal{O}(n)$ dargestellt werden
- jede DNF hat mindestens 2^n Disjunkte
- jede KNF hat mindestens 2^n Konjunkte

Aus logischer Sicht interessant sind die gültigen und unerfüllbaren Formeln

Definition B.6 (Erfüllbarkeit / Gültigkeit)

Eine Formel φ heisst

- erfüllbar wenn es eine Belegung V gibt, die φ erfüllt;
- gültig wenn φ von jeder Belegung V erfüllt wird.

Gültige Formeln sind also allein aufgrund ihrer logischen Form wahr, unabhängig vom Wahrheitswert der verwendeten Variablen.

Unerfüllbare Formeln sind allein aufgrund ihrer logischen Form falsch.

Lemma B.7

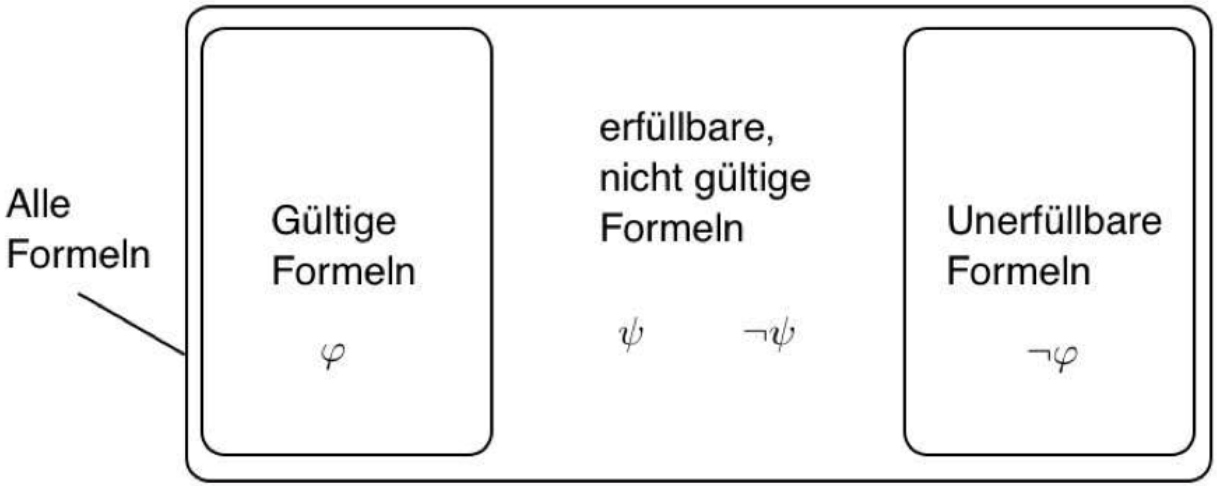
Für jede Formel φ gilt:

1. φ ist erfüllbar gdw. $\neg\varphi$ nicht gültig ist;
2. φ ist gültig gdw. $\neg\varphi$ unerfüllbar ist.

Beweis:

(1) φ ist erfüllbar
gdw. eine Belegung V existiert, die φ erfüllt
gdw. eine Belegung V existiert, die $\neg\varphi$ nicht erfüllt (Semantik "$\neg$")
gdw. $\neg\varphi$ nicht gültig.

(2) ähnlich


Alle
Formeln
Gültige
Formeln
φ
erfüllbare,
nicht gültige
Formeln
ψ
$\neg\psi$
Unerfüllbare
Formeln
$\neg\varphi$

Die wichtigsten Entscheidungsprobleme der Aussagenlogik sind:

Definition B.8 (Entscheidungsprobleme)

Auswertungsproblem: gegeben Formel φ und Belegung V der Variablen in φ, entscheide, ob φ von V erfüllt wird.

Erfüllbarkeitsproblem: gegeben Formel φ, entscheide, ob φ erfüllbar.

Gültigkeitsproblem: gegeben Formel φ, entscheide, ob φ gültig.

Das Auswertungsproblem läßt sich einfach und effizient lösen

Erfüllbarkeits- und Gültigkeitsproblem erweisen sich als wesentlich anspruchsvoller

Lemma B.9

Das Auswertungsproblem der Aussagenlogik ist in polynomieller Zeit lösbar.

Beweis:

Semantik legt rekursiven Algorithmus nahe:

procedure auswertung(φ, V)
case
$\varphi = x$ do return $V(x)$
$\varphi = \neg\psi$ do return $1-$auswertung(ψ, V)
$\varphi = \psi \wedge \vartheta$ do return $\min\{$auswertung(ψ, V),auswertung$(\vartheta, V)\}$
$\varphi = \psi \vee \vartheta$ do return $\max\{$auswertung(ψ, V),auswertung$(\vartheta, V)\}$
end case

Anzahl rekursiver Aufrufe ist beschränkt durch Anzahl Teilformeln von φ

Also läuft der Algorithmus in polynomieller Zeit.

Ein naiver Algorithmus für das Erfüllbarkeitsproblem:

- Gegeben sei Formel φ mit Variaben $x_1, \ldots, x_n$
- Zähle alle Belegungen V für $x_1, \ldots, x_n$ auf
- Für jede Belegung prüfe, ob $V(\varphi) = 1$ (Auswertungsproblem)
- Wenn das jemals der Fall ist, antworte ja; sonst antworte nein

Das Gültigkeitsproblem löst man in analoger Weise.

Der Algorithmus benötigt offensichtlich exponentielle Zeit (2^n Belegungen)

Es gibt Algorithmen, die zielgerichteter Arbeiten, wie etwa Resolution

Allerdings ist kein Algorithmus bekannt, der in polynomieller Zeit arbeitet.

Das zuvor gesagte gilt auch für Erfüllbarkeit von KNF-Formeln und Gültigkeit von DNF-Formeln

Die umgekehrten Fälle sind allerdings leicht in polynomieller Zeit lösbar:

Lemma B.10 (Einfache Fälle)

1. Eine DNF-Formel ist erfüllbar gdw. es ein Disjunkt gibt, das keine Literale der Form $x, \neg x$ enthält.
2. Eine KNF-Formel ist gültig gdw. jedes Konjunkt zwei Literale der Form $x, \neg x$ enthält.

Erfüllbarkeits- und Gültigkeitsproblem sind schwierig wegen Disjunktion

Zum Beispiel: Formel der Form

$$\varphi_i = \bigwedge_{i=1..n} y_i \vee y_i'$$

erzeugt 2^n viele Auswahlmöglichkeiten

Negation führt implizit zu Disjunktion, zum Beispiel erzeugt

$$\varphi_i \equiv \bigwedge_{i=1..n} (x \to y_i \wedge \neg x \to y_i')$$

dieselben 2^n Auswahlmöglichkeiten.

Horn Formeln sind ein disjunktionsfreies Fragment der Aussagenlogik.

Definition B.11 (Horn Formel)

Eine aussagenlogische Formel $\varphi = \bigwedge_i \bigvee_j \ell_{i,j}$ in KNF heißt Horn-Formel wenn jedes Konjunkt $\bigvee_j \ell_{i,j}$ höchstens ein positives Literal enthält.

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

Anschaulicher schreibt man die Disjunktionen als Implikationen:

		x	Fakt
$\neg x_1 \vee \cdots \vee \neg x_k \vee y$	$\equiv$	$x_1 \wedge \cdots \wedge x_k \rightarrow y$	Regel
$\neg x_1 \vee \cdots \vee \neg x_k$	$\equiv$	$x_1 \wedge \cdots \wedge x_k \rightarrow 0$	Constraint

Abkürzung für $x \wedge \neg x$

Horn Formeln wichtig für Datenbanken und in der künstlichen Intelligenz

Theorem B.12

Das Erfüllbarkeitsproblem für Horn-Formeln ist in polynomieller Zeit lösbar.

Der folgende Algorithmus HORN erhält als Eingabe Horn-Formel φ:

$V := \{x \in \mathsf{VAR} \mid x \text{ ist Konjunkt von } \varphi\}$
while es gibt Konjunkt $x_1 \wedge \cdots \wedge x_k \to y$ mit $\{x_1, \ldots, x_k\} \subseteq V$ und $y \notin V$
do
    $V := V \cup \{y\}$
done
if es gibt ein Konjunkt $x_1 \wedge \cdots \wedge x_k \to 0$ mit $\{x_1, \ldots, x_k\} \subseteq V$ then
    return "unerfüllbar"
else
    return "erfüllbar"

Lemma B.13

Der Algorithmus HORN ist korrekt und benötigt nur polynomielle Zeit.

Polynomielle Zeit:

- In jedem Durchlauf der while-Schleife wird die Menge V größer
- Anzahl Durchläufe also beschränkt durch Anzahl Variablen in φ
- Alle anderen Operationen in polynomieller Zeit realisierbar.

Lemma B.13

Der Algorithmus HORN ist korrekt und benötigt nur polynomielle Zeit.

Angenommen, der Algorithmus antwortet "erfüllbar"

Wir zeigen: V erfüllt φ (wobei $V(x) = 1$ wenn $x \in \varphi$)

Drei Arten von Konjunkten:

- Fakt x

  Dann $x \in V$, also $V(x) = 1$

- Regel $x_1 \wedge \cdots \wedge x_n \rightarrow y$

  Wenn $\{x_1, \ldots, x_n\} \not\subseteq V$, dann erfüllt V die Regel.

  Andernfalls $y \in V$ (while-Scheife), also erfüllt V die Regel.

- Constraint $x_1 \wedge \cdots \wedge x_n \rightarrow 0$

  Dann $\{x_1, \ldots, x_n\} \not\subseteq V$ wegen Antwort "erfüllbar",

  also erfüllt V das Constraint.

Lemma B.13

Der Algorithmus HORN ist korrekt und benötigt nur polynomielle Zeit.

Angenommen, die Eingabeformel φ ist erfüllbar

Wir zeigen zunächst: $(*)$ $V \subseteq \widehat{V}$ für alle Modelle $\widehat{V}$ von φ

Induktion über Anzahl Durchläufe der while-Schleife:

Anfang: $x \in V$ weil x Konjunkt von φ

Also $x \in \widehat{V}(x)$ wenn $\widehat{V}$ Modell von φ.

Schritt: $y \in V$ wegen $x_1 \wedge \cdots \wedge x_n \rightarrow y$ in φ und $\{x_1, \ldots, x_n\} \subseteq V$

Ind.Vor. liefert $\{x_1, \ldots, x_n\} \subseteq \widehat{V}$ wenn $\widehat{V}$ Modell von φ, also $y \in \widehat{V}$

Zu zeigen: für alle $x_1 \wedge \cdots \wedge x_n \rightarrow 0$ in φ gilt $\{x_1, \ldots, x_n\} \not\subseteq V$,

denn dann antwortet der Algorithmus "erfüllbar''.

Das folgt aber aus $(*)$ und der Erfüllbarkeit von φ.