Technische Universität Dresden

Theoretische Informatik und Logik

3. Vorlesung: Verfahren zum logischen Schließen

Markus Krötzsch

Professur Wissensbasierte Systeme

TU Dresden, 20. April 2026

Nachtrag: Currying

Logisches Schließen

Allgemein umfasst logisches Schließen jede Berechnung, bei der semantische Eigenschaften von Formeln ermittelt werden.

Vielfach hat man es mit Entscheidungsproblemen zu tun, z.B.:

Schlussfolgerung:

- **Eingabe:** endliche Formelmenge $\mathcal{F}$, Formel G
- **Ausgabe:** „ja“ wenn $\mathcal{F} \models G$ gilt; sonst „nein.“

Allgemeingültigkeit:

- **Eingabe:** Formel F
- **Ausgabe:** „ja“ wenn F allgemeingültig ist; sonst „nein.“

Unerfüllbarkeit:

- **Eingabe:** Formel F
- **Ausgabe:** „ja“ wenn F unerfüllbar ist; sonst „nein.“

Viele logische Probleme – viele Algorithmen?

Oft kann man unterschiedliche Probleme leicht ineinander umwandeln:

- $\mathcal{F} \models G$ (für endliche $\mathcal{F}$) gdw.
  $\bigwedge_{F \in \mathcal{F}} F \to G$ allgemeingültig gdw. $\bigwedge_{F \in \mathcal{F}} F \wedge \neg G$ unerfüllbar
- F allgemeingültig gdw. $\emptyset \models F$ gdw. $\neg F$ unerfüllbar
- F unerfüllbar gdw. $F \models \bot$ gdw. $\neg F$ allgemeingültig

$\rightsquigarrow$ Jedes Verfahren, welches eines dieser Probleme lösen kann, lässt sich prinzipiell auch zur Lösung der anderen einsetzen.

(Es gibt auch Probleme des logischen Schließens, für die das höchstwahrscheinlich nicht zutrifft, z.B. „Gegeben eine Formel F, entscheide ob es eine kürzere Formel gibt, die zu F äquivalent ist.“)

Schlussfolgerungen checken mit Z3 in Python

```
from z3 import *

# Atome deklarieren:
p,q = Bools('p q')
rs = [ Bool("r_%s" % (i+1))
     for i in range(10) ]

# Formeln hinzufügen:
s = Solver()
s.add(...)

# Schlussfolgerungen prüfen:
print('Checking r_3: ',
   check_entailment(rs[2],s))
```

```
def follows(formula, solver):
   solver.push() # start changes ...
   solver.add(Not(formula))
   result = solver.check()
   solver.pop() # ... undo changes
   return result == unsat

def check_entailment(formula, solver):
   if follows(formula, solver):
      return "true"
   elif follows(Not(formula), solver):
      return "false"
   else:
      return "uncertain"
```

Schließen mit Wahrheitswertetabellen

Die genannten Probleme sind mittels Wahrheitswertetabelle lösbar:

- Allgemeingültigkeit: Ist der Wert von F für jede Belegung 1?
- Unerfüllbarkeit: Ist der Wert von F für jede Belegung 0?
- Schlussfolgerung: Ist der Wert von G für jede Belegung 1, unter der alle Formeln von $\mathcal{F}$ den Wert 1 annehmen?

Nachteil: Die Tabelle hat 2^n Zeilen, wenn die betrachteten Formeln n Atome haben

Beispiel: Für die Modellierung von Sudoku nutzen wir $9 \times 9 \times 9$ Atome. Die entsprechende Tabelle hat also 2^{729} (ca. 2.8×10^{219}) Zeilen. Zum Vergleich:

- Anzahl der Neuronen im menschlichen Gehirn: ca. 1.5×10^{14}
- Jemals digital gespeicherte Daten in Bytes: $<< 10^{24}$
- Nanosekunden bis der Kern der Sonne ausbrennt: ca. 10^{29}
- Anzahl der Atome im beobachtbaren Universum: $10^{78} - 10^{82}$

Normalformen

Normalform

Normalformen sind besondere syntaktische Formen aussagenlogischer Formeln

Sie heißen so, weil sich jede aussagenlogische Formel in eine äquivalente Formel in Normalform umformen lässt

Zweck von Normalformen: Vereinfachung von Algorithmen

- Reduktion der syntaktischen Vielfalt (weniger Fälle)
- Leichtere Verarbeitung (algorithmenfreundliches Format)
- Manche Softwarewerkzeuge (z.B. manche SAT Solver) verlangen Eingaben in einer Normalform

Wir werden hier drei wichtige Normalformen kennenlernen:

- Negationsnormalform (NNF)
- Konjunktive Normalform (KNF)
- Disjunktive Normalform (DNF)

Negationsnormalform

Eine Formel F ist in Negationsnormalform (NNF) wenn

(a) sie nur die Junktoren $\wedge$, $\vee$ und $\neg$ enthält und

(b) der Junktor $\neg$ nur direkt vor Atomen vorkommt (d.h. nur in Teilformeln der Form $\neg p$ mit $p \in \mathbf{P}$).

Formeln, die negierte oder nichtnegierte Atome sind, nennt man Literale. In NNF darf Negation also nur in Literalen auftauchen.

Beispiele:

- $(\neg p \wedge q) \vee (p \wedge \neg q)$ ist in NNF
- $(b \wedge b) \vee \neg(b \wedge b)$ ist nicht in NNF
- $q \vee \neg\neg p$ ist nicht in NNF
- $p \leftrightarrow p$ ist nicht in NNF

Umwandlung in NNF

Es ist möglich, eine Formel rekursiv in NNF umzuformen.

Dazu ersetzen wir zunächst alle Vorkommen von $\rightarrow$ und $\leftrightarrow$ durch äquivalente Formeln (wie zuvor).

Sei F eine Formel, die nur die Junktoren $\wedge$, $\vee$ und $\neg$ enthält. Wir definieren eine Formel $\mathrm{NNF}(F)$ rekursiv wie folgt:

- $\mathrm{NNF}(p) = p$ falls $p \in \mathbf{P}$
- $\mathrm{NNF}(F \wedge G) = \mathrm{NNF}(F) \wedge \mathrm{NNF}(G)$
- $\mathrm{NNF}(F \vee G) = \mathrm{NNF}(F) \vee \mathrm{NNF}(G)$
- $\mathrm{NNF}(\neg p) = \neg p$ falls $p \in \mathbf{P}$
- $\mathrm{NNF}(\neg\neg F) = \mathrm{NNF}(F)$
- $\mathrm{NNF}(\neg(F \wedge G)) = \mathrm{NNF}(\neg F) \vee \mathrm{NNF}(\neg G)$
- $\mathrm{NNF}(\neg(F \vee G)) = \mathrm{NNF}(\neg F) \wedge \mathrm{NNF}(\neg G)$

Beispiel

Wir betrachten die Formel $((p \to q) \to p) \to p$.

Zunächst eliminieren wir Vorkommen von $\to$ in beliebiger Reihenfolge:

$$\begin{aligned} \underline{((p \to q)} \to p) \to p &\equiv \underline{((\neg p \vee q) \to p)} \to p \\ &\equiv \underline{(\neg(\neg p \vee q) \vee p) \to p} \\ &\equiv \neg(\neg(\neg p \vee q) \vee p) \vee p \end{aligned}$$

Anschließend wenden wir NNF an:

$$\begin{aligned} \mathrm{NNF}(\neg(\neg(\neg p \vee q) \vee p) \vee p) &= \mathrm{NNF}(\neg(\neg(\neg p \vee q) \vee p)) \vee \mathrm{NNF}(p) \\ &= (\mathrm{NNF}(\neg\neg(\neg p \vee q)) \wedge \mathrm{NNF}(\neg p)) \vee p \\ &= (\mathrm{NNF}(\neg p \vee q) \wedge \neg p) \vee p \\ &= ((\mathrm{NNF}(\neg p) \vee \mathrm{NNF}(q)) \wedge \neg p) \vee p \\ &= ((\neg p \vee q) \wedge \neg p) \vee p \end{aligned}$$

NNF-Definition (Wiederholung)

Sei F eine Formel, die nur die Junktoren $\wedge$, $\vee$ und $\neg$ enthält. Wir definieren eine Formel $\mathrm{NNF}(F)$ rekursiv wie folgt:

- $\mathrm{NNF}(p) = p$ falls $p \in \mathbf{P}$
- $\mathrm{NNF}(F \wedge G) = \mathrm{NNF}(F) \wedge \mathrm{NNF}(G)$
- $\mathrm{NNF}(F \vee G) = \mathrm{NNF}(F) \vee \mathrm{NNF}(G)$
- $\mathrm{NNF}(\neg p) = \neg p$ falls $p \in \mathbf{P}$
- $\mathrm{NNF}(\neg\neg F) = \mathrm{NNF}(F)$
- $\mathrm{NNF}(\neg(F \wedge G)) = \mathrm{NNF}(\neg F) \vee \mathrm{NNF}(\neg G)$
- $\mathrm{NNF}(\neg(F \vee G)) = \mathrm{NNF}(\neg F) \wedge \mathrm{NNF}(\neg G)$

NNF-Umwandlung: Korrektheit

Ist diese Umformung korrekt?

- Wohldefiniertheit: Deckt die Rekursion wirklich jeden Fall ab?
  Ja, wie man leicht überprüfen kann (der Fall $\neg$ ist nach der Form der negierten Formel nochmals in vier Unterfälle aufgespalten).
- Terminierung: Ist sichergestellt, dass die rekursive Berechnung terminiert?
  Ja, denn NNF wir in jedem Rekursionsschritt auf Formeln angewendet, die insgesamt weniger Junktoren haben als zuvor.
- Korrektheit: Ist das Ergebnis der rekusiven Umwandlung in NNF?
  Ja, denn $\neg$ kommt im Ergebnis nur in Fall $\neg p$ ($p \in \mathbf{P}$) vor.
  Ist das Ergebnis semantisch äquivalent zur Eingabe?
  Ja, wie man durch ein induktives Argument leicht zeigen kann
  Induktionsanfang: Äquivalenz gilt für p und $\neg p$; Induktionshypothese: $F \equiv \mathrm{NNF}(F)$ und $G \equiv \mathrm{NNF}(G)$; Induktionsschritt: für jeden rekursiven Fall folgt Äquivalenz aus Hypothese, Ersetzungstheorem und bekannten Äquivalenzgesetzen.

Satz: Jede Formel kann (in linearer Zeit) in eine äquivalente Formel in NNF umgewandelt werden.

Konjunktive und Disjunktive Normalform

Zur Erinnerung: Literale = negierte oder nichtnegierte Atome

Eine Formel F ist in konjunktiver Normalform (KNF) wenn sie eine Konjunktion von Disjunktionen von Literalen ist, d.h. wenn sie die Form hat:

$$(L_{1,1} \vee \ldots \vee L_{1,m_1}) \wedge (L_{2,1} \vee \ldots \vee L_{2,m_2}) \wedge \ldots \wedge (L_{n,1} \vee \ldots \vee L_{n,m_n})$$

wobei die Formeln $L_{i,j}$ Literale sind. Eine Disjunktion von Literalen heißt Klausel.

Eine Formel F ist in disjunktiver Normalform (DNF) wenn sie eine Disjunktion von Konjunktionen von Literalen ist, d.h. wenn sie die Form hat:

$$(L_{1,1} \wedge \ldots \wedge L_{1,m_1}) \vee (L_{2,1} \wedge \ldots \wedge L_{2,m_2}) \vee \ldots \vee (L_{n,1} \wedge \ldots \wedge L_{n,m_n})$$

wobei die Formeln $L_{i,j}$ Literale sind. Eine Konjunktion von Literalen heißt Monom.

KNF und DNF bilden (Methode 1)

Man kann KNF und DNF direkt aus der Wahrheitswertetabelle ablesen:

p	q	$\neg p \leftrightarrow q$
0	0	0
1	0	1
0	1	1
1	1	0

Disjunktive Normalform:

- Für jede Zeile mit Wert 1, bilde ein Monom mit allen Atomen, wobei genau die Atome mit Wert 0 negiert werden
- Beispiel: $(p \wedge \neg q) \vee (\neg p \wedge q)$

Konjunktive Normalform:

- Für jede Zeile mit Wert 0, bilde eine Klausel mit allen Atomen, wobei genau die Atome mit Wert 1 negiert werden
- Beispiel: $(p \vee q) \wedge (\neg p \vee \neg q)$

KNF und DNF bilden (Methode 2)

Man kann KNF und DNF bilden, indem man die NNF erzeugt und anschließend Distributivgesetze anwendet $\rightsquigarrow$ oft direkter

Konjunktive Normalform

Distributivgesetz: $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$

Beispiel:

$$
\begin{aligned}
(p \wedge \neg q) \vee (\neg p \wedge q) &\equiv ((p \wedge \neg q) \vee \neg p) \wedge ((p \wedge \neg q) \vee q)) \\
&\equiv (p \vee \neg p) \wedge (\neg q \vee \neg p) \wedge ((p \wedge \neg q) \vee q)) \\
&\equiv (p \vee \neg p) \wedge (\neg q \vee \neg p) \wedge (p \vee q) \wedge (\neg q \vee q)
\end{aligned}
$$

(Man könnte die wahren Klauseln $(p \vee \neg p)$ und $(\neg q \vee q)$ streichen.)

Disjunktive Normalform

Distributivgesetz: $F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$ (analog)

Wie effizient sind die Umformungen in KNF?

Ableitung aus Wahrheitswertetabelle

Anzahl der Zeilen (=Klauseln) exponentiell in Variablenzahl

Umformung mit Distributivgesetz

- Negationsnormalform hat lineare Größe
- Distributivgesetz kann Vorkommen von Formeln verdoppeln $\rightsquigarrow$ exponentielles Wachstum

Beispiel: Wir betrachten die Atome a_i und b_i für $i \in \{1, \ldots, n\}$. Für

$$(a_1 \wedge b_1) \vee (a_2 \wedge b_2) \vee \ldots \vee (a_n \wedge b_n)$$

ergibt sich die KNF:

$$\underbrace{(a_1 \vee a_2 \vee \ldots \vee a_n) \wedge (b_1 \vee a_2 \vee \ldots \vee a_n) \wedge \ldots \wedge (b_1 \vee b_2 \vee \ldots \vee b_n)}_{2^n \text{ Klauseln mit allen Kombinationen aus } a_i \text{ und } b_i}$$

$\rightsquigarrow$ Anzahl der Klauseln exponentiell in Variablenzahl

Effizientere Normalformen?

Unsere Umformungen sind schlimmstenfalls exponentiell

- Allgemein unvermeidbar: Es gibt oft keine kleinere KNF/DNF
- Für KNF (aber nicht für DNF!) kann das Problem durch Einführung zusätzlicher Hilfsvariablen gelöst werden

  (Es gibt eine polynomielle Formel in KNF mit zusätzlichen Variablen, welche „bezüglich der Variablen der ursprünglichen Formel“ semantisch äquivalent ist)

Beispiel: $(a_1 \wedge b_1) \vee (a_2 \wedge b_2) \vee \ldots \vee (a_n \wedge b_n)$ ist ausdrückbar als:

$$\begin{aligned} & ((a_1 \wedge b_1) \vee c_1) \\ \wedge\ & (c_1 \leftrightarrow ((a_2 \wedge b_2) \vee c_2)) \\ \wedge\ & (c_2 \leftrightarrow ((a_3 \wedge b_3) \vee c_3)) \\ \ldots \wedge\ & (c_{n-1} \leftrightarrow (a_n \wedge b_n)) \end{aligned}$$

Diese Formel ist „fast“ äquivalent (bis auf die zusätzlichen c_i) und hat eine polynomielle KNF

(Jede Zeile kann einzeln in KNF übersetzt werden, wobei Formeln fester Größe entstehen, von denen es linear viele gibt.)

Schließen mit DNF

Man kann (Un)Erfüllbarkeit leicht aus DNF ablesen:

Satz: Eine Formel in DNF ist genau dann erfüllbar, wenn eines ihrer Monome erfüllbar ist. Dies ist genau dann der Fall, wenn das Monom kein Atom gleichzeitig negiert und nichtnegiert enthält.

Mögliches Verfahren:

- Führe logisches Problem auf (Un)Erfüllbarkeit zurück
- Bilde die DNF und prüfe jedes Monom auf komplementäre Literale

Nachteil: Die DNF kann ebenfalls exponentiell groß werden
(muss sie aber nicht in jedem Fall)

Schließen mit KNF

Man kann Widerlegbarkeit/Allgemeingültigkeit aus KNF ablesen:

Satz: Eine Formel in KNF ist genau dann widerlegbar, wenn eine ihrer Klauseln widerlegbar ist. Dies ist genau dann der Fall, wenn die Klausel kein Atom gleichzeitig negiert und nichtnegiert enthält.

Mögliches Verfahren:

- Führe logisches Problem auf Allgemeingültigkeit zurück
- Bilde die KNF und prüfe jede Klausel auf komplementäre Literale

Nachteil: Die KNF kann ebenfalls exponentiell groß werden
(muss sie aber nicht in jedem Fall)

Können wir durch Hilfsatome eine kleinere KNF erzeugen?
Ja, aber dabei wird nur Erfüllbarkeit erhalten, nicht Allgemeingültigkeit!
$\rightsquigarrow$ nicht geeignet, um das Problem effizienter zu lösen

Zusammenfassung und Ausblick

Das logische Schließen umfasst viele Fragen (meist Entscheidungsprobleme), die aber oft auf Erfüllbarkeit zurückgeführt werden können

Die Negationsnormalform vereinfacht Formeln, indem Negationen „nach unten“ verschoben werden

Konjunktive und disjunktive Normalform stellen beliebige Formeln als Konjunktion bzw. Disjunktion dar, allerdings zum Teil mit erheblichen (exponentiellen) Kosten

Offene Fragen:

- Welche Methoden des logischen Schließens sind praktikabel?
- Was hat Aussagenlogik mit Sprachen, Berechnung und TMs zu tun?