Vorlesung Logik SoSe 2014

Prof.Dr. Jacobo Torán

Mo 12–14, H20

Übungen:

Simon Straub

Logik

Logik ist ein Versuch zur Formalisierung und Mechanisierung des menschlischen Schließens.

In der formalen Logik wird untersucht, wie man Aussagen miteinander verknüpfen kann und auf welche Weise man Schlüsse zieht und Beweise durchführt.

Traditionelle Logik Aristoteles, Leibniz ... stark an der Philosophie verbunden.

Mathematische Logik Boole, Frege, Russel, Hilbert ... mathematisch geprägt.

Logik und Informatik

Programmverifikation und Spezifikation

Semantik der Programmirsprachen

Automatisches Beweisen

Logik-Programmierung

Komplexe Datenstrukturen

Automatische Erzeugung von Algorithmen

Vorlesungsinhalt

- Aussagenlogik
  - Syntax und Semantik
  - Hornformeln
  - Endlichkeitssatz
  - Resolution
- Prädikatenlogik
  - Mathematische Theorien
  - Normalformen
  - Herbrand-Theorie
  - Resolution
  - Logik-Programme

Heute ist Montag oder Mittwoch

Mittwochs findet meine Vorlesung im Raum H121 statt

Wir sind im H20

Heute ist Montag

Sokrates ist ein Mensch

alle Menschen sind sterblich,

dann ist Sokrates sterblich.

Drachen mögen Jungfrauen

Grüne Drachen mögen keine Menschen

Jungfrauen sind Menschen

Das Schubfachprinzip

A_{ij} bezeichnet, dass Ding i in Kasten j ist.

$n+1$ Dinge, n Kasten.

$(A_{i1} \vee A_{i2} \vee A_{i3} \vee \ldots \vee A_{in})$

$F_n = \bigwedge_{i=1}^{n+1} \bigvee_{j=1}^{n} A_{ij}$

$\neg A_{ij} \vee \neg A_{kj}$ für $i \neq k$

$G_n = \bigwedge_{j=1}^{n} \bigwedge_{i=1}^{n} \bigwedge_{k=i+1}^{n+1} (\neg A_{ij} \vee \neg A_{kj})$

Schubfachprinzip $\neg(F_n \wedge G_n)$

Aussagenlogik

Aussagen sind atomare sprachliche Gebilde die falsch oder wahr sein können, wie z.B.

–Heute ist Dienstag

–Berlin ist die Haupstadt von Frankreich

Aussagenlogische Formeln (Syntax)

Menge von atomaren Formeln oder Aussagen $A, B, C, \ldots$ oder $A_1, A_2, A_3, \ldots$.

Formel F:

- Wenn $F = 0$ oder $F = 1$ oder F eine atomare Formel ist, so ist F eine Formel.
- Wenn F bereits eine Formel ist, dann auch $\neg F$.
- Wenn F und G bereits Formeln sind, dann auch $(F \wedge G)$ und $(F \vee G)$.

Semantik

Wahrheitswerte $\{0, 1\}$

Belegung $\mathcal{A} : \{\text{Atomaren Formeln}\} \to \{0, 1\}$

Wir erweitern den Definitionsbereich einer Belegung $\mathcal{A}$ auf beliebige Formeln (die aus den atomaren Formeln, auf denen $\mathcal{A}$ definiert ist, aufgebaut sind):

1. $\mathcal{A}(0) = 0, \quad \mathcal{A}(1) = 1.$
2. $\mathcal{A}(\neg F) = \begin{cases} 1, \text{ falls } \mathcal{A}(F) = 0 \\ 0, \text{ sonst} \end{cases}$

3. $\mathcal{A}((F \wedge G)) = \begin{cases} 1, \text{ falls } \mathcal{A}(F) = 1 \text{ und } \mathcal{A}(G) = 1 \\ 0, \text{ sonst} \end{cases}$

4. $\mathcal{A}((F \vee G)) = \begin{cases} 1, \text{ falls } \mathcal{A}(F) = 1 \text{ oder } \mathcal{A}(G) = 1 \\ 0, \text{ sonst} \end{cases}$

Sei F eine Formel und $\mathcal{A}$ eine Belegung. Falls $\mathcal{A}$ für alle in F vorkommenden atomaren Formeln definiert ist, so heißt $\mathcal{A}$ zu F *passend*.

Falls $\mathcal{A}$ zu F passend ist und $\mathcal{A}(F) = 1$ gilt, so schreiben wir auch: $\mathcal{A} \models F$.

$\mathcal{A}$ *ist ein Modell für* F.

Eine Formel F heißt *erfüllbar*, falls F mindestens ein Modell besitzt, andernfalls heißt F *unerfüllbar* (oder widerspruchsvoll).

Eine (evtl. unendliche) Menge von Formeln M heißt erfüllbar, falls es eine Belegung $\mathcal{A}$ gibt, die für jede Formel in M ein Modell ist.

Eine Formel F heißt *gültig* (oder *Tautologie*), falls jede zu F passende Belegung ein Modell für F ist. Diesen Sachverhalt notieren wir durch $\models F$.

Satz: F ist eine Tautologie genau dann, wenn $\neg F$ unerfüllbar ist.

Um den Wahrheitswert einer Formel F festzustellen – unter allen möglichen Belegungen der in F vorkommenden atomaren Formeln – kann man Wahrheitstafeln zu Hilfe nehmen:

	A_1	A_2	$\cdots$	A_{n-1}	A_n	F
$\mathcal{A}_1$:	0	0		0	0	$\mathcal{A}_1(F)$
$\mathcal{A}_2$:	0	0		0	1	$\mathcal{A}_2(F)$
$\vdots$			$\ddots$			$\vdots$
$\mathcal{A}_{2^n}$:	1	1		1	1	$\mathcal{A}_{2^n}(F)$

Def: $F \equiv G$

Wenn zwei Formeln F und G unter jeder (zu F und G passenden) Belegung denselben Wahrheitswert erhalten, so nennen wir F und G (semantisch) *äquivalent*

$F \equiv G$ genau dann, wenn $\models (F \leftrightarrow G)$.

Def: $F \models G$

Wenn für jede (zu F und G passenden) Belegung gilt $\mathcal{A}(F) \to \mathcal{A}(G)$ ($\mathcal{A}(F) = 0$ oder $\mathcal{A}(G) = 1$) dann sagen wir, dass G logische Folgerung von F ist.

$F \models G$ genau dann, wenn $\models (F \to G)$.

Hornformeln

Def: Eine Formel F heißt *Hornformel*, falls F in KNF vorliegt und jedes Disjunktionsglied in F *höchstens ein* positives Literal enthält.

Die Disjunktionsglieder in einer KNF-Formel nennt man auch *Klauseln*.

Zwei Arten von Hornklauseln:

diejenigen in denen *genau ein* positives Literal auftritt, heißen *definite Hornklauseln*;

diejenigen in denen kein positives, sondern nur negative Literale enthalten sind, heißen *Zielklauseln*;

Erfüllbarkeitstest für Hornformeln

$F = F_1 \wedge F_2$,

F_1 sind die definite Hornklauseln und F_2 die Zielklauseln in F.

Wir konstruieren ein Modell M für F_1:

$M := \emptyset$;

WHILE (es gibt eine Klausel der Form $(A_1 \wedge \ldots \wedge A_n) \rightarrow B$ $(n \geq 0)$ in F_1 mit $A_1, \ldots, A_n \in M$ und $B \notin M$) DO
$M := M \cup \{B\}$

END;

Belege alle Atomformeln in M mit 1 und die anderen mit 0;

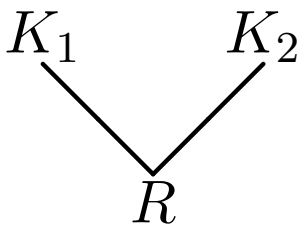
Resolution

F Formel in KNF.

Seien K_1, K_2 und R Klauseln. Dann heißt R *Resolvent* von K_1 und K_2, $(R = Res(K_1, K_2)$ falls es ein Literal L gibt mit $L \in K_1$ und $\overline{L} \in K_2$ und R die Form hat:

$$R = (K_1 - \{L\}) \cup (K_2 - \{\overline{L}\}).$$

R wird aus K_1, K_2 nach L resolviert.



Def: Sei F in KNF. Eine *Resolutionswiderlegung* von F ist eine Folge von Klauseln $K_1, K_2, \ldots, K_n$ mit $K_n = \square$ und der Eigenschaft, dass für jedes i die Klausel K_i entweder Bestandteil von F ist oder ein Resolvent zweier Klauseln K_a und K_b mit $a, b < i$.

Resolutionssatz: Eine Klauselmenge F ist unerfüllbar genau dann, wenn es eine *Resolutionswiderlegung* der leeren Klausel aus F gibt.

Resolution als Operator

Für eine Formel F in KNF können wir definieren:

$$Res(F) = F \cup \{Res(\{K, K'\}) \mid K, K' \in F\}$$

Außerdem setzen wir:

$$\begin{aligned} Res^0(F) &= F \\ Res^{n+1}(F) &= Res(Res^n(F)) \qquad \text{für } n \geq 0 \end{aligned}$$

und schließlich sei

$$Res^*(F) = \bigcup_{n \geq 0} Res^n(F).$$

F ist unerfüllbar $\Longleftrightarrow \square \in Res^*(F)$

Einschränkung des Resolutionskalküls

Positive Resolution oder auch **P-Resolution**

Vorteil: weniger Resolventen als in **Resolution** aber immer noch vollständig.

Definition: Eine Klausel heißt *positiv*, wenn in ihr keine Negation vorkommt.

$$PosRes(F) = F \cup \{Res(\{K, K'\}) \mid K \text{ oder } K' \text{ positiv}, K, K' \in F\}$$

$$\begin{aligned} PosRes^0(F) &= F \\ PosRes^{n+1}(F) &= PosRes(PosRes^n(F)) \qquad \text{für } n \geq 0 \end{aligned}$$

$$PosRes^*(F) = \bigcup_{n \geq 0} PosRes^n(F).$$

Satz: Positive Resolution ist korrekt und vollständig, also

F ist unerfüllbar $\iff \square \in PosRes^*(F)$

Suchen nach einer erfüllenden Belegung

Wir betrachten die folgende Situation:

1. $\square \notin F = Res(F)$. (F ist unter Res abgeschlossen).
2. Wir wollen zeigen, dass F erfüllbar ist und einen Algorithmus geben, der eine (die kleinste) erfüllende Belegung für F konstruiert.

Dieser Algorithmus arbeitet in Runden $i = 1, 2, \ldots$ und konstruiert eine Belegung $\mathcal{A}$, sowie eine Folge von Klauselmengen $F_0 = F, F_1, F_2, F_3, \ldots$. In Runde i wird $\mathcal{A}(x_i)$ festgelegt und F_i berechnet. Es soll gelten:

1: In F_i kommen die Variablen $x_1, x_2, \ldots, x_i$ nicht mehr vor.

2: $\square \notin F_i = Res(F_i)$.

Algorithmus in Runde i

Zwei Fälle unterscheiden:

Fall 1: $\{x_i\} \in F_{i-1}$: $\Rightarrow \mathcal{A}(x_i) = 1; F_i := F_{i-1|_{x_i=1}}$

Fall 2: $\{x_i\} \notin F_{i-1}$: $\Rightarrow \mathcal{A}(x_i) = 0; F_i := F_{i-1|_{x_i=0}}$

Variablenelimination:

Für eine Klausel K bedeutet $K_{|x_i=1}$ (Substitution von x_i durch 1 in K).

$$K_{|x_i=1} = \begin{cases} \top, & \text{falls } x_i \in K \\ K \setminus \{\neg x_i\} & \text{sonst.} \end{cases}$$

$$F_{|x_i=1} = \{K_{|x_i=1} \mid K \in F\}.$$

Die Substitution von x_i durch 0 in K ist in ähnlicherweise definiert:

$$K_{|x_i=0} = \begin{cases} \top, & \text{falls } \neg x_i \in K \\ K \setminus \{x_i\} & \text{sonst.} \end{cases}$$

Für jedes i gilt:

Res$(F_i) = F_i$.

F_{i-1} erfüllbar $\Longrightarrow$ F_i erfüllbar.

Der Algorithmus produziert deswegen eine erfüllende Belegung für F.

Der Algorithmus ist auch korrekt wenn man mit einer Formel F anfängt die unter **positiven** Resolution abgeschlossen ist ($F = PosRes(F)$).
Das kann den Suchraum kleiner machen.

Endlichkeitssatz

Eine Menge M von Formeln ist erfüllbar genau dann, wenn

jede der endlichen Teilmengen von M erfüllbar ist.

(M unerfüllbar $\Longleftrightarrow$ es gibt eine endliche unerfüllbare Teilmenge von M)

Lemma von König: Falls ein Binärbaum T unendlich viele Knoten hat, dann hat T einen unendlichen Pfad der in der Würzel anfängt.

Sei M_i die Menge von Formeln in M die nur die ersten i Atomare Formeln enthalten. (M_i enthält nur endlich viele Formeln die nicht äquivalent zueinander sind).

T Binärbaum. Die Knoten auf Tiefe i repräsentieren die Belegungen für die ersten i Atomare Formeln in M, die M_i erfüllen.