Technische Universität Dresden

Theoretische Informatik und Logik

2. Vorlesung: Logische Schlussfolgerungen und Äquivalenzen

Markus Krötzsch

Professur Wissensbasierte Systeme

TU Dresden, 16. April 2026

Rückblick: Aussagenlogik

Syntax:

$$F \to \mathbf{P} \mid \neg F \mid (F \wedge F) \mid (F \vee F) \mid (F \to F) \mid (F \leftrightarrow F)$$

Semantik:

- Wertzuweisungen $w : \mathbf{P} \to \{1, 0\}$ zur Interpretation von Atomen
- Erweiterung von Atomen auf Formeln:

$w(F)$	$w(\neg F)$
0	1
1	0

$w(F)$	$w(G)$	$w(F \wedge G)$	$w(F \vee G)$	$w(F \to G)$	$w(F \leftrightarrow G)$
0	0	0	0	1	1
1	0	0	1	0	0
0	1	0	1	1	0
1	1	1	1	1	1

Beispiel: Logelei

Anna behauptet: „Barbara lügt!“ $A \leftrightarrow \neg B$

Barbara behauptet: „Chris lügt!“ $B \leftrightarrow \neg C$

Chris behauptet: „Anna und Barbara lügen!“ $C \leftrightarrow (\neg A \wedge \neg B)$

Wer lügt?

(Und wie kann man das beweisen?)

Logische Konsequenzen

Logische Konsequenzen

Eine Wertzuweisung w ist Modell einer Formel F wenn $w \models F$ (also wenn $w(F) = 1$).
Ist $\mathcal{F}$ eine (möglicherweise unendliche) Menge von Formeln, dann ist w ein Modell der Menge $\mathcal{F}$ wenn $w \models F$ für alle $F \in \mathcal{F}$ gilt. In diesem Fall schreiben wir $w \models \mathcal{F}$.

Die logischen Schlussfolgerungen aus einer Formel(menge) ergeben sich aus ihren Modellen:

Sei $\mathcal{F}$ eine Menge von Formeln. Eine Formel G ist eine logische Konsequenz aus $\mathcal{F}$ wenn jedes Modell von $\mathcal{F}$ auch ein Modell von G ist.
In diesem Fall schreiben wir $\mathcal{F} \models G$.

- $\rightsquigarrow$ Die Formeln in $\mathcal{F}$ schränken die möglichen Interpretationen ein:
  Je mehr Formeln wahr sein sollen,
  desto weniger Freiheiten gibt es bei der Wahl der Modelle
- $\rightsquigarrow$ G ist eine logische Konsequenz wenn gilt:
  Falls $\mathcal{F}$ wahr ist, dann ist auch G garantiert wahr

Beispiel: Logelei

Anna behauptet: „Barbara lügt!“ $A \leftrightarrow \neg B$

Barbara behauptet: „Chris lügt!“ $B \leftrightarrow \neg C$

Chris behauptet: „Anna und Barbara lügen!“ $C \leftrightarrow (\neg A \wedge \neg B)$

$w(A)$	$w(B)$	$w(C)$	$w(A \leftrightarrow \neg B)$	$w(B \leftrightarrow \neg C)$	$w(C \leftrightarrow (\neg A \wedge \neg B))$
0	0	0	0	0	0
1	0	0	1	0	1
0	1	0	1	1	1
1	1	0	0	1	1
0	0	1	0	1	1
1	0	1	1	1	0
0	1	1	1	0	0
1	1	1	0	0	0

Beispiel: Logelei (2)

Anna behauptet: „Barbara lügt!“ $A \leftrightarrow \neg B$

Barbara behauptet: „Chris lügt!“ $B \leftrightarrow \neg C$

Chris behauptet: „Anna und Barbara lügen!“ $C \leftrightarrow (\neg A \wedge \neg B)$

$w(A)$	$w(B)$	$w(C)$	$w(A \leftrightarrow \neg B)$	$w(B \leftrightarrow \neg C)$	$w(C \leftrightarrow (\neg A \wedge \neg B))$
0	1	0	1	1	1

$\rightsquigarrow$ Genau ein Modell (bezüglich der relevanten Atome)

Logische Konsequenzen: Alle Formeln, die unter einer Wertzuweisung mit $w(A) = 0$, $w(B) = 1$ und $w(C) = 0$ wahr sind, z.B.:

- $\neg A$ („Anna lügt“)
- $\neg C$ („Chris lügt“)
- $\neg A \wedge \neg C$ („Anna und Chris lügen“)
- B („Barbara sagt die Wahrheit“)
- Aber auch: $D \vee \neg D$

Allgemeingültigkeit und Co.

Eine Formel F ist:

- unerfüllbar (oder inkonsistent), wenn sie keine Modelle hat
- erfüllbar (oder konsistent), wenn sie Modelle hat
- allgemeingültig (oder eine Tautologie), wenn alle Wertzuweisungen Modelle für die Formel sind
- widerlegbar, wenn sie nicht allgemeingültig ist

Diese Begriffe kann man für Mengen von Formeln genauso definieren.

Beispiel:

- $p \wedge \neg p$ ist unerfüllbar (und widerlegbar)
- $p \vee \neg p$ ist allgemeingültig (und erfüllbar)
- $p \wedge q$ ist erfüllbar und widerlegbar

Allgemeingültigkeit und Unerfüllbarkeit

Satz:

(1) Eine allgemeingültige Formel ist logische Konsequenz jeder anderen Formel(menge).

(2) Eine unerfüllbare Formel(menge) hat jede andere Formel als logische Konsequenz.

Beweis: Folgt direkt aus Definition von logischer Konsequenz. Für (2) ist es wichtig, dass eine Eigenschaft „für alle Modelle“ gilt, wenn es keine Modelle gibt (sie gilt dann für „alle null Modelle“). □

Der unerwartete(?) Effekt (2) ist im Deutschen sprichwörtlich:

„Wenn das stimmt, dann bin ich der Kaiser von China!“

drückt aus, dass aus einer mutmaßlich falschen Annahme alles folgt, selbst wenn es offensichtlich unwahr ist.

Logische Schlussfolgerung

Modelle:

$w \models F$ gdw. $w(F) = 1$ gdw. „w ist Modell von F“ gdw. „w erfüllt F“

Logische Konsequenzen:

- $\mathcal{F} \models G$ gdw.
- jedes Modell von $\mathcal{F}$ ist auch ein Modell von G gdw.
- G ist immer wahr wenn alle Formeln in $\mathcal{F}$ wahr sind

Dualität von Modellen und Formeln:

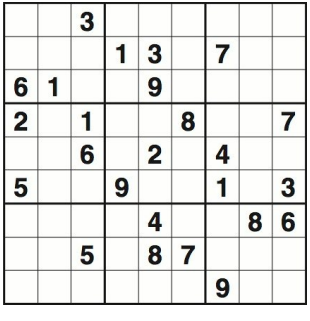
- Je mehr Formeln in $\mathcal{F}$
- desto weniger Modelle erfüllen alle Formeln in $\mathcal{F}$
- desto mehr Formeln sind in allen diesen Modellen wahr
- desto mehr Konsequenzen hat $\mathcal{F}$

$\rightsquigarrow$ Aussagenlogik ist monoton (mehr Annahmen $\Rightarrow$ mehr Schlüsse)

Modellierung in Aussagenlogik

Modellierungsbeispiel: Sudoku (1)

Sudoku ist ein bekanntes Zahlenpuzzle.



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

Aufgabe:

- Fülle jedes Feld mit einer Ziffer von 1 bis 9, so dass
- Spalten, Zeilen und die fetten Teilquadrate jede Ziffer nur einmal enthalten

Modellierungsbeispiel: Sudoku (2)

Wir können Sudoku aussagenlogisch modellieren:

- Atome: Für jede Ziffer $z \in \{1, \ldots, 9\}$ und alle Koordinaten $i, j \in \{1, \ldots, 9\}$ verwenden wir ein Atom $p_z[i, j]$ für die Aussage „an Position (i, j) steht die Ziffer z“
- Spielregeln modellieren wir als logische Formeln:

$$\begin{aligned}
\text{Für alle } i, j\text{:} \quad & p_1[i,j] \vee p_2[i,j] \vee \ldots \vee p_9[i,j] \\
\text{Für alle } i, j, z, z' \text{ mit } z \neq z'\text{:} \quad & p_z[i,j] \rightarrow \neg p_{z'}[i,j] \\
\text{Für alle } i, i', j, z \text{ mit } i \neq i'\text{:} \quad & p_z[i,j] \rightarrow \neg p_z[i',j] \\
\text{Für alle } i, j, j', z \text{ mit } j \neq j'\text{:} \quad & p_z[i,j] \rightarrow \neg p_z[i,j'] \\
\text{Für alle } i, j, i', j', z \text{ mit } (i,j) \neq (i',j')\text{;} & \\
(i,j), (i',j') \text{ im gleichen Teilquadrat:} \quad & p_z[i,j] \rightarrow \neg p_z[i',j']
\end{aligned}$$

- Vorgegebene Ziffern werden als atomare Formeln dargestellt, z.B. $p_3[3, 1]$ im vorigen Beispiel

$\rightsquigarrow$ Modelle der Formelmenge entsprechen Lösungen des Sudoku

Logische Modellierung: Allgemeines Vorgehen

Es ist oft nicht sofort ersichtlich, wie (oder ob) eine Problemstellung gut in Aussagenlogik ausgedrückt werden kann.

Allgemeines Vorgehen:

1. Vokabular festlegen: Welche Atome sollen verwendet werden, um eine Instanz der Frage und seiner Lösung darzustellen?
2. Bedingungen festlegen: Welche aussagenlogischen Formeln müssen gelten, damit die Belegung der Atome eine korrekte Lösung des Problems kodiert?

- Schritt 1 kann zu hunderten oder tausenden Atome führen (die ja jeweils nur ein Bit)
  $\rightsquigarrow$ keine Angst vor großen Vokabularen
- Schritt 2 kann je nach gewählter Kodierung direkter oder umständlicher sein
  $\rightsquigarrow$ gegebenenfalls Kodierung aus Schritt 1 überdenken
- Oft gut darstellbar: Aufgaben deren Lösung eine fest vorgegebene Größe hat und leicht überprüft werden kann (obwohl sie nicht leicht zu finden ist)
  (Sudoku: Ja; nächster Zug im Schach: Nein (nicht einfach zu prüfen); Plan zum Lösen eines Sokoban-Puzzles: Nein (unbekannte, möglicherweise große Schrittzahl))

Äquivalenzen

Logische Äquivalenz

Formeln sind äquivalent, wenn sie die gleiche Semantik haben:

Zwei Formeln F und G sind semantisch äquivalent, in Symbolen $F \equiv G$, wenn sie genau die selben Modelle haben, d.h. wenn

für alle Wertzuweisungen w gilt: $w(F) = w(G)$

Beispiel: $p \to q \equiv \neg p \vee q$, wie man mithilfe der Wahrheitswertetabelle zeigen kann (die Spalten der Formeln sind gleich):

p	q	$p \to q$	$\neg p$	$\neg p \vee q$	*
0	0	1	1	1	
1	0	0	0	0	
0	1	1	1	1	
1	1	1	0	1	

* Vereinfachung: Wir beschriften Tabellenspalten ab jetzt nur mit F statt $w(F)$.

Nützliche Eigenschaften von ≡

Satz: ≡ ist eine Äquivalenzrelation, d.h. reflexiv, symmetrisch und transitiv.

Beweis: ≡ ist definiert als die Gleichheit der Modellmengen. Die gesuchten Eigenschaften ergeben sich, da auch die Relation = auf Mengen eine Äquivalenzrelation ist. □

Weitere Eigenschaften folgen direkt aus den Definitionen:

Satz:

- Alle Tautologien sind semantisch äquivalent
- Alle unerfüllbaren Formeln sind semantisch äquivalent

Satz: Semantische Äquivalenz entspricht wechselseitiger logischer Konsequenz:

$$F \equiv G \qquad \text{genau dann wenn} \qquad F \models G \text{ und } G \models F$$

Nützliche Eigenschaften von $\equiv$

Satz (Ersetzungstheorem): Sei F eine Formel mit einer Teilformel G. Wenn $G \equiv G'$ und wenn F' aus F gebildet werden kann, indem man ein beliebiges Vorkommen von G in F durch G' ersetzt, dann gilt auch $F \equiv F'$.

Beweisskizze*: Aus $G \equiv G'$ folgt, dass $w \models G$ gdw. $w \models G'$ für beliebige Wertzuweisungen w gilt. In der rekursiven Definition von $\models$ erfüllt daher die Teilformel G genau die selben Bedingungen wie G', so dass sich auch für F und F' der selbe Wert ergibt. □

Dieser Satz ist ein Sonderfall der allgemeingültigen und intuitiven Tatsache, dass der Wert von rekursiv definierten Funktionen gleich bleibt, wenn man die syntaktische Definition einer Teilberechnung durch eine andere ersetzt, die stets den gleichen Wert liefert.

* Streng genommen folgt aus der Definition von $\models$ zunächst nur, dass die Ersetzung einer direkten Unterformel Äquivalenz erhält. Zum Beispiel folgt $G \wedge H \equiv G' \wedge H$ aus $G \equiv G'$. Um zu zeigen, dass auch Ersetzungen in beliebiger Tiefe Äquivalenz erhalten, ist noch ein induktives Argument nötig, welches die Äquivalenz im Formelbaum „nach oben“ propagiert. Aus $G \wedge H \equiv G' \wedge H$ folgt z.B. $(G \wedge H) \vee J \equiv (G' \wedge H) \vee J$ usw. usf.

Junktoren äquivalent ausdrücken

Viele Junktoren können durch andere ausgedrückt werden:

$$F \to G \equiv \neg F \vee G \equiv \neg(F \wedge \neg G)$$
$$F \leftrightarrow G \equiv (F \to G) \wedge (G \to F) \equiv (F \wedge G) \vee (\neg F \wedge \neg G)$$
$$F \wedge G \equiv \neg(\neg F \vee \neg G) \quad \text{(De Morgansches Gesetz)}$$
$$F \vee G \equiv \neg(\neg F \wedge \neg G) \quad \text{(De Morgansches Gesetz)}$$

Weitere Junktoren können mithilfe äquivalenter Formeln definiert werden:

$$F \uparrow G \equiv \neg(F \wedge G) \equiv \neg F \vee \neg G \qquad \text{(NAND)}$$
$$F \downarrow G \equiv \neg(F \vee G) \equiv \neg F \wedge \neg G \qquad \text{(NOR)}$$
$$\top \equiv \neg p \vee p \equiv p \to p \qquad \text{(Wahrheit; } p \in \mathbf{P} \text{ beliebig)}$$
$$\bot \equiv \neg p \wedge p \qquad \text{(Falschheit; } p \in \mathbf{P} \text{ beliebig)}$$

Wir werden insbesondere $\top$ und $\bot$ manchmal verwenden.

Junktoren äquivalent ausdrücken (2)

Satz: Sei F eine beliebige aussagenlogische Formel.

- Es gibt eine zu F äquivalente Formel, die nur die Junktoren $\wedge$ und $\neg$ enthält.
- Es gibt eine zu F äquivalente Formel, die nur die Junktoren $\vee$ und $\neg$ enthält.

Beweis: Für den ersten Fall ersetzen wir iterativ Teilformeln, die nicht die gewünschte Form haben, gemäß den folgenden Regeln:

$$
\begin{aligned}
(G \vee H) &\quad\mapsto\quad \neg(\neg G \wedge \neg H) \\
(G \to H) &\quad\mapsto\quad \neg(G \wedge \neg H) \\
(G \leftrightarrow H) &\quad\mapsto\quad (\neg(G \wedge \neg H) \wedge \neg(H \wedge \neg G))
\end{aligned}
$$

Jede Ersetzung führt zu einer äquivalenten Formel (Ersetzungstheorem), der Algorithmus terminiert (jeder Junktor wird höchstens einmal ersetzt) und das Ergebnis hat die gewünschte Form. Der zweite Fall ist analog. □

Nützliche Äquivalenzen (1)

$F \wedge G \equiv G \wedge F$ $F \vee G \equiv G \vee F$	Kommutativität
$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$ $(F \vee G) \vee H \equiv F \vee (G \vee H)$	Assoziativität
$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$ $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$	Distributivität
$F \wedge F \equiv F$ $F \vee F \equiv F$	Idempotenz
$F \wedge (F \vee G) \equiv F$ $F \vee (F \wedge G) \equiv F$	Absorption

Nützliche Äquivalenzen (2)

$\neg\neg F \equiv F$	doppelte Negation
$\neg(F \wedge G) \equiv (\neg F \vee \neg G)$ $\neg(F \vee G) \equiv (\neg F \wedge \neg G)$	De Morgansche Gesetze
$F \wedge \top \equiv F$ $F \vee \top \equiv \top$	Gesetze mit $\top$
$F \wedge \bot \equiv \bot$ $F \vee \bot \equiv F$	Gesetze mit $\bot$
$\neg\top \equiv \bot$ $\neg\bot \equiv \top$	

Alle diese Äquivalenzen können leicht mit Wahrheitswertetabellen überprüft werden.

Äquivalente Mengen von Formeln

Die Definition von Äquivalenz ist leicht auf Formelmengen erweiterbar:

Zwei Formelmengen $\mathcal{F}$ und $\mathcal{G}$ sind äquivalent, in Symbolen $\mathcal{F} \equiv \mathcal{G}$, wenn sie die selben Modelle haben.

Formelmengen verallgemeinern die Konjunktion, denn es gilt:

$$\{F_1, \ldots, F_n\} \equiv \{F_1 \wedge \ldots \wedge F_n\}$$

Vereinfachung: Dank Assoziativität verzichten wir ab jetzt in Formeln wie $(((F_1 \wedge F_2) \wedge F_3) \wedge F_4)$ auf Klammern (ebenso für $\vee$).

Allerdings dürfen Formelmengen auch unendlich sein (Konjunktionen dagegen nicht).

$\models$ entspricht $\rightarrow$ und $\equiv$ entspricht $\leftrightarrow$

Satz (Deduktionstheorem): Für jede Formelmenge $\mathcal{F}$ und Formeln G und H gilt $\mathcal{F} \models G \rightarrow H$ genau dann wenn $\mathcal{F} \cup \{G\} \models H$.

Beweis: Einfache Anwendung der Definitionen. ($\Rightarrow$) Wenn jedes Modell von $\mathcal{F}$ die Formel $G \rightarrow H$ erfüllt, dann muss jedes dieser Modelle, welches zudem G erfüllt, H erfüllen. ($\Leftarrow$) Wenn jedes Modell von $\mathcal{F}$, welches zudem G erfüllt, auch H erfüllt, dann muss jedes Modell von $\mathcal{F}$ auch $G \rightarrow H$ erfüllen. □

Notation: Wir schreiben $\models F$ statt $\emptyset \models F$ um auszudrücken, dass F allgemeingültig ist.

Satz: $F \equiv G$ genau dann wenn $\models F \leftrightarrow G$.

Beweis: $F \equiv G$ gdw. $F \models G$ und $G \models F$ gdw. $\models F \rightarrow G$ und $\models G \rightarrow F$ (wegen Deduktionstheorem) gdw. $\models F \leftrightarrow G$. □

Curry

Mit Deduktionstheorem und dem Bezug von Formelmengen und Konjunktionen erhält man ein weiteres interessantes Ergebnis:

$$\models (F \wedge G) \to H \text{ gdw. } \{F \wedge G\} \models H \text{ gdw. } \{F, G\} \models H$$
$$\text{gdw. } \{F\} \models G \to H \text{ gdw. } \models F \to (G \to H)$$

Tatsächlich gilt allgemein:

$$(F \wedge G) \to H \equiv F \to (G \to H)$$

Die Ersetzung von $(F \wedge G) \to H$ durch $F \to (G \to H)$ wird Currying (im Deutschen manchmal auch: Schönfinkeln) genannt
$\rightsquigarrow$ vgl. funktionale Programmierung

Aussagenlogisch Schließen in der Praxis

SAT Solver

Programme zum aussagenlogischen Schließen heißen SAT Solver.

- Hochoptimierte Systeme treten auf der jährlichen SAT-Competition gegeneinander an (`https://satcompetition.github.io/`)
- Praxissysteme sind teils weniger extrem optimiert aber anwendungsfreundlicher oder vielseitiger
- Alle relevanten Programmiersprachen sollten geeignete Bibliotheken anbieten, zumeist frei und open source
- Oft mit weiteren Features (nicht-boolesche Datentypen, weiche Optimierungskriterien, bestimmte prädikatenlogische Features, usw.)

$\rightsquigarrow$ wir werden mit Z3 in Python arbeiten

Der SMT-Solver Z3

Z3 ist ein leistungsfähiger Theorembeweiser

- Typ SMT-Solver (insbesondere vollständig & korrekt für Aussagenlogik)
- Ursprüngliches Anwendungsgebiet Softwareverifikation; inzwischen vielfältig eingesetzt
- Frei und OSS (MIT-Lizenz), entwickelt von Microsoft Research seit 2012
- C++, viele Sprachbindungen (Python, JavaScript, C, Rust, Java, ...) sowie Unterstützung des Austauschformats SMTLIB2

```
pip install z3-solver
```

Dokumentation: `https://microsoft.github.io/z3guide/`
(→Programming Z3→Python)

Z3 in Python

```
from z3 import *

p,q,r = Bools('p q r')

s = Solver()
s.add(And(p, Or(Not(q), r)))

print(s.check())
print(s.model())
```

```
sat
[p = True, q = False, r = False]
```

Zusammenfassung und Ausblick

Die Modelle einer Formel(menge) definieren ihre logischen Konsequenzen (=„alles, was in diesen Fällen noch gilt“)

In der Aussagenlogik gelten viele nützliche Äquivalenzen die man für Umformungen ausnutzen kann

Es gibt viele freie Softwarepakete zum aussagenlogischen Schließen, zum Beispiel Z3

Offene Fragen:

- Wie funktioniert logisches Schließen ohne Wahrheitswertetabellen?
- Wie (in)effizient ist logisches Schließen?
- Was hat das mit Komplexität zu tun?