Vorlesung
Logik in der Informatik

Wintersemester 2018/19

Dr. Christoph Berkholz

Lehrstuhl Logik in der Informatik
Institut für Informatik
Humboldt-Universität zu Berlin

Kapitel 1:

Einleitung

Abschnitt 1.1:
Von der Bibel bis zu den Simpsons

Logik

- altgriechisch „logos“: Vernunft
- die Lehre des vernünftigen Schlussfolgerns
- Teilgebiet u.a. der Disziplinen Philosophie, Mathematik und Informatik
- zentrale Frage:

  Wie kann man Aussagen miteinander verknüpfen, und auf welche Weise kann man formal Schlüsse ziehen und Beweise durchführen?

Das Lügnerparadoxon von Epimenides

Brief des Paulus an Titus 1:12-13:

> *Es hat einer von ihnen gesagt, ihr eigener Prophet:*
> *Die Kreter sind immer Lügner, böse Tiere und faule Bäuche.*

Angenommen, die Aussage des Propheten ist wahr.

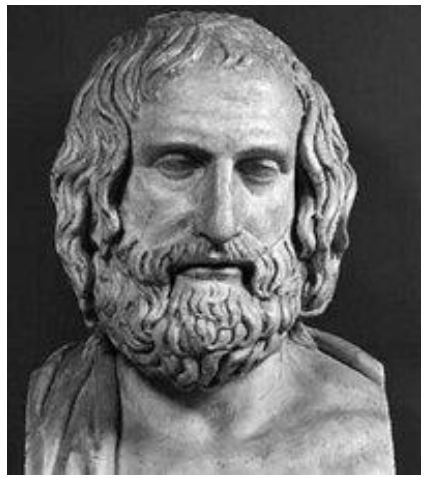
Da der Prophet selbst Kreter ist, lügt er also immer (und ist ein böses Tier und ein fauler Bauch). Dann hat er aber insbesondere in dem Satz *„Die Kreter sind immer Lügner, böse Tiere und faule Bäuche“* gelogen. D.h. die Aussage des Propheten ist *nicht* wahr. Dies ist ein Widerspruch!

Angenommen, die Aussage des Propheten ist falsch.

Dann gibt es Kreter, die nicht immer Lügner, böse Tiere und faule Bäuche sind. Dies stellt keinen Widerspruch dar.

Insgesamt wissen wir also, dass der Prophet in seiner obigen Aussage nicht die Wahrheit gesagt hat.

Protagoras und sein Student Euthalus vor Gericht



Protagoras (490 – 420 v.Chr.)

Quelle: `http://www.greatthoughtstreasury.com/author/protagoras`

Euthalus studierte die Kunst der Argumentation beim Meister Protagoras, um Anwalt zu werden.
Er vereinbart mit Protagoras, die Gebühren für den Unterricht zu bezahlen, sobald er seinen ersten Prozess gewonnen hat.

Aber dann zögert Euthalus seine Anwaltstätigkeit immer weiter hinaus, und schließlich beschließt Protagoras, seine Gebühren einzuklagen.
Euthalus verteidigt sich selbst . . .

Protagoras denkt:

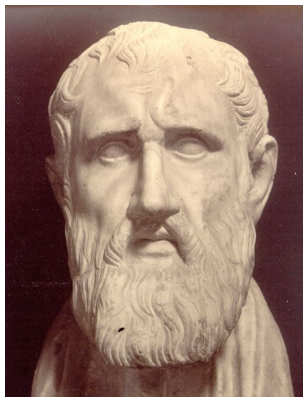
Wenn ich den Prozess gewinne, muss Euthalus gemäß Gerichtsbeschluss zahlen.
Wenn ich den Prozess verliere, muss Euthalus gemäß unserer Vereinbarung zahlen, da er dann seinen ersten Prozess gewonnen hat.

Euthalus denkt:

Wenn ich den Prozess gewinne, muss ich gemäß Gerichtsbeschluss nicht zahlen.
Wenn ich den Prozess verliere, muss ich gemäß unserer Vereinbarung nicht zahlen.

Achilles und die Schildkröte

Achilles und die Schildkröte laufen ein Wettrennen. Achilles gewährt der Schildkröte einen Vorsprung. Zenon behauptet, dass Achilles die Schildkröte niemals einholen kann.



Zenon von Elea (490 – 425 v.Chr.) *Quelle:* `http://aefucr.blogspot.de/2008/04/resolucin-de-la-paradoja-de-zenn-de.html`

Zenons Begründung: Zu dem Zeitpunkt, an dem Achilles den Startpunkt der Schildkröte erreicht, ist die Schildkröte schon ein Stück weiter. Etwas später erreicht Achilles diesen Punkt, aber die Schildkröte ist schon etwas weiter. Wenn Achilles diesen Punkt erreicht, ist die Schildkröte wieder etwas weiter. So kann Achilles zwar immer näher an die Schildkröte herankommen, sie aber niemals einholen.

Auflösung durch die Infinitesimalrechnung:

Gottfried Wilhelm von Leibniz (1646 – 1716)
und Isaac Newton (1643 – 1727)

Quelle: `http://www-history.mcs.st-and.ac.uk/PictDisplay/Leibniz.html`
und *Quelle:* `http://de.wikipedia.org/wiki/Isaac_Newton`

Der Barbier von Sonnenthal

Im Städtchen Sonnenthal (in dem bekanntlich viele seltsame Dinge passieren) wohnt ein Barbier, der genau diejenigen männlichen Einwohner von Sonnenthal rasiert, die sich nicht selbst rasieren.

Frage: Rasiert der Barbier sich selbst?

Angenommen, der Barbier rasiert sich selbst.

Da er ein männlicher Einwohner von Sonnenthal ist, der sich selbst rasiert, wird er *nicht* vom Barbier rasiert. Aber er selbst ist der Barbier. Dies ist ein Widerspruch!

Angenommen, der Barbier rasiert sich nicht selbst.

Da er in Sonnenthal wohnt und dort alle Einwohner rasiert, die sich nicht selbst rasieren, muss er sich rasieren. Dies ist ein Widerspruch!

Die Anfänge der formalen Logik

Aristoteles' Syllogismen

Die folgende Schlussweise ist aus rein formalen Gründen korrekt.

Annahme 1:	Alle Menschen sind sterblich.
Annahme 2:	Sokrates ist ein Mensch.
Folgerung:	Also ist Sokrates sterblich.

Diese Art von Schluss und eine Reihe verwandter Schlussweisen nennt man Syllogismen.

Annahme 1:	Alle A sind B.
Annahme 2:	C ist ein A.
Folgerung:	Also ist C B.

Beispiele

Annahme 1:	Alle Borg sind assimiliert worden.
Annahme 2:	Seven of Nine ist eine Borg.
Folgerung:	Also ist Seven of Nine assimiliert worden.

Annahme 1:	Alle Substitutionschiffren sind anfällig gegen Brute-Force-Angriffe.
Annahme 2:	Der Julius-Cäsar-Chiffre ist ein Substitutionschiffre.
Folgerung:	Also ist der Julius-Cäsar-Chiffre anfällig gegen Brute-Force-Angriffe.

Aristoteles (384 - 322 v.Chr.)

Quelle: `http://de.wikipedia.org/wiki/Aristoteles`

Charles Lutwidge Dodgson a.k.a. Lewis Carroll (1838 – 1898)
Quelle: `http://en.wikiquote.org/wiki/Lewis_Carroll`

"Contrariwise," continued Tweedledee, "if it was so, it might be; and if it were so, it would be; but as it isn't, it ain't. That's logic."

aus: *Alice in Wonderland*

Carrolls formaler Schluss

Annahme 1:	Es gibt keine Schweine, die fliegen können.
Annahme 2:	Alle Schweine sind gefräßige Tiere.
Annahme 3:	Es gibt Schweine.
Folgerung:	Also gibt es gefräßige Tiere, die nicht fliegen können.

Die Form des Schlusses ist:

Annahme 1:	Es gibt keine A, die B (sind).
Annahme 2:	Alle A sind C.
Annahme 3:	Es gibt A.
Folgerung:	Also gibt es C, die nicht B (sind).

Nicht jeder formale Schluss ist korrekt

Annahme 1:	Es gibt Vögel, die fliegen können.
Annahme 2:	Es gibt keine fliegenden (Tiere), die Klavier spielen können.
Folgerung:	Also gibt es keine Vögel, die Klavier spielen können.

Kein korrekter Schluss, auch wenn in diesem Fall die Folgerung wahr ist.

Der folgende, offensichtlich falsche, Schluss hat dieselbe Form:

Annahme 1:	Es gibt Menschen, die stumm sind.
Annahme 2:	Es gibt keine stummen (Lebewesen), die sprechen können.
Folgerung:	Also gibt es keine Menschen, die sprechen können.

Aber wie merkt man es?

Man kann einen falschen Schluss entlarven, indem man einen formal gleichen Schluss findet, der klar falsch ist.

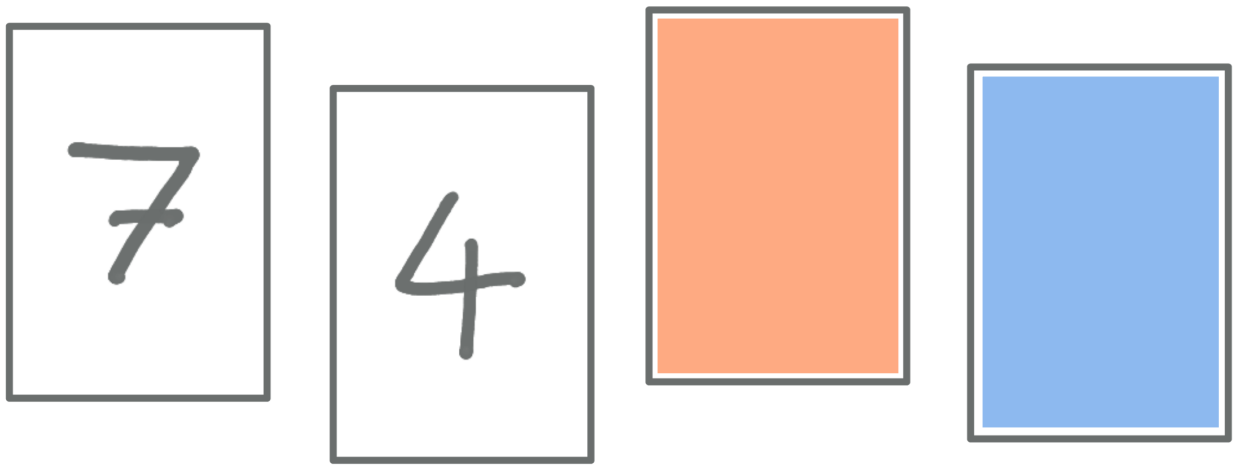
Annahme 1:	Erbeeren schmecken gut.
Annahme 2:	Schlagsahne schmeckt gut.
Folgerung:	Also schmecken Erdbeeren mit Schlagsahne gut.

Aber:

Annahme 1:	Pizza schmeckt gut.
Annahme 2:	Schlagsahne schmeckt gut.
Folgerung:	Also schmeckt Pizza mit Schlagsahne gut.

Wasons Auswahlaufgabe (Wason's selection task)

Uns stehen vier Karten der folgenden Art zur Verfügung:
Auf jeder Karte steht auf der Vorderseite eine Ziffer zwischen 0 und 9. Die Rückseite jeder Karte ist komplett rot oder komplett blau.
Wir sehen Folgendes:

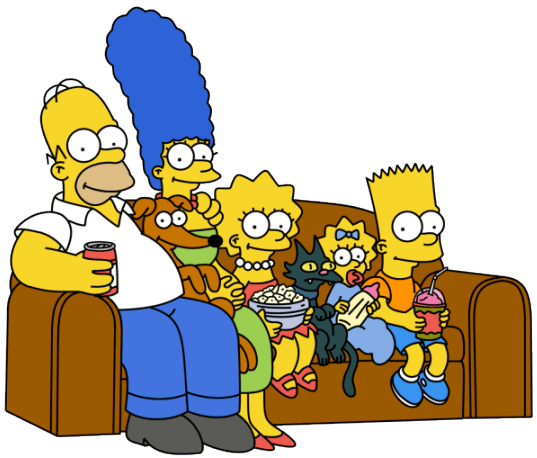


Jemand hat folgende **Hypothese** aufgestellt:

> Wenn auf der Vorderseite eine gerade Zahl steht,
> dann ist die Rückseite rot.

Welche Karte(n) müssen Sie umdrehen, um zu überprüfen, ob die Hypothese stimmt? Übermitteln Sie Ihre Lösung jetzt hier: `http://pingo.upb.de/160267`

Und was sagen die Simpsons?



Quelle: `http://en.wikipedia.org/wiki/Simpson_family`

Homer: Not a bear in sight. The Bear Patrol must be working like a charm.

Lisa: That's specious reasoning, Dad.

Homer: Thank you, dear.

Lisa: By your logic I could claim that this rock keeps tigers away.

Homer: Oh, how does it work?

Lisa: It doesn't work.

Homer: Uh-huh.

Lisa: It's just a stupid rock.

Homer: Uh-huh.

Lisa: But I don't see any tigers around, do you?
(Pause)

Homer: Lisa, I want to buy your rock.

[Lisa refuses at first, then takes the exchange]

Abschnitt 1.2:

Logik in der Informatik

Die Rolle der Logik in der Informatik

Halpern, Harper, Immerman, Kolaitis, Vardi, Vianu (2001):

> *Concepts and methods of logic occupy a central place in computer science, insomuch that logic has been called "the calculus of computer science".*

aus: *On the unusual effectiveness of logic in computer science*, Bulletin of Symbolic Logic 7(2): 213-236 (2001)

Anwendungsbereiche der Logik in der Informatik

- Repräsentation von Wissen (z.B. im Bereich der künstlichen Intelligenz) [siehe Kapitel 2 und 3]
- Grundlage für Datenbank-Anfragesprachen [siehe Kapitel 3]
- Bestandteil von Programmiersprachen (z.B. um Bedingungen in IF-Anweisungen zu formulieren) [siehe Kapitel 2]
- automatische Generierung von Beweisen (so genannte *Theorembeweiser*) [siehe Kapitel 4]
- Berechenbarkeits- und Komplexitätstheorie
- Verifikation von
  - Schaltkreisen (*Ziel:* beweise, dass ein Schaltkreis bzw. Chip „richtig“ funktioniert)
  - Programmen (*Ziel:* beweise, dass ein Programm gewisse wünschenswerte Eigenschaften hat)
  - Protokollen (*Ziel:* beweise, dass die Kommunikation zwischen zwei „Agenten“, die nach einem gewissen Protokoll abläuft, „sicher“ ist)
- Logik-Programmierung [siehe folgende Folien und Kapitel 5]

Einführung in die Logik-Programmierung

„Was“ statt „Wie“ am Beispiel von Tiramisu

Tiramisu — Deklarativ

Aus Eigelb, Mascarpone und in Likör und Kaffee getränkten Biskuits hergestellte cremige Süßspeise

(aus: DUDEN, Fremdwörterbuch, 6. Auflage)

Tiramisu — Imperativ

1/4 l Milch mit 2 EL Kakao und 2 EL Zucker aufkochen. 1/4 l starken Kaffee und 4 EL Amaretto dazugeben.

5 Eigelb mit 75 g Zucker weißschaumig rühren, dann 500 g Mascarpone dazumischen.

ca 200 g Löffelbiskuit.

Eine Lage Löffelbiskuit in eine Auflaufform legen, mit der Flüssigkeit tränken und mit der Creme überziehen. Dann wieder Löffelbiskuit darauflegen, mit der restlichen Flüssigkeit tränken und mit der restlichen Creme überziehen.

Über Nacht im Kühlschrank durchziehen lassen und vor dem Servieren mit Kakao bestäuben.

(aus: Gisela Schweikardt, handschriftliche Kochrezepte)

Der große Traum der Informatik

Imperative Vorgehensweise:

Beschreibung, wie das gewünschte Ergebnis erzeugt wird . . . . . „Wie“

Deklarative Vorgehensweise:

Beschreibung der Eigenschaften des gewünschten Ergebnisses . . . . . „Was“

Traum der Informatik:

Möglichst wenig „wie“, möglichst viel „was“

D.h.: Automatische Generierung eines Ergebnisses aus seiner Spezifikation

Realität:

Datenbanken: Deklarative Anfragesprache ist Industriestandard (SQL)

Software-Entwicklung: Generierungs-Tools

Programmiersprachen: Logik-Programmierung, insbes. Prolog

ABER: Imperativer Ansatz überwiegt in der Praxis

Logik-Programmierung

- Logik-Programmierung bezeichnet die Idee, Logik direkt als Programmiersprache zu verwenden.

- Logik-Programmierung (in Sprachen wie Prolog) und die verwandte funktionale Programmierung (in Sprachen wie LISP, ML, Haskell) sind deklarativ,
  im Gegensatz zur imperativen Programmierung (in Sprachen wie Java, C, Perl).

- Die Idee er deklarativen Programmierung besteht darin, dem Computer lediglich sein Wissen über das Anwendungsszenario und sein Ziel mitzuteilen und dann die Lösung des Problems dem Computer zu überlassen.

  Bei der imperativen Programmierung hingegen gibt man dem Computer die einzelnen Schritte zur Lösung des Problems vor.

Prolog

- Prolog
  - ist die wichtigste logische Programmiersprache,
  - geht zurück auf Kowalski und Colmerauer (Anfang der 1970er Jahre, Marseilles),
  - steht für (franz.) Programmation en logique.
  - Mitte/Ende der 1970er Jahre: effiziente Prolog-Implementierung durch den von Warren (in Edinburgh) entwickelten Prolog-10 Compiler.

- Aus Effizienzgründen werden in Prolog die abstrakten Ideen der logischen Programmierung nicht in Reinform umgesetzt, Prolog hat auch „nichtlogische“ Elemente.

- Prolog ist eine voll entwickelte und mächtige Programmiersprache, die vor allem für symbolische Berechnungsprobleme geeignet ist.

Anwendungen

Die wichtigsten Anwendungsgebiete sind die künstliche Intelligenz und die Computerlinguistik.

Beispiele

Das Interface für natürliche Sprache

- in der International Space Station wurde von der NASA
- beim IBM Watson System, das in 2011 die Jeopardy! Man vs. Machine Challenge gewonnen hat, wurde

in Prolog implementiert.

Mehr Informationen dazu finden sich z.B. unter
`https://sicstus.sics.se/customers.html` und
`http://www.cs.nmsu.edu/ALP/2011/03/natural-language-processing-with-prolog-in-the-ibm-watson-system/`

Learn Prolog Now!

Im Rahmen der Übungsaufgaben zur Vorlesung werden wir jede Woche eins der 12 Kapitel des Buchs

> „Learn Prolog Now!“ von Patrick Blackburn, Johan Bos und Kristina Striegnitz (Kings College Publications, 2006)
>
> ... auch erhältlich als Online-Kurs unter
> `http://www.learnprolognow.org`

durcharbeiten.
Als Unterstützung dazu gibt es jede Woche eine 2-stündige Prolog-Übung.

Am Ende des Semesters, in Kapitel 5, werden wir von Prolog abstrahieren und uns die Grundprinzipien der Logik-Programmierung anschauen.

Lernziele und Semesterausblick

Lernziele

Mein teurer Freund, ich rat Euch drum
Zuerst Collegium Logicum.
Da wird der Geist Euch wohl dressiert,
In spanische Stiefeln eingeschnürt,
Daß er bedächtiger so fortan
Hinschleiche die Gedankenbahn,
Und nicht etwa, die Kreuz und Quer,
Irrlichteliere hin und her.

Mephistopheles in *Faust*

Aus der Studienordnung:

- Studierende erlangen die Fähigkeit, Sachverhalte in geeigneten formalen Systemen zu formalisieren und die grundlegenden Begriffe und Ergebnisse der mathematischen Logik zu verstehen und anzuwenden.
- Darüber hinaus erlernen sie anhand der deklarativen Programmiersprache Prolog ein neues Programmierparadigma.

Semesterüberblick

1. Einleitung — heute
2. Aussagenlogik — Woche 1–6
   Syntax und Semantik, Normalformen, Modellierung, Resolution, Erfüllbarkeitsalgorithmen
3. Logik erster Stufe — Woche 7–10
   Syntax und Semantik, Normalformen, Modellierung, Nichtausdrückbarkeit
4. Grundlagen des automatischen Schließens — Woche 11–14
   Sequenzenkalkül, Vollständigkeits- und Endlichkeitssatz, Grenzen der Berechenbarkeit, automatische Theorembeweiser
5. Logik-Programmierung — Woche 15–16
   theoretische Grundlagen der Logik-Programmierung

Learn Prolog Now! Einführung in Prolog findet semesterbegleitend statt.

Kapitel 2:

Aussagenlogik

Abschnitt 2.1:

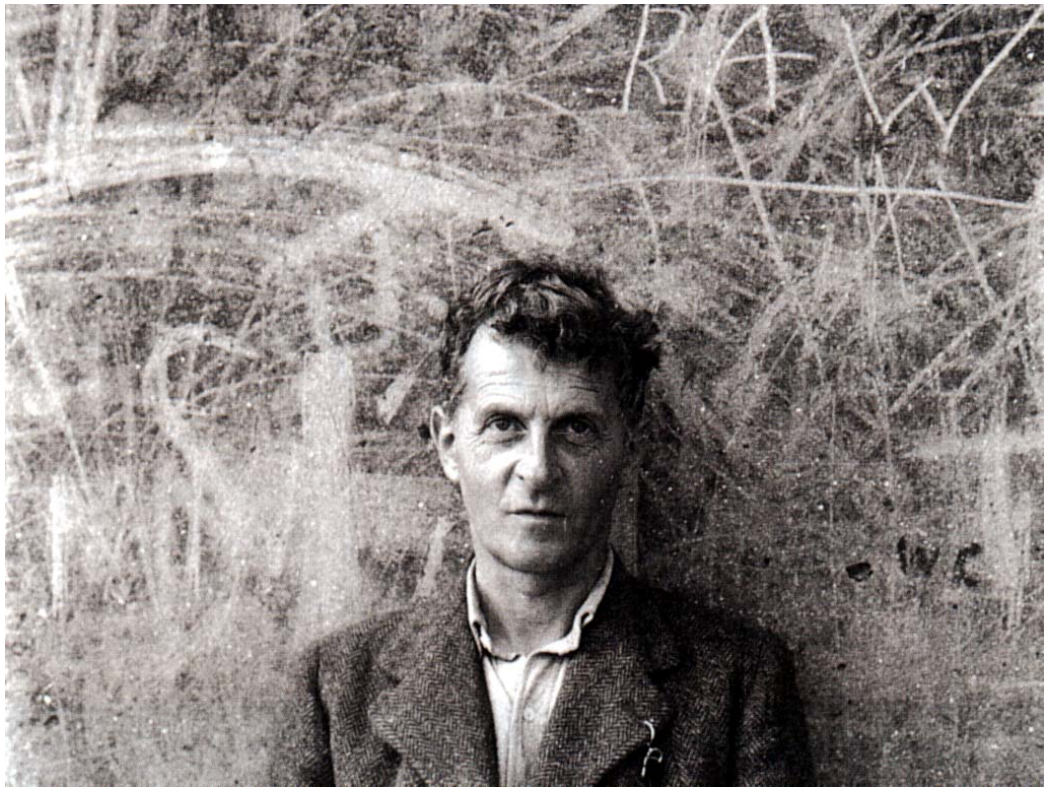
Syntax und Semantik

Aussagen

Die Frage „Was ist eigentlich ein Wort?“ ist analog der „Was ist eine Schachfigur?“

Ludwig Wittgenstein, *Philosophische Untersuchungen*

- Aussagen (im Sinne der Aussagenlogik) sind sprachliche Gebilde, die entweder wahr oder falsch sind.
- Aussagen können mit Junktoren wie nicht, und, oder oder wenn . . . dann zu komplexeren Aussagen verknüpft werden.
- Aussagenlogik beschäftigt sich mit allgemeinen Prinzipien des korrekten Argumentierens und Schließens mit Aussagen und Kombinationen von Aussagen.



Ludwig Wittgenstein (1889 – 1951)

Quelle: `http://en.wikipedia.org/wiki/Ludwig_Wittgenstein`

Beispiel 2.1 (Geburtstagsfeier)

Fred möchte mit möglichst vielen seiner Freunde Anne, Bernd, Christine, Dirk und Eva seinen Geburtstag feiern. Er weiß Folgendes:
Wenn Bernd und Anne beide zur Party kommen, dann wird Eva auf keinen Fall kommen. Und Dirk wird auf keinen Fall kommen, wenn Bernd und Eva beide zur Feier kommen. Aber Eva kommt allenfalls dann, wenn Christine und Dirk kommen. Andererseits kommt Christine nur dann, wenn auch Anne kommt. Anne wiederum wird nur dann kommen, wenn auch Bernd oder Christine dabei sind.

Frage: Wie viele Freunde (und welche) werden im besten Fall zur Party kommen?

Das Wissen, das in dem Text wiedergegeben ist, lässt sich in „atomare Aussagen“ zerlegen, die mit Junktoren verknüpft werden können.
Die atomaren Aussagen, um die sich der Text dreht, kürzen wir folgendermaßen ab:

A	:	Anne kommt zur Feier
B	:	Bernd kommt zur Feier
C	:	Christine kommt zur Feier
D	:	Dirk kommt zur Feier
E	:	Eva kommt zur Feier

Das im Text zusammengefasste Wissen lässt sich wie folgt repräsentieren.

(1) Wenn Bernd und Anne beide zur Party kommen, dann wird Eva auf keinen Fall kommen.

kurz: Wenn (B und A), dann nicht E *kürzer:* $(B \wedge A) \rightarrow \neg E$

(2) Dirk wird auf keinen Fall kommen, wenn Bernd und Eva beide zur Feier kommen.

kurz: Wenn (B und E), dann nicht D *kürzer:* $(B \wedge E) \rightarrow \neg D$

(3) Eva kommt allenfalls dann, wenn Christine und Dirk kommen.

kurz: Wenn E, dann (C und D) *kürzer:* $E \rightarrow (C \wedge D)$

(4) Christine kommt nur dann, wenn auch Anne kommt.

kurz: Wenn C, dann A *kürzer:* $C \rightarrow A$

(5) Anne kommt nur dann, wenn auch Bernd oder Christine dabei sind.

kurz: Wenn A, dann (B oder C) *kürzer:* $A \rightarrow (B \vee C)$

Fallstricke natürlichsprachlicher Aussagen

Die Verwendung der Wörter und, wenn . . . dann, oder, nicht in der Alltagssprache entspricht nicht immer exakt unseren logischen Junktoren.

(1) Anne hat mit dem Kaffeetrinken aufgehört.

kurz: V und nicht G *kürzer:* $V \wedge \neg G$

(2) Anne hat nicht mit dem Kaffeetrinken aufgehört.

kurz: V und G *kürzer:* $V \wedge G$

Ist (2) die Negation von (1)? In dem Fall, dass Anne noch nie Kaffee getrunken hat, ist keine der beiden Aussagen wahr.

V : Anne war in der Vergangenheit Kaffeetrinkerin.

G : Anne ist zur Zeit Kaffeetrinkerin.

Zwei weitere Beispiele:

- Ich werde mir ein rotes oder ein blaues Fahrrad kaufen.
- Wenn Regen vorhergesagt ist, dann nehme ich einen Schirm mit.

Syntax und Semantik

Syntax: legt fest, welche Zeichenketten Formeln der Aussagenlogik sind

Semantik: legt fest, welche „Bedeutung“ einzelne Formeln haben

Dies ist analog zur Syntax und Semantik von Java-Programmen:

Die Syntax legt fest, welche Zeichenketten Java-Programme sind, während die Semantik bestimmt, was das Programm tut.

Zur Verdeutlichung werden wir im Folgenden syntaktische Objekte oft in orange darstellen, während wir semantische Aussagen in grün angeben.

Syntax der Aussagenlogik

Notationen

- Die Menge $\mathbb{N}$ der natürlichen Zahlen besteht aus allen nicht-negativen ganzen Zahlen, d.h.

$$\mathbb{N} := \{ 0, 1, 2, 3, \ldots \}.$$

- Für ein $n \in \mathbb{N}$ ist

$$[n] := \{1, \ldots, n\} = \{ i \in \mathbb{N} : 1 \leqslant i \leqslant n \}.$$

Definition 2.2

Ein Aussagensymbol (oder eine Aussagenvariable, kurz: Variable) hat die Form A_i für ein $i \in \mathbb{N}$.
Die Menge aller Aussagensymbole bezeichnen wir mit AS, d.h.

$$\mathsf{AS} \;=\; \{A_i \;:\; i \in \mathbb{N}\} \;=\; \{A_0,\; A_1,\; A_2,\; A_3,\; \ldots\}$$

Aussagenlogische Formeln sind Wörter, die über dem folgenden Alphabet gebildet sind.

Definition 2.3

Das Alphabet der Aussagenlogik besteht aus

- den Aussagesymbolen in AS,
- den Junktoren $\neg$, $\wedge$, $\vee$, $\rightarrow$,
- den booleschen Konstanten $\mathbf{0}$, $\mathbf{1}$,
- den Klammersymbolen $($, $)$.

Wir schreiben A_{AL}, um das Alphabet der Aussagenlogik zu bezeichnen, d.h.

$$A_{\mathsf{AL}} \;:=\; \mathsf{AS} \;\cup\; \{\, \neg,\; \wedge,\; \vee,\; \rightarrow,\; \mathbf{0},\; \mathbf{1},\; (,\;) \,\}$$

Definition 2.4 (Syntax der Aussagenlogik)

Die Menge AL der aussagenlogischen Formeln (kurz: Formeln) ist die folgendermaßen rekursiv definierte Teilmenge von A^*_{AL}:

Basisregeln: (zum Bilden der so genannten atomaren Formeln)

(B0) $\mathbf{0} \in \mathrm{AL}$

(B1) $\mathbf{1} \in \mathrm{AL}$

(BS) Für jedes Aussagensymbol $A_i \in \mathrm{AS}$ gilt: $A_i \in \mathrm{AL}$

Rekursive Regeln:

(R1) Ist $\varphi \in \mathrm{AL}$, so ist auch $\neg\varphi \in \mathrm{AL}$ (Negation)

(R2) Ist $\varphi \in \mathrm{AL}$ und $\psi \in \mathrm{AL}$, so ist auch

- $(\varphi \wedge \psi) \in \mathrm{AL}$ (Konjunktion)
- $(\varphi \vee \psi) \in \mathrm{AL}$ (Disjunktion)
- $(\varphi \to \psi) \in \mathrm{AL}$ (Implikation)

Beispiele

- $(\neg A_0 \vee (A_0 \to A_1)) \quad \in \mathsf{AL}$
- $\neg((A_0 \wedge \mathbf{0}) \to \neg A_3) \quad \in \mathsf{AL}$
- $A_1 \vee A_2 \wedge A_3 \quad \notin \mathsf{AL}$
- $(\neg A_1) \quad \notin \mathsf{AL}$

Griechische Buchstaben

In der Literatur werden Formeln einer Logik traditionell meistens mit griechischen Buchstaben bezeichnet.

Hier eine Liste der gebräuchlichsten Buchstaben:

Buchstabe	φ	ψ	χ	θ bzw. ϑ	λ	μ	ν	τ	κ
Aussprache	phi	psi	chi	theta	lambda	mü	nü	tau	kappa

Buchstabe	σ	ρ	ξ	ζ	α	β	γ	δ	ω
Aussprache	sigma	rho	xi	zeta	alpha	beta	gamma	delta	omega

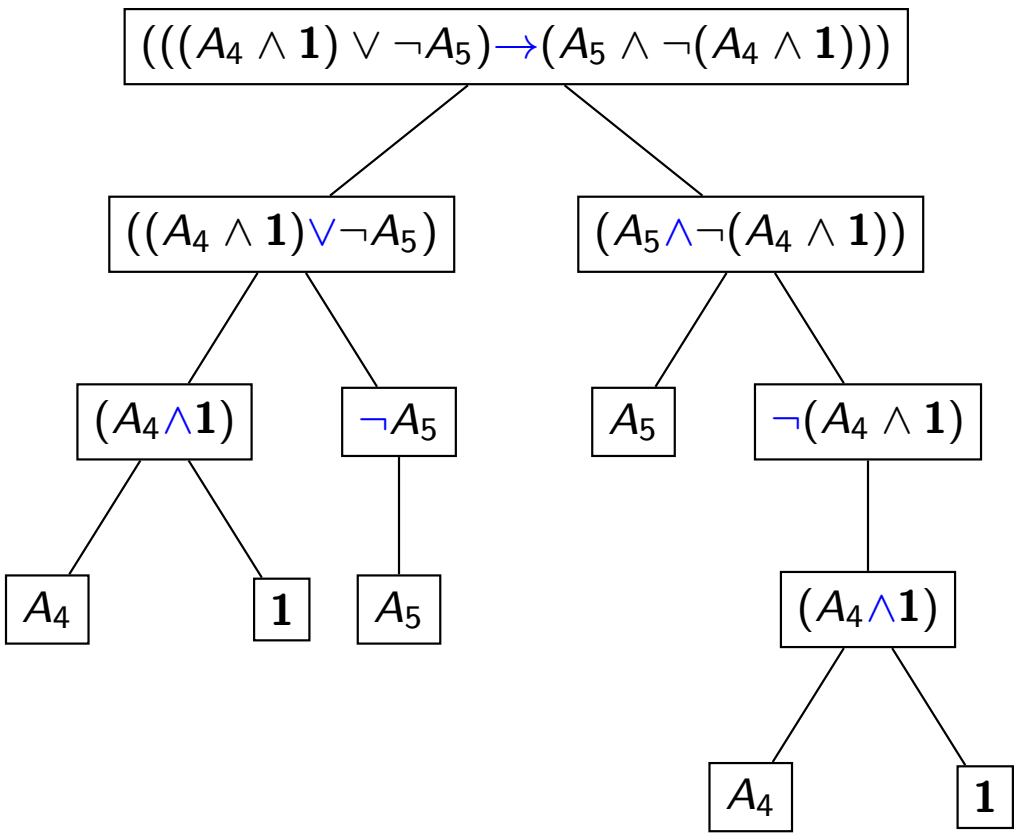
Buchstabe	ε	ι	π	Δ	Γ	Σ	Π	Φ
Aussprache	epsilon	iota	pi	Delta	Gamma	Sigma	Pi	Phi

Syntaxbäume

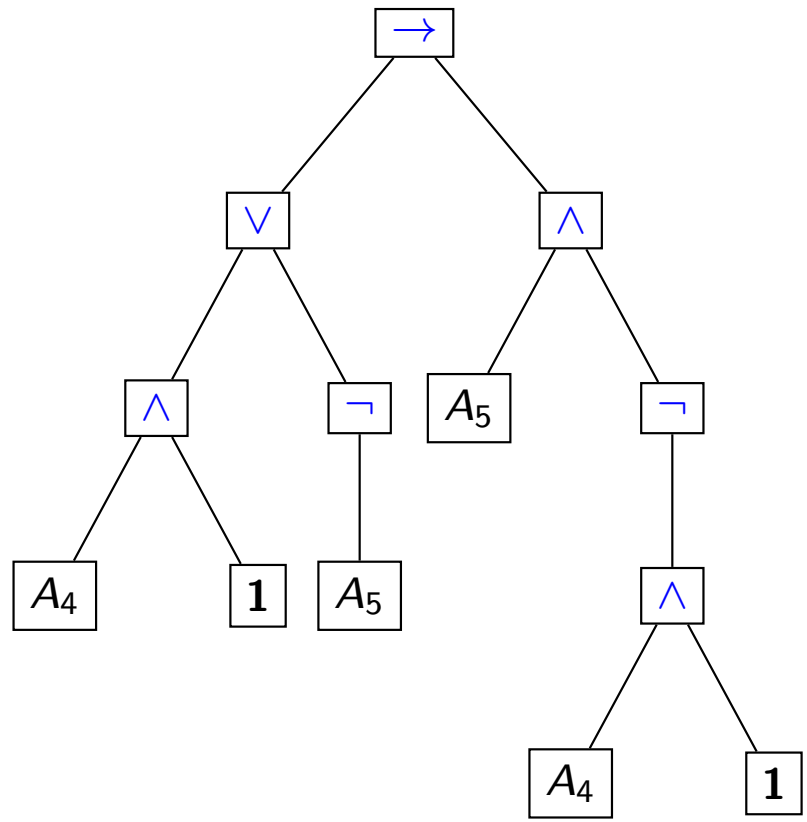
Die Struktur einer Formel lässt sich bequem in einem Syntaxbaum (englisch: parse tree) darstellen.

Beispiel: Syntaxbaum der Formel $(((A_4 \wedge \mathbf{1}) \vee \neg A_5) \rightarrow (A_5 \wedge \neg(A_4 \wedge \mathbf{1})))$

Ausführlich:



Kurzform:



Subformeln und eindeutige Lesbarkeit

- Jede Formel hat genau einen Syntaxbaum. Diese Aussage ist als das Lemma über die eindeutige Lesbarkeit aussagenlogischer Formeln bekannt.
- Die Formeln ψ, die im ausführlichen Syntaxbaum einer Formel φ als Knotenbeschriftung vorkommen, nennen wir Subformeln (bzw. Teilformeln) von φ.
- Eine Subformel ψ von φ kann an mehreren Knoten des Syntaxbaums vorkommen. Wir sprechen dann von verschiedenen Vorkommen von ψ in φ.

Semantik der Aussagenlogik

Vorüberlegung zur Semantik

- Eine aussagenlogische Formel wird erst zur Aussage, wenn wir alle in ihr vorkommenden Aussagensymbole durch Aussagen ersetzen.
- Wir interessieren uns hier nicht so sehr für die tatsächlichen Aussagen, sondern nur für ihren **Wahrheitswert**, also dafür, ob sie wahr oder falsch sind.
- Um das festzustellen, reicht es, den Aussagensymbolen die Wahrheitswerte der durch sie repräsentierten Aussagen zuzuordnen.
- Die Bedeutung einer Formel besteht also aus ihren Wahrheitswerten unter allen möglichen Wahrheitswerten für die in der Formel vorkommenden Aussagensymbole.

Interpretationen (d.h. Variablenbelegungen)

Wir repräsentieren die Wahrheitswerte wahr und falsch durch 1 und 0.

Definition 2.5

Eine aussagenlogische Interpretation (kurz: Interpretation oder Belegung) ist eine Abbildung

$$\mathcal{I} : \mathrm{AS} \to \{0, 1\}.$$

D.h.: $\mathcal{I}$ „belegt“ jedes Aussagensymbol $X \in \mathrm{AS}$ mit einem der beiden Wahrheitswerte 1 (für „wahr“) oder 0 (für „falsch“); und $\mathcal{I}(X)$ ist der Wahrheitswert, mit dem das Aussagensymbol X belegt wird.

Semantik der Aussagenlogik

Definition 2.6

Zu jeder Formel $\varphi \in \mathrm{AL}$ und jeder Interpretation $\mathcal{I}$ definieren wir einen Wahrheitswert $[\![\varphi]\!]^{\mathcal{I}}$ rekursiv wie folgt:

Rekursionsanfang:

- $[\![\mathbf{0}]\!]^{\mathcal{I}} := 0$.
- $[\![\mathbf{1}]\!]^{\mathcal{I}} := 1$.
- Für alle $X \in \mathrm{AS}$ gilt: $[\![X]\!]^{\mathcal{I}} := \mathcal{I}(X)$.

Rekursionsschritt:

- Ist $\varphi \in \mathrm{AL}$, so ist $[\![\neg\varphi]\!]^{\mathcal{I}} := \begin{cases} 1 & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = 0, \\ 0 & \text{sonst.} \end{cases}$

Semantik der Aussagenlogik (Fortsetzung)

- Ist $\varphi \in \mathsf{AL}$ und $\psi \in \mathsf{AL}$, so ist
  - $[\![(\varphi \wedge \psi)]\!]^{\mathcal{I}} := \begin{cases} 1 & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = [\![\psi]\!]^{\mathcal{I}} = 1, \\ 0 & \text{sonst.} \end{cases}$
  - $[\![(\varphi \vee \psi)]\!]^{\mathcal{I}} := \begin{cases} 0 & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = [\![\psi]\!]^{\mathcal{I}} = 0, \\ 1 & \text{sonst.} \end{cases}$
  - $[\![(\varphi \rightarrow \psi)]\!]^{\mathcal{I}} := \begin{cases} 0 & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = 1 \text{ und } [\![\psi]\!]^{\mathcal{I}} = 0, \\ 1 & \text{sonst.} \end{cases}$

Intuitive Bedeutung der Semantik

Boolesche Konstanten: **1** und **0** bedeuten einfach „wahr“ und „falsch“.

Aussagensymbole: Die Aussagensymbole stehen für irgendwelche Aussagen, von denen uns aber nur der Wahrheitswert interessiert. Dieser wird durch die Interpretation festgelegt.

Negation: $\neg\varphi$ bedeutet „nicht φ“.

Konjunktion: $(\varphi \wedge \psi)$ bedeutet „φ und ψ“.

Disjunktion: $(\varphi \vee \psi)$ bedeutet „φ oder ψ“.

Implikation: $(\varphi \to \psi)$ bedeutet „φ impliziert ψ“ (oder „wenn φ dann ψ“).

Rekursive Definitionen über Formeln

- Ähnlich wie Funktionen auf den natürlichen Zahlen, wie zum Beispiel die Fakultätsfunktion oder die Fibonacci Folge, können wir Funktionen auf den aussagenlogischen Formeln rekursiv definieren.
- Dabei gehen wir von den atomaren Formeln aus und definieren dann den Funktionswert einer zusammengesetzten Formel aus den Funktionswerten ihrer Bestandteile.
- Zur Rechtfertigung solcher Definitionen benötigt man die eindeutige Lesbarkeit aussagenlogischer Formeln, die besagt, dass sich jede Formel eindeutig in ihre Bestandteile zerlegen lässt.
- Wir haben auf diese Weise die Semantik definiert. Wir haben nämlich für jede Interpretation $\mathcal{I}$ rekursiv eine Funktion $[\![\cdot]\!]^{\mathcal{I}} : \mathsf{AL} \to \{0, 1\}$ definiert.

Rekursive Definitionen (Forts.)

Schematisch sieht die rekursive Definition einer Funktion $f : \mathrm{AL} \to M$ (für eine beliebige Menge M) folgendermaßen aus:

Rekursionsanfang:

- Definiere $f(\mathbf{0})$ und $f(\mathbf{1})$.
- Definiere $f(X)$ für alle $X \in \mathrm{AS}$.

Rekursionsschritt:

- Definiere $f(\neg\varphi)$ aus $f(\varphi)$.
- Definiere $f((\varphi \wedge \psi))$ aus $f(\varphi)$ und $f(\psi)$.
- Definiere $f((\varphi \vee \psi))$ aus $f(\varphi)$ und $f(\psi)$.
- Definiere $f((\varphi \to \psi))$ aus $f(\varphi)$ und $f(\psi)$.

Beispiel 2.7

Betrachte die Formel $\varphi := \big(\neg A_0 \vee (A_5 \to A_1) \big)$

und die Interpretation $\mathcal{I} : \mathsf{AS} \to \{0,1\}$ mit

$$\mathcal{I}(A_0) = 1, \quad \mathcal{I}(A_1) = 1, \quad \mathcal{I}(A_5) = 0$$

und $\mathcal{I}(Y) = 0$ für alle $Y \in \mathsf{AS} \setminus \{A_0, A_1, A_5\}$.

Der Wahrheitswert $[\![\varphi]\!]^{\mathcal{I}}$ ist der Wert

$$\begin{aligned}
[\![\varphi]\!]^{\mathcal{I}} &\overset{\text{Def. 2.6}}{=} \begin{cases} 0, & \text{falls } [\![\neg A_0]\!]^{\mathcal{I}} = 0 \text{ und } [\![(A_5 \to A_1)]\!]^{\mathcal{I}} = 0 \\ 1, & \text{sonst} \end{cases} \\
&\overset{\text{Def. 2.6}}{=} \begin{cases} 0, & \text{falls } [\![A_0]\!]^{\mathcal{I}} = 1 \text{ und } \big([\![A_5]\!]^{\mathcal{I}} = 1 \text{ und } [\![A_1]\!]^{\mathcal{I}} = 0\big) \\ 1, & \text{sonst} \end{cases} \\
&\overset{\text{Def. 2.6}}{=} \begin{cases} 0, & \text{falls } \mathcal{I}(A_0) = 1 \text{ und } \mathcal{I}(A_5) = 1 \text{ und } \mathcal{I}(A_1) = 0 \\ 1, & \text{sonst} \end{cases} \\
&= \quad 1 \quad \text{(denn gemäß obiger Wahl von } \mathcal{I} \text{ gilt } \mathcal{I}(A_5) = 0).
\end{aligned}$$

Alternative Art, den Wert $[\![\varphi]\!]^{\mathcal{I}}$ zu bestimmen

- Ersetze in φ jedes Aussagensymbol X durch seinen gemäß $\mathcal{I}$ festgelegten Wahrheitswert, d.h. durch den Wert $\mathcal{I}(X)$, und rechne dann den Wert des resultierenden booleschen Ausdrucks aus.

- Speziell für die Formel φ und die Interpretation $\mathcal{I}$ aus Beispiel 2.7 ergibt die Ersetzung der Aussagensymbole durch die gemäß $\mathcal{I}$ festgelegten Wahrheitswerte den booleschen Ausdruck

$$\big(\ \neg 1\ \vee\ (0 \to 1)\ \big).$$

- Ausrechnen von $\neg 1$ ergibt den Wert 0.
  Ausrechnen von $(0 \to 1)$ ergibt den Wert 1.

- Insgesamt erhalten wir also $(0 \vee 1)$, was sich zum Wert 1 errechnet.
  Somit erhalten wir, dass $[\![\varphi]\!]^{\mathcal{I}} = 1$ ist.

Die Modellbeziehung

Definition 2.8

(a) Eine Interpretation $\mathcal{I}$ erfüllt eine Formel $\varphi \in \mathrm{AL}$ (wir schreiben: $\mathcal{I} \models \varphi$), wenn $[\![\varphi]\!]^{\mathcal{I}} = 1$.

Wir schreiben kurz $\mathcal{I} \not\models \varphi$ um auszudrücken, dass $\mathcal{I}$ die Formel φ *nicht* erfüllt (d.h., es gilt $[\![\varphi]\!]^{\mathcal{I}} = 0$).

(b) Eine Interpretation $\mathcal{I}$ erfüllt eine Formelmenge $\Phi \subseteq \mathrm{AL}$ (wir schreiben: $\mathcal{I} \models \Phi$), wenn $\mathcal{I} \models \varphi$ für alle $\varphi \in \Phi$.

(c) Ein Modell einer Formel φ (bzw. einer Formelmenge Φ) ist eine Interpretation $\mathcal{I}$ mit $\mathcal{I} \models \varphi$ (bzw. $\mathcal{I} \models \Phi$).

Das Koinzidenzlemma

- Offensichtlich hängt der Wert $[\![\varphi]\!]^{\mathcal{I}}$ nur von den Werten $\mathcal{I}(X)$ der Aussagensymbole $X \in \mathrm{AS}$ ab, die auch in φ vorkommen.

  Diese Aussage ist als das Koinzidenzlemma der Aussagenlogik bekannt.

- Um $[\![\varphi]\!]^{\mathcal{I}}$ festzulegen, reicht es also, die Werte $\mathcal{I}(X)$ nur für diejenigen Aussagensymbole $X \in \mathrm{AS}$ anzugeben, die in φ vorkommen.

Vereinbarungen zu Interpretationen

- Statt der vollen Interpretation $\mathcal{I} : \mathrm{AS} \to \{0,1\}$ geben wir in der Regel nur endlich viele Werte $\mathcal{I}(X_1), \ldots, \mathcal{I}(X_n)$ an und legen fest, dass $\mathcal{I}(Y) := 0$ für alle $Y \in \mathrm{AS} \setminus \{X_1, \ldots, X_n\}$.

- In den Beispielen legen wir eine Interpretation oft durch eine Wertetabelle fest. Beispielsweise beschreibt die Tabelle

X	A_0	A_1	A_5
$\mathcal{I}(X)$	1	1	0

  die Interpretation $\mathcal{I}$ mit $\mathcal{I}(A_0) = \mathcal{I}(A_1) = 1$ und $\mathcal{I}(A_5) = 0$ und $\mathcal{I}(Y) = 0$ für alle $Y \in \mathrm{AS} \setminus \{A_0, A_1, A_5\}$.

- Wir schreiben $\varphi(X_1, \ldots, X_n)$, um anzudeuten, dass in φ nur Aussagensymbole aus der Menge $\{X_1, \ldots, X_n\}$ vorkommen.

  Für Wahrheitswerte $b_1, \ldots, b_n \in \{0,1\}$ schreiben wir dann $\varphi[b_1, \ldots, b_n]$ anstatt $[\![\varphi]\!]^{\mathcal{I}}$ für eine (bzw. alle) Interpretationen $\mathcal{I}$ mit $\mathcal{I}(X_i) = b_i$ für alle $i \in [n] := \{1, \ldots, n\}$.

Vereinbarungen

- Wir schreiben $(\varphi \leftrightarrow \psi)$ als Abkürzung für $((\varphi \to \psi) \wedge (\psi \to \varphi))$.
- Statt mit $A_0, A_1, A_2, \ldots$ bezeichnen wir Aussagensymbole auch oft mit $A, B, C, \ldots, X, Y, Z, \ldots$ oder mit Varianten wie $X', Y_1, \ldots$.
- Die äußeren Klammern einer Formel lassen wir manchmal weg und schreiben z.B. $(X \wedge Y) \to Z$ an Stelle des (formal korrekten) $((X \wedge Y) \to Z)$.
- Bezüglich Klammerung vereinbaren wir, dass $\neg$ am stärksten bindet, und dass $\wedge$ und $\vee$ stärker binden als $\to$.

  Wir können also z.B. $X \wedge \neg Y \to Z \vee X$ schreiben und meinen damit $((X \wedge \neg Y) \to (Z \vee X))$.

  Nicht schreiben können wir z.B. $X \wedge Y \vee Z$ (da wir nichts darüber vereinbart haben, wie fehlende Klammern hier zu setzen sind).

- Wir schreiben $\bigwedge_{i=1}^{n} \varphi_i$ bzw. $(\varphi_1 \wedge \ldots \wedge \varphi_n)$ an Stelle von

$$(\cdots((\varphi_1 \wedge \varphi_2) \wedge \varphi_3) \wedge \ldots \wedge \varphi_n)$$

und nutzen analoge Schreibweisen auch für „$\vee$“ an Stelle von „$\wedge$“.

- Ist M eine endliche Menge aussagenlogischer Formeln, so schreiben wir

$$\bigwedge_{\varphi \in M} \varphi$$

um die Formel $(\varphi_1 \wedge \cdots \wedge \varphi_n)$ zu bezeichnen, wobei $n = |M|$ die Anzahl der Formeln in M ist und $\varphi_1, \ldots, \varphi_n$ die Auflistung aller Formeln in M in lexikographischer Reihenfolge ist. Formeln sind hierbei Worte über dem Alphabet der Aussagenlogik, wobei die einzelnen Symbole dieses Alphabets folgendermaßen aufsteigend sortiert sind:

$$\mathbf{0},\ \mathbf{1},\ \neg,\ \wedge,\ \vee,\ \rightarrow,\ (,\ ),\ A_0,\ A_1,\ A_2,\ A_3,\ \ldots.$$

Die analoge Schreibweise nutzen wir auch für „$\vee$“ an Stelle von „$\wedge$“.

- Diese Schreibweisen werden wir manchmal auch kombinieren. Sind zum Beispiel $I = \{i_1, \ldots, i_m\}$ und $J = \{j_1, \ldots, j_n\}$ endliche Mengen und ist für jedes $i \in I$ und $j \in J$ eine Formel $\varphi_{i,j}$ gegeben, so schreiben wir

$$\bigwedge_{i \in I} \bigvee_{j \in J} \varphi_{i,j}$$

um die Formel $(\psi_{i_1} \wedge \cdots \wedge \psi_{i_m})$ zu bezeichnen, wobei für jedes $i \in I$ die Formel ψ_i durch $\psi_i := (\varphi_{i,j_1} \vee \cdots \vee \varphi_{i,j_n})$ definiert ist.

Wahrheitstafeln

Für jede Formel $\varphi(X_1, \ldots, X_n)$ kann man die Wahrheitswerte unter allen möglichen Interpretationen in einer Wahrheitstafel darstellen. Für alle $(b_1, \ldots, b_n) \in \{0,1\}^n$ enthält die Tafel eine Zeile mit den Werten
$b_1 \ \cdots \ b_n \ \mid \ \varphi[b_1, \ldots, b_n]$.
Um die Wahrheitstafel für φ auszufüllen, ist es bequem, auch Spalten für (alle oder einige) Subformeln von φ einzufügen.

Beispiel: Wahrheitstafel für die Formel $\varphi(X, Y, Z) := X \vee Y \to X \wedge Z$:

X	Y	Z	$X \vee Y$	$X \wedge Z$	φ
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	0	0
1	1	1	1	1	1

Die Reihenfolge der Zeilen ist dabei unerheblich. Wir vereinbaren allerdings, die Zeilen stets so anzuordnen, dass die Werte $b_1 \cdots b_n \in \{0,1\}^n$, aufgefasst als Binärzahlen, in aufsteigender Reihenfolge aufgelistet werden.

Wahrheitstafeln für die Junktoren

Die Bedeutung der Junktoren kann man mittels ihrer Wahrheitstafeln beschreiben:

X	$\neg X$
0	1
1	0

X	Y	$X \wedge Y$
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	$X \vee Y$
0	0	0
0	1	1
1	0	1
1	1	1

X	Y	$X \rightarrow Y$
0	0	1
0	1	1
1	0	0
1	1	1

Genauso kann man eine Wahrheitstafel für die Formel $X \leftrightarrow Y$, die ja eine Abkürzung für $(X \rightarrow Y) \wedge (Y \rightarrow X)$ ist, bestimmen:

X	Y	$X \leftrightarrow Y$
0	0	1
0	1	0
1	0	0
1	1	1

$X \leftrightarrow Y$ bedeutet also „X genau dann wenn Y“.

Ein Logikrätsel

Beispiel 2.9

Auf der Insel Wafa leben zwei Stämme: Die Was, die immer die Wahrheit sagen, und die Fas, die immer lügen. Ein Reisender besucht die Insel und trifft auf drei Einwohner A, B, C, die ihm Folgendes erzählen:

- A sagt:
  „*B und C sagen genau dann die Wahrheit, wenn C die Wahrheit sagt.*“
- B sagt:
  „*Wenn A und C die Wahrheit sagen, dann ist es nicht der Fall, dass A die Wahrheit sagt, wenn B und C die Wahrheit sagen.*“
- C sagt:
  „*B lügt genau dann, wenn A oder B die Wahrheit sagen.*“

Frage: Welchen Stämmen gehören A, B und C an?

Aussagenlogische Modellierung

Aussagensymbole:

- W_A steht für „A sagt die Wahrheit.“
- W_B steht für „B sagt die Wahrheit.“
- W_C steht für „C sagt die Wahrheit.“

Aussagen der drei Inselbewohner:

- $\varphi_A := (W_B \wedge W_C) \leftrightarrow W_C$
- $\varphi_B := (W_A \wedge W_C) \rightarrow \neg\big((W_B \wedge W_C) \rightarrow W_A\big)$
- $\varphi_C := \neg W_B \leftrightarrow (W_A \vee W_B)$

Wir suchen nach einer Interpretation, die die Formel

$$\psi := (W_A \leftrightarrow \varphi_A) \wedge (W_B \leftrightarrow \varphi_B) \wedge (W_C \leftrightarrow \varphi_C)$$

erfüllt.

Lösung mittels Wahrheitstafel

W_A	W_B	W_C	φ_A	φ_B	φ_C	$W_A \leftrightarrow \varphi_A$	$W_B \leftrightarrow \varphi_B$	$W_C \leftrightarrow \varphi_C$	ψ
0	0	0	1	1	0	0	0	1	0
0	0	1	0	1	0	1	0	0	0
0	1	0	1	1	0	0	1	1	0
0	1	1	1	1	0	0	1	0	0
1	0	0	1	1	1	1	0	0	0
1	0	1	0	0	1	0	1	1	0
1	1	0	1	1	0	1	1	1	1
1	1	1	1	0	0	1	0	0	0

Die Interpretation $\mathcal{I}$ mit $\mathcal{I}(W_A) = 1, \mathcal{I}(W_B) = 1, \mathcal{I}(W_C) = 0$ in Zeile 7 ist die einzige, die die Formel ψ erfüllt.

Gemäß dieser Interpretation sind die Aussagen, die durch die Symbole W_A und W_B repräsentiert werden, wahr, während die Aussage, die durch W_C repräsentiert wird, falsch ist.

Das heißt, die Personen A und B sagen die Wahrheit und sind somit Was, und Person C lügt und ist daher ein Fa.

Computerlesbare Darstellung von Formeln

Wir betrachten das Alphabet `ASCII` aller ASCII-Symbole.

Die Menge $\mathsf{AS}_{\texttt{ASCII}}$ aller ASCII-Repräsentationen von Aussagensymbolen besteht aus allen nicht-leeren Worten über dem Alphabet `ASCII`, deren erstes Symbol ein Buchstabe ist, und bei dem alle weiteren Symbole Buchstaben oder Ziffern sind.

Die Menge $\mathsf{AL}_{\texttt{ASCII}}$ aller ASCII-Repräsentationen von aussagenlogischen Formeln ist die rekursiv wie folgt definierte Teilmenge von $\texttt{ASCII}^*$:

Basisregeln:

- $0 \in \mathsf{AL}_{\texttt{ASCII}}$, $1 \in \mathsf{AL}_{\texttt{ASCII}}$ und $w \in \mathsf{AL}_{\texttt{ASCII}}$ für alle $w \in \mathsf{AS}_{\texttt{ASCII}}$.

Rekursive Regeln:

- Ist $\varphi \in \mathsf{AL}_{\texttt{ASCII}}$, so ist auch $\texttt{~}\varphi \in \mathsf{AL}_{\texttt{ASCII}}$. (Negation)
- Ist $\varphi \in \mathsf{AL}_{\texttt{ASCII}}$ und $\psi \in \mathsf{AL}_{\texttt{ASCII}}$, so ist auch
  - $(\varphi \texttt{ /\textbackslash } \psi) \in \mathsf{AL}_{\texttt{ASCII}}$ (Konjunktion)
  - $(\varphi \texttt{ \textbackslash / } \psi) \in \mathsf{AL}_{\texttt{ASCII}}$ (Disjunktion)
  - $(\varphi \texttt{ -> } \psi) \in \mathsf{AL}_{\texttt{ASCII}}$ (Implikation)
  - $(\varphi \texttt{ <-> } \psi) \in \mathsf{AL}_{\texttt{ASCII}}$ (Biimplikation).

Abstrakte vs. computerlesbare Syntax

Es ist offensichtlich, wie man Formeln aus AL in ihre entsprechende ASCII-Repräsentation übersetzt und umgekehrt. Zum Beispiel ist

$$((A_0 \wedge \mathbf{0}) \rightarrow \neg A_{13})$$

eine Formel in AL, deren ASCII-Repräsentation die folgende Zeichenkette aus $\mathrm{AL}_{\mathtt{ASCII}}$ ist:

```
( (A0 /\ 0) -> ~A13 ).
```

Wir werden meistens mit der „abstrakten Syntax“, d.h. mit der in Definition 2.4 festgelegten Menge AL, arbeiten. Um aber Formeln in Computer-Programme einzugeben, können wir die ASCII-Repräsentation verwenden.

Demo: snippets of logic

- ein Formelchecker für die Aussagenlogik
- entwickelt von André Frochaux
- Funktionalitäten u.a.:
  - Syntaxcheck für eingegebene Formeln
  - Ausgabe eines Syntaxbaums
  - Ausgabe einer Wahrheitstafel
- Zugänglich via

  `http://www.snippets-of-logic.net/index_AL.php?lang=de`

Zurück zu Beispiel 2.1 („Geburtstagsfeier“)

Das in Beispiel 2.1 aufgelistete Wissen kann durch folgende aussagenlogische Formel repräsentiert werden:

$$\varphi \;:=\; \big((B \wedge A) \to \neg E\big) \;\wedge\; \big((B \wedge E) \to \neg D\big) \;\wedge\; \big(E \to (C \wedge D)\big) \;\wedge\; (C \to A) \;\wedge\; \big(A \to (B \vee C)\big)$$

Die Frage

> *„Wie viele (und welche) Freunde werden im besten Fall zur Party kommen?“*

kann nun durch Lösen der folgenden Aufgabe beantwortet werden:

Finde eine Interpretation $\mathcal{I}$ für φ, so dass gilt:

- $\mathcal{I} \models \varphi$ (d.h., $\mathcal{I}$ ist ein Modell von φ) und
- $|\{X \in \{A, B, C, D, E\} \;:\; \mathcal{I}(X) = 1\}|$ ist so groß wie möglich.

Diese Frage können wir lösen, indem wir

(1) die Wahrheitstafel für φ ermitteln,

(2) alle Zeilen raussuchen, in denen in der mit „φ“ beschrifteten Spalte der Wert 1 steht (das liefert uns genau die Modelle von φ) und

(3) aus diesen Zeilen all jene raussuchen, bei denen in den mit A, B, C, D, E beschrifteten Spalten möglichst viele Einsen stehen. Jede dieser Zeilen repräsentiert dann eine größtmögliche Konstellation von gleichzeitigen Partybesuchern.

Prinzipiell führt diese Vorgehensweise zum Ziel.

Leider ist das Verfahren aber recht aufwändig, da die Wahrheitstafel, die man dabei aufstellen muss, sehr groß wird: Sie hat $2^5 = 32$ Zeilen.

	A	B	C	D	E	$E \to (C \wedge D)$	$C \to A$	$(B \wedge E) \to \neg D$	$A \to (B \vee C)$	$(B \wedge A) \to \neg E$	φ
	0	0	0	0	0	1	1	1	1	1	1
	0	0	0	0	1	0	1	1	1	1	0
	0	0	0	1	0	1	1	1	1	1	1
	0	0	0	1	1	0	1	1	1	1	0
	0	0	1	0	0	1	0	1	1	1	0
	0	0	1	0	1	0	0	1	1	1	0
	0	0	1	1	0	1	0	1	1	1	0
	0	0	1	1	1	1	0	1	1	1	0
	0	1	0	0	0	1	1	1	1	1	1
	0	1	0	0	1	0	1	1	1	1	0
	0	1	0	1	0	1	1	1	1	1	1
	0	1	0	1	1	0	1	0	1	1	0
	0	1	1	0	0	1	0	1	1	1	0
	0	1	1	0	1	0	0	1	1	1	0
	0	1	1	1	0	1	0	1	1	1	0
	0	1	1	1	1	1	0	0	1	1	0
	1	0	0	0	0	1	1	1	0	1	0
	1	0	0	0	1	0	1	1	0	1	0
	1	0	0	1	0	1	1	1	0	1	0
	1	0	0	1	1	0	1	1	0	1	0
	1	0	1	0	0	1	1	1	1	1	1
	1	0	1	0	1	0	1	1	1	1	0
	1	0	1	1	0	1	1	1	1	1	1
★	1	0	1	1	1	1	1	1	1	1	1
	1	1	0	0	0	1	1	1	1	1	1
	1	1	0	0	1	0	1	1	1	0	0
	1	1	0	1	0	1	1	1	1	1	1
	1	1	0	1	1	0	1	0	1	0	0
	1	1	1	0	0	1	1	1	1	1	1
	1	1	1	0	1	0	1	1	1	0	0
★	1	1	1	1	0	1	1	1	1	1	1
	1	1	1	1	1	1	1	0	1	0	0

Modelle für φ werden hier durch grau unterlegte Zeilen repräsentiert.

In der Wahrheitstafel sieht man:

- Es gibt kein Modell für φ, bei dem in den mit A bis E beschrifteten Spalten insgesamt 5 Einsen stehen.
- Es gibt genau zwei Modelle für φ, bei denen in den mit A bis E beschrifteten Spalten insgesamt 4 Einsen stehen, nämlich die beiden Interpretationen $\mathcal{I}_1$ und $\mathcal{I}_2$ mit

$$\mathcal{I}_1(A) = \mathcal{I}_1(C) = \mathcal{I}_1(D) = \mathcal{I}_1(E) = 1 \text{ und } \mathcal{I}_1(B) = 0$$

und

$$\mathcal{I}_2(A) = \mathcal{I}_2(B) = \mathcal{I}_2(C) = \mathcal{I}_2(D) = 1 \text{ und } \mathcal{I}_2(E) = 0.$$

Die Antwort auf die Frage *„Wie viele (und welche) Freunde werden bestenfalls zur Party kommen?“* lautet also:

Bestenfalls werden 4 der 5 Freunde kommen, und dafür gibt es zwei Möglichkeiten, nämlich

(1) dass alle außer Bernd kommen, und

(2) dass alle außer Eva kommen.

Erfüllbarkeit, Allgemeingültigkeit und die Folgerungsbeziehung

Erfüllbarkeit

Definition 2.10

Eine Formel $\varphi \in \mathrm{AL}$ heißt erfüllbar, wenn es eine Interpretation gibt, die φ erfüllt.

Eine Formelmenge Φ heißt erfüllbar, wenn es eine Interpretation $\mathcal{I}$ gibt, die Φ erfüllt (d.h. es gilt $\mathcal{I} \models \varphi$ für jedes $\varphi \in \Phi$).

Eine Formel oder Formelmenge, die nicht erfüllbar ist, nennen wir unerfüllbar.

Beobachtung 2.11

(a) *Eine aussagenlogische Formel ist genau dann erfüllbar, wenn in der letzten Spalte ihrer Wahrheitstafel mindestens eine* 1 *steht.*

(b) *Eine endliche Formelmenge* $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ *ist genau dann erfüllbar, wenn die Formel* $\bigwedge_{i=1}^{n} \varphi_i$ *erfüllbar ist.*

Beispiele:

- Die Formel X ist erfüllbar.
- Die Formel $(X \wedge \neg X)$ ist unerfüllbar.

Allgemeingültigkeit

Definition 2.12

Eine Formel $\varphi \in \text{AL}$ ist allgemeingültig, wenn *jede* Interpretation $\mathcal{I}$ die Formel φ erfüllt.

Bemerkung

Allgemeingültige Formeln nennt man auch Tautologien.
Man schreibt auch $\models \varphi$ um auszudrücken, dass φ allgemeingültig ist.

Beobachtung 2.13

Eine aussagenlogische Formel ist genau dann allgemeingültig, wenn in der letzten Spalte ihrer Wahrheitstafel nur 1*en stehen.*

Beispiel: Die Formel $(X \vee \neg X)$ ist allgemeingültig.

Beispiel 2.14

Die Formel $(X \vee Y) \wedge (\neg X \vee Y)$ ist

- erfüllbar, da z.B. die Interpretation $\mathcal{I}$ mit $\mathcal{I}(X) = 0$ und $\mathcal{I}(Y) = 1$ die Formel erfüllt.
- nicht allgemeingültig, da z.B. die Interpretation $\mathcal{I}'$ mit $\mathcal{I}'(X) = 0$ und $\mathcal{I}'(Y) = 0$ die Formel nicht erfüllt.

Die Folgerungsbeziehung

Definition 2.15

Eine Formel $\psi \in \mathrm{AL}$ folgt aus einer Formelmenge $\Phi \subseteq \mathrm{AL}$ (wir schreiben: $\Phi \models \psi$), wenn für jede Interpretation $\mathcal{I}$ gilt: Wenn $\mathcal{I}$ die Formelmenge Φ erfüllt, dann erfüllt $\mathcal{I}$ auch die Formel ψ.

Notation

Für zwei Formeln $\varphi, \psi \in \mathrm{AL}$ schreiben wir kurz $\varphi \models \psi$ an Stelle von $\{\varphi\} \models \psi$ und sagen, dass die Formel ψ aus der Formel φ folgt.

Beispiel 2.16

Sei $\varphi := ((X \vee Y) \wedge (\neg X \vee Y))$ und $\psi := (Y \vee (\neg X \wedge \neg Y))$.
Dann gilt $\varphi \models \psi$, aber es gilt *nicht* $\psi \models \varphi$ (kurz: $\psi \not\models \varphi$), denn:

X	Y	$(X \vee Y)$	$(\neg X \vee Y)$	φ	ψ
0	0	0	1	0	1
0	1	1	1	1	1
1	0	1	0	0	0
1	1	1	1	1	1

In jeder Zeile der Wahrheitstafel, in der in der mit „φ“ beschrifteten Spalte eine 1 steht, steht auch in der mit „ψ“ beschrifteten Spalte eine 1. Somit gilt $\varphi \models \psi$.

Andererseits steht in Zeile 1 in der mit „ψ“ beschrifteten Spalte eine 1 und in der mit „φ“ beschrifteten Spalte eine 0. Für die entsprechende Interpretation $\mathcal{I}$ (mit $\mathcal{I}(X) = 0$ und $\mathcal{I}(Y) = 0$) gilt also $[\![\psi]\!]^{\mathcal{I}} = 1$ und $[\![\varphi]\!]^{\mathcal{I}} = 0$. Daher gilt $\psi \not\models \varphi$.

Beispiel 2.17

Für alle Formeln $\varphi, \psi \in \text{AL}$ gilt:

$$\{\varphi,\ \varphi \to \psi\} \models \psi.$$

Dies folgt unmittelbar aus der Definition der Semantik:
Sei $\mathcal{I}$ eine Interpretation mit $\mathcal{I} \models \{\varphi, \varphi \to \psi\}$. Dann gilt:

(1) $[\![\varphi]\!]^{\mathcal{I}} = 1$ und

(2) $[\![\varphi \to \psi]\!]^{\mathcal{I}} = 1$, d.h. es gilt $[\![\varphi]\!]^{\mathcal{I}} = 0$ oder $[\![\psi]\!]^{\mathcal{I}} = 1$.

Da $[\![\varphi]\!]^{\mathcal{I}} = 1$ gemäß (1) gilt, folgt gemäß (2), dass $[\![\psi]\!]^{\mathcal{I}} = 1$.

Bemerkung

Man kann die Folgerungsbeziehung $\{\varphi, \varphi \to \psi\} \models \psi$ als eine formale Schlussregel auffassen (ähnlich den Syllogismen in Kapitel 1):
Wenn φ und $\varphi \to \psi$ gelten, so muss auch ψ gelten.
Diese Regel, die unter dem Namen Modus Ponens bekannt ist, ist von grundlegender Bedeutung in der Logik.

Pingo-Übung

http://pingo.upb.de/160267

Zusammenhänge

Lemma 2.18 (Allgemeingültigkeit, Unerfüllbarkeit und Folgerung)

Für jede Formel $\varphi \in \mathsf{AL}$ *gilt:*

(a) φ *ist allgemeingültig* $\iff$ $\neg\varphi$ *ist unerfüllbar* $\iff$ $\mathbf{1} \models \varphi$.

(b) φ *ist unerfüllbar* $\iff$ $\varphi \models \mathbf{0}$.

Lemma 2.19 (Erfüllbarkeit und die Folgerungsbeziehung)

Für alle Formelmengen $\Phi \subseteq \mathsf{AL}$ *und für alle Formeln* $\psi \in \mathsf{AL}$ *gilt:*

$$\Phi \models \psi \quad \Longleftrightarrow \quad \Phi \cup \{\neg\psi\} \text{ ist unerfüllbar.}$$

Lemma 2.20 (Allgemeingültigkeit und die Folgerungsbeziehung)

(a) *Für jede Formel* $\varphi \in \mathsf{AL}$ *gilt:*

$$\varphi \text{ ist allgemeingültig} \iff \varphi \text{ folgt aus der leeren Menge,}$$

kurz:

$$\models \varphi \iff \emptyset \models \varphi.$$

(b) *Für jede Formel* $\psi \in \mathsf{AL}$ *und jede endliche Formelmenge* $\Phi = \{\varphi_1, \ldots, \varphi_n\} \subseteq \mathsf{AL}$ *gilt:*

$$\Phi \models \psi \iff (\varphi_1 \wedge \cdots \wedge \varphi_n) \to \psi \text{ ist allgemeingültig.}$$

Bemerkung 2.21

Aus den beiden vorigen Lemmas erhält man leicht, dass für alle Formeln $\varphi, \psi \in \text{AL}$ gilt:

$$\varphi \models \psi \iff (\varphi \to \psi) \text{ ist allgemeingültig} \iff (\varphi \wedge \neg\psi) \text{ ist unerfüllbar.}$$

Beweis.

Es sei $\Phi := \{\varphi\}$. Gemäß Lemma 2.20 gilt:

$$\Phi \models \psi \iff (\varphi \to \psi) \text{ ist allgemeingültig.}$$

Somit gilt: $\varphi \models \psi \iff (\varphi \to \psi)$ ist allgemeingültig.

Außerdem gilt gemäß Lemma 2.19:

$$\Phi \models \psi \iff \Phi \cup \{\neg\psi\} \text{ ist unerfüllbar.}$$

Somit gilt: $\varphi \models \psi \iff (\varphi \wedge \neg\psi)$ ist unerfüllbar. □

Abschnitt 2.2:
Aussagenlogische Modellierung

Beispiel 1: Sudoku

Sudoku

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

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

Aussagenlogisches Modell

Koordinaten der Felder:

Feld (i, j) ist das Feld in Zeile i und Spalte j.

Aussagensymbole:

Aussagensymbol $P_{i,j,k}$, für $i, j, k \in [9]$, steht für die Aussage

„Das Feld mit den Koordinaten (i, j) enthält die Zahl k."

Interpretationen beschreiben also Beschriftungen des 9×9-Gitters.

Ziel:

Für jede Anfangsbeschriftung A eine Formelmenge Φ_A, so dass für alle Interpretationen $\mathcal{I}$ gilt:

$$\mathcal{I} \models \Phi_A \quad \Longleftrightarrow \quad \mathcal{I} \text{ beschreibt eine korrekte Lösung.}$$

Wir beschreiben zunächst eine Formelmenge $\Phi = \{\varphi_1, \ldots, \varphi_5\}$, die die Grundregeln des Spiels beschreibt.

Beschriftungen:

"Auf jedem Feld steht mindestens eine Zahl":

$$\varphi_1 \;:=\; \bigwedge_{i,j=1}^{9} \bigvee_{k=1}^{9} P_{i,j,k}.$$

"Auf jedem Feld steht höchstens eine Zahl":

$$\varphi_2 \;:=\; \bigwedge_{i,j=1}^{9} \bigwedge_{\substack{k,\ell=1 \\ k\neq\ell}}^{9} \neg(P_{i,j,k} \wedge P_{i,j,\ell}).$$

Zeilen:

„Jede Zahl kommt in jeder Zeile vor“:

$$\varphi_3 := \bigwedge_{i=1}^{9} \bigwedge_{k=1}^{9} \bigvee_{j=1}^{9} P_{i,j,k}.$$

Spalten:

„Jede Zahl kommt in jeder Spalte vor“:

$$\varphi_4 := \bigwedge_{j=1}^{9} \bigwedge_{k=1}^{9} \bigvee_{i=1}^{9} P_{i,j.k}.$$

Blöcke:

„Jede Zahl kommt in jedem Block vor“:

$$\varphi_5 := \bigwedge_{i,j=0}^{2} \bigwedge_{k=1}^{9} \bigvee_{i',j'=1}^{3} P_{3i+i',3j+j',k}.$$

Anfangsbeschriftung:

Sei A die Anfangsbeschriftung. Wir setzen

$$\Phi_A \;:=\; \Phi \cup \{\, P_{i,j,k} \;:\; A \text{ beschriftet Feld } (i,j) \text{ mit der Zahl } k \,\}.$$

Automatische Lösung von Sudokus:

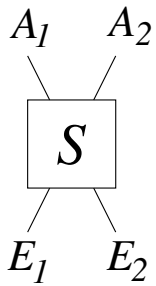
Um ein Sudoku mit Anfangsbeschriftung A zu lösen, können wir nun einfach die Formel $\psi_A := \bigwedge_{\varphi \in \Phi_A} \varphi$ bilden und die Wahrheitstafel zu dieser Formel aufstellen. Falls die Wahrheitstafel zeigt, dass ψ_A kein Modell besitzt, so ist das Sudoku nicht lösbar. Andernfalls können wir ein beliebiges Modell $\mathcal{I}$ von ψ_A hernehmen und daran die Lösung des Sudokus ablesen: Für jedes Feld (i,j) gibt es gemäß unserer Konstruktion der Formel ψ_A genau eine Zahl $k \in [9]$, so dass $\mathcal{I}(P_{i,j,k}) = 1$ ist. Diese Zahl k können wir in Feld (i,j) eintragen und erhalten damit eine Lösung des Sudokus.

Beispiel 2: Automatische Hardwareverifikation

Digitale Schaltkreise

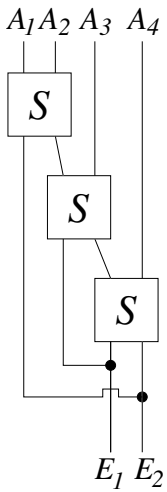
- Digitale Signale werden beschrieben durch 0 („aus“) und 1 („ein“).
- Schaltelemente berechnen ein oder mehrere Ausgangssignale aus einem oder mehreren Eingangssignalen. Das Ein-/Ausgabeverhalten eines Schaltelements lässt sich durch Wahrheitstafeln beschreiben.

  Beispiel:



E_1	E_2	A_1	A_2
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

- Schaltkreise sind Kombinationen solcher Schaltelemente. Beispiel:



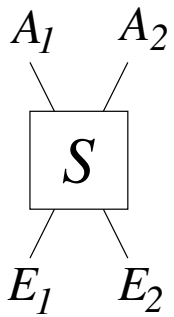
E_1	E_2	A_1	A_2	A_3	A_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	1	0	1	0
1	1	0	1	0	1

Formalisierung in der Aussagenlogik

Schaltelement:

- Für jeden Ein- und Ausgang ein Aussagensymbol.
- Für jeden Ausgang eine Formel, die den Wert der Ausgangs in Abhängigkeit von den Eingängen beschreibt.

Beispiel:



E_1	E_2	A_1	A_2
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Aussagensymbole:

$P_1, P_2, \quad Q_1, Q_2$

Formeln:

$Q_1 \leftrightarrow \neg(P_1 \wedge P_2)$

$Q_2 \leftrightarrow (P_1 \wedge P_2)$

Schaltkreis:

- Für jeden Ein- und Ausgang ein Aussagensymbol,
  sowie für jedes Schaltelement ein Sortiment von Aussagensymbolen.
- Formeln für die Schaltelemente und Formeln für die „Verdrahtung“.

Verifikation

Ziel:

Nachweis, dass der Schaltkreis eine gewisse *Korrektheitsbedingung* erfüllt.

Methode:

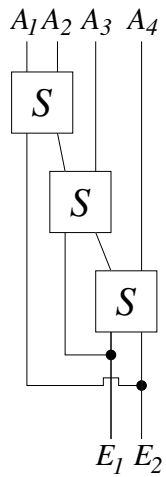
1. Beschreibe den Schaltkreis durch eine Menge Φ von Formeln.
2. Formuliere die Korrektheitsbedingung als Formel ψ.
3. Weise nach, dass ψ aus Φ folgt
   (bzw., dass $\Phi \cup \{\neg\psi\}$ unerfüllbar ist).

Bemerkung

Bei Bedarf kann die Korrektheitsbedingung insbesondere so gewählt werden, dass sie das gewünschte Ein-/Ausgabeverhalten des Schaltkreises vollständig spezifiziert.

Beispiele für Korrektheitsbedingungen

Schaltkreis:



Einige Korrektheitsbedingungen:

- Bei jeder Eingabe ist mindestens eine Ausgabe 1:

$$Q_1 \vee Q_2 \vee Q_3 \vee Q_4.$$

- Bei keiner Eingabe sind mehr als zwei Ausgaben 1:

$$\neg \bigvee_{1 \leqslant i < j < k \leqslant 4} (Q_i \wedge Q_j \wedge Q_k)$$

Vollständige Spezifikation des Ein-/Ausgabeverhaltens:

$$\begin{array}{lrcl}
 & (\ \neg P_1 \wedge \neg P_2 & \rightarrow & Q_1 \wedge \neg Q_2 \wedge \neg Q_3 \wedge \neg Q_4\ ) \\
\wedge & (\ \neg P_1 \wedge P_2 & \rightarrow & \neg Q_1 \wedge Q_2 \wedge \neg Q_3 \wedge \neg Q_4\ ) \\
\wedge & (\ P_1 \wedge \neg P_2 & \rightarrow & Q_1 \wedge \neg Q_2 \wedge Q_3 \wedge \neg Q_4\ ) \\
\wedge & (\ P_1 \wedge P_2 & \rightarrow & \neg Q_1 \wedge Q_2 \wedge \neg Q_3 \wedge Q_4\ )
\end{array}$$

Pingo-Übung

Welche Aussage gilt für $\varphi = A \wedge B \to C$ und $\psi = (A \to C) \wedge (B \to C)$?

(1) $\varphi \models \psi$ und $\psi \models \varphi$
(2) $\varphi \models \psi$ und $\psi \not\models \varphi$
(3) $\varphi \not\models \psi$ und $\psi \models \varphi$
(4) $\varphi \not\models \psi$ und $\psi \not\models \varphi$

http://pingo.upb.de/160267

Abschnitt 2.3:

Äquivalenz und Adäquatheit

Äquivalenz

Definition 2.22

Zwei Formeln $\varphi, \psi \in \text{AL}$ sind äquivalent (wir schreiben $\varphi \equiv \psi$), wenn sie von den selben Interpretationen erfüllt werden, d.h., wenn für alle Interpretationen $\mathcal{I}$ gilt:
$\mathcal{I} \models \varphi \iff \mathcal{I} \models \psi$.

Zwei Formelmengen $\Phi, \Psi \subseteq \text{AL}$ sind äquivalent (wir schreiben $\Phi \equiv \Psi$), wenn sie von den selben Interpretationen erfüllt werden, d.h., wenn für alle Interpretationen $\mathcal{I}$ gilt:
$\mathcal{I} \models \Phi \iff \mathcal{I} \models \Psi$.

Beobachtung 2.23

(a) *Zwei Formeln $\varphi, \psi \in \text{AL}$ sind genau dann äquivalent, wenn in den letzten Spalten ihrer Wahrheitstafeln jeweils die gleichen Einträge stehen.*

(b) *Für endliche Formelmengen $\Phi = \{\varphi_1, \ldots, \varphi_m\}$, $\Psi = \{\psi_1, \ldots, \psi_n\} \subseteq \text{AL}$ gilt*

$$\Phi \equiv \Psi \quad \iff \quad \bigwedge_{i=1}^{m} \varphi_i \equiv \bigwedge_{j=1}^{n} \psi_j.$$

Beispiel: Für alle $X, Y \in \text{AS}$ gilt: $\neg(X \vee Y) \equiv (\neg X \wedge \neg Y)$ und $X \equiv \neg\neg X$.

Äquivalenz und Allgemeingültigkeit

Lemma 2.24

(a) *Für alle Formeln* $\varphi, \psi \in \mathsf{AL}$ *gilt:*

$$\varphi \equiv \psi \qquad \Longleftrightarrow \qquad (\varphi \leftrightarrow \psi) \text{ ist allgemeingültig.}$$

(b) *Für alle* $\varphi \in \mathsf{AL}$ *gilt:*

$$\varphi \text{ ist allgemeingültig} \qquad \Longleftrightarrow \qquad \varphi \equiv \mathbf{1}.$$

Fundamentale Äquivalenzen

Satz 2.25

Für alle Formeln $\varphi, \psi, \chi \in \mathsf{AL}$ gelten die folgenden Äquivalenzen:

(a) *Idempotenz:*

$$\varphi \wedge \varphi \;\equiv\; \varphi\,, \qquad \varphi \vee \varphi \;\equiv\; \varphi.$$

(b) *Kommutativität:*

$$\varphi \wedge \psi \;\equiv\; \psi \wedge \varphi\,, \qquad \varphi \vee \psi \;\equiv\; \psi \vee \varphi.$$

(c) *Assoziativität:*

$$(\varphi \wedge \psi) \wedge \chi \;\equiv\; \varphi \wedge (\psi \wedge \chi)\,, \qquad (\varphi \vee \psi) \vee \chi \;\equiv\; \varphi \vee (\psi \vee \chi).$$

(d) *Absorption:*

$$\varphi \wedge (\varphi \vee \psi) \;\equiv\; \varphi\,, \qquad \varphi \vee (\varphi \wedge \psi) \;\equiv\; \varphi.$$

(Fortsetzung: nächste Folie)

(e) *Distributivität:*

$$\varphi \wedge (\psi \vee \chi) \;\equiv\; (\varphi \wedge \psi) \vee (\varphi \wedge \chi)\,, \qquad \varphi \vee (\psi \wedge \chi) \;\equiv\; (\varphi \vee \psi) \wedge (\varphi \vee \chi).$$

(f) *Doppelte Negation:*

$$\neg\neg\varphi \;\equiv\; \varphi.$$

(g) *De Morgansche Regeln:*

$$\neg(\varphi \wedge \psi) \;\equiv\; \neg\varphi \vee \neg\psi\,, \qquad \neg(\varphi \vee \psi) \;\equiv\; \neg\varphi \wedge \neg\psi.$$

(h) *Tertium Non Datur:*

$$\varphi \wedge \neg\varphi \;\equiv\; \mathbf{0}\,, \qquad \varphi \vee \neg\varphi \;\equiv\; \mathbf{1}.$$

(Fortsetzung: nächste Folie)

(i)

$$\varphi \wedge \mathbf{1} \equiv \varphi\,, \qquad \varphi \vee \mathbf{0} \equiv \varphi\,,$$
$$\varphi \wedge \mathbf{0} \equiv \mathbf{0}\,, \qquad \varphi \vee \mathbf{1} \equiv \mathbf{1}.$$

(j)

$$\mathbf{1} \equiv \neg\mathbf{0}\,, \qquad \mathbf{0} \equiv \neg\mathbf{1}.$$

(k) *Elimination der Implikation:*

$$\varphi \to \psi \equiv \neg\varphi \vee \psi.$$

Beweis.

Alle hier genannten Äquivalenzen können leicht mit Hilfe der Wahrheitstafelmethode überprüft werden.

Zum Beispiel die erste de Morgansche Regel:

$$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi.$$

Wir berechnen dazu folgende Wahrheitstafeln:

φ	ψ	$\varphi \wedge \psi$	$\neg(\varphi \wedge \psi)$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

φ	ψ	$\neg\varphi$	$\neg\psi$	$\neg\varphi \vee \neg\psi$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

Die letzten Spalten der beiden Wahrheitstafeln sind gleich, also sind die Formeln äquivalent.

Rest: Übung. □

Bemerkung

- Sei
  - $\varphi \in \mathsf{AL}$,
  - $\psi \in \mathsf{AL}$ eine Teilformel von φ,
  - $\psi' \in \mathsf{AL}$ mit $\psi \equiv \psi'$.

  Wenn φ' eine Formel ist, die aus φ entsteht,
  indem ein oder mehrere Vorkommen von ψ in φ durch ψ' ersetzt werden,
  dann gilt $\varphi' \equiv \varphi$.
- Durch schrittweises Anwenden der in Satz 2.25 aufgelisteten Äquivalenzen auf Teilformeln kann man somit eine gegebene Formel in eine zu ihr äquivalente Formel umformen.

Das Dualitätsprinzip

Definition 2.26

Sei $\varphi \in \mathrm{AL}$ eine Formel, in der keine Implikationen vorkommt.
Die zu φ duale Formel ist die Formel $\widetilde{\varphi} \in \mathrm{AL}$, die aus φ entsteht, indem man überall $\mathbf{0}$ durch $\mathbf{1}$, $\mathbf{1}$ durch $\mathbf{0}$, $\wedge$ durch $\vee$ und $\vee$ durch $\wedge$ ersetzt.

Beobachtung 2.27

In Satz 2.25(a)–(e) *und* (g)–(j) *stehen auf der linken Seite jeweils die dualen Formeln der Formeln auf der rechten Seite.*

Satz 2.28 (Dualitätssatz der Aussagenlogik)

Für alle Formeln $\varphi, \psi \in \mathrm{AL}$, in denen keine Implikation vorkommt, gilt:

$$\varphi \equiv \psi \qquad \Longleftrightarrow \qquad \widetilde{\varphi} \equiv \widetilde{\psi}.$$

Um den Dualitätssatz zu beweisen benötigen wir zunächst noch eine Definition. Der Kern des Beweises steckt im darauf folgenden Lemma.

Definition 2.29

Sei $\mathcal{I}$ eine Interpretation. Die zu $\mathcal{I}$ duale Interpretation $\widetilde{\mathcal{I}}$ ist definiert durch $\widetilde{\mathcal{I}}(X) := 1 - \mathcal{I}(X)$ für alle $X \in \mathsf{AS}$.

D.h. für alle Aussagensymbole X gilt:

$$\widetilde{\mathcal{I}}(X) \;=\; \begin{cases} 0\,, & \text{falls } \mathcal{I}(X) = 1 \\ 1\,, & \text{falls } \mathcal{I}(X) = 0 \end{cases}$$

Lemma 2.30

Für alle Formeln $\varphi \in \mathsf{AL}$, in denen keine Implikation vorkommt, und alle Interpretationen $\mathcal{I}$ gilt:

$$\mathcal{I} \models \widetilde{\varphi} \quad \Longleftrightarrow \quad \widetilde{\mathcal{I}} \not\models \varphi.$$

Beweis von Satz 2.28 unter Verwendung von Lemma 2.30.

Seien $\varphi, \psi \in$ AL Formeln, in denen keine Implikation vorkommt.
Wir wollen zeigen, dass gilt: $\varphi \equiv \psi \iff \widetilde{\varphi} \equiv \widetilde{\psi}$.

„$\Longrightarrow$“: Es gilt:[1]

$$\varphi \equiv \psi$$

$\Longrightarrow$ F.a. Interpretationen $\mathcal{I}$ gilt: $\big(\widetilde{\mathcal{I}} \models \varphi \iff \widetilde{\mathcal{I}} \models \psi\big)$

$\overset{\text{Lemma 2.30}}{\Longrightarrow}$ F.a. Interpretationen $\mathcal{I}$ gilt: $\big(\mathcal{I} \not\models \widetilde{\varphi} \iff \mathcal{I} \not\models \widetilde{\psi}\big)$

$\Longrightarrow$ F.a. Interpretationen $\mathcal{I}$ gilt: $\big(\mathcal{I} \models \widetilde{\varphi} \iff \mathcal{I} \models \widetilde{\psi}\big)$

$\Longrightarrow$ $\widetilde{\varphi} \equiv \widetilde{\psi}$.

„$\Longleftarrow$“: Es gilt:

$\widetilde{\varphi} \equiv \widetilde{\psi} \Longrightarrow \widetilde{\widetilde{\varphi}} \equiv \widetilde{\widetilde{\psi}}$ (andere Beweisrichtung)

$\Longrightarrow \varphi \equiv \psi$ (weil $\widetilde{\widetilde{\varphi}} = \varphi$ und $\widetilde{\widetilde{\psi}} = \psi$).

□

[1] Wir schreiben kurz „f.a.“ als Abkürzung für die Worte „für alle“

Beweise per Induktion über den Aufbau von Formeln

- Ähnlich wie wir Aussagen über die natürlichen Zahlen durch vollständige Induktion beweisen können, können wir Aussagen über Formeln per Induktion über den Aufbau der Formeln beweisen.

- Im Induktionsanfang beweisen wir die Aussagen für die atomaren Formeln, und im Induktionschritt schließen wir von den Bestandteilen einer Formel auf die Formel selbst.

- Dieses Vorgehen ist z.B. dadurch gerechtfertigt, dass es sich auch als vollständige Induktion über die Höhe des Syntaxbaumes auffassen lässt.

Schematisch sieht der Beweis einer Aussage $\mathbb{A}(\varphi)$ für alle Formeln $\varphi \in \mathsf{AL}$ wie folgt aus:

Induktionsanfang:

- Beweise $\mathbb{A}(\mathbf{0})$ und $\mathbb{A}(\mathbf{1})$.
- Beweise $\mathbb{A}(X)$ für alle $X \in \mathsf{AS}$.

Induktionsschritt:

- Beweise $\mathbb{A}(\neg\varphi)$ unter der Annahme, dass $\mathbb{A}(\varphi)$ gilt.
- Beweise $\mathbb{A}(\varphi \wedge \psi)$ unter der Annahme, dass $\mathbb{A}(\varphi)$ und $\mathbb{A}(\psi)$ gelten.
- Beweise $\mathbb{A}(\varphi \vee \psi)$ unter der Annahme, dass $\mathbb{A}(\varphi)$ und $\mathbb{A}(\psi)$ gelten.
- Beweise $\mathbb{A}(\varphi \to \psi)$ unter der Annahme, dass $\mathbb{A}(\varphi)$ und $\mathbb{A}(\psi)$ gelten.

Funktionale Vollständigkeit der Aussagenlogik

Im Folgenden bezeichnen wir als Wahrheitstafel eine Tabelle mit $n+1$ Spalten und 2^n Zeilen, die für jedes Tupel $(b_1, \ldots, b_n) \in \{0,1\}^n$ genau eine Zeile enthält, deren erste n Einträge $b_1, \ldots, b_n$ sind.

Satz 2.31 (Funktionale Vollständigkeit der Aussagenlogik)

Zu jeder Wahrheitstafel gibt es eine Formel $\varphi \in AL$ mit dieser Wahrheitstafel.

Mathematisch präzise lässt sich dieser Satzes wie folgt formulieren:

Für alle $n \in \mathbb{N}$ gibt es zu jeder Funktion $F : \{0,1\}^n \to \{0,1\}$ eine Formel $\varphi(A_1, \ldots, A_n) \in \mathrm{AL}$, so dass für alle $(b_1, \ldots, b_n) \in \{0,1\}^n$ gilt:

$$F(b_1, \ldots, b_n) = 1 \iff \varphi[b_1, \ldots, b_n] = 1.$$

Definition 2.32

Funktionen $F : \{0,1\}^n \to \{0,1\}$ (mit $n \in \mathbb{N}$) nennt man Boolesche Funktionen (der Stelligkeit n).

Bevor wir Satz 2.31 beweisen, betrachten wir zunächst ein Beispiel.

Beispiel 2.33

Betrachte die Wahrheitstafel T:

b_1	b_2	b_3	$F(b_1, b_2, b_3)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Eine Formel $\varphi(A_1, A_2, A_3)$, so dass T die Wahrheitstafel für φ ist, kann man folgendermaßen erzeugen:

- Betrachte alle Zeilen von T, bei denen in der letzten Spalte eine „1“ steht.
- Für jede solche Zeile konstruiere eine Formel, die genau von der zu der Zeile gehörenden Belegung von b_1, b_2, b_3 erfüllt wird.
- Bilde die Disjunktion (d.h. die „Ver*oder*ung“) über all diese Formeln. Dies liefert die gesuchte Formel φ.

In unserer Beispiel-Wahrheitstafel T gibt es genau 3 Zeilen, bei denen in der letzten Spalte eine „1“ steht, nämlich die Zeilen

b_1	b_2	b_3	$F(b_1, b_2, b_3)$	zur jeweiligen Zeile gehörende Formel:
0	0	0	1	$(\neg A_1 \wedge \neg A_2 \wedge \neg A_3)$
0	0	1	1	$(\neg A_1 \wedge \neg A_2 \wedge A_3)$
⋮	⋮	⋮	⋮	
1	0	1	1	$(A_1 \wedge \neg A_2 \wedge A_3)$
⋮	⋮	⋮	⋮	

Insgesamt erhalten wir dadurch die zur Wahrheitstafel T passende Formel

$$\varphi \;=\; (\neg A_1 \wedge \neg A_2 \wedge \neg A_3) \vee (\neg A_1 \wedge \neg A_2 \wedge A_3) \vee (A_1 \wedge \neg A_2 \wedge A_3).$$

Adäquatheit

Satz 2.31 besagt, dass die Aussagenlogik AL die größtmögliche Aussdruckstärke hat. Dafür reichen allerdings schon „kleinere“ Logiken, wie wir im Folgenden sehen werden.

Definition 2.34

Sei $\tau \subseteq \{\mathbf{0}, \mathbf{1}, \neg, \wedge, \vee, \rightarrow\}$.

(a) AL(τ) sei das Fragment der Logik AL, das aus den Formeln besteht, in denen nur Junktoren und Konstanten aus τ vorkommen.

(b) τ heißt adäquat, wenn jede Formel $\varphi \in$ AL äquivalent zu einer Formel in AL(τ) ist.

Beispiele 2.35

(a) $\{\neg, \wedge\}$, $\{\neg, \vee\}$, $\{\mathbf{0}, \rightarrow\}$ sind adäquat.

(b) $\{\wedge, \vee, \rightarrow\}$ ist nicht adäquat.

Andere Junktoren

- Die Auswahl der Junktoren $\neg, \wedge, \vee, \rightarrow$ (und $\leftrightarrow$ als Abkürzung) für „unsere“ aussagenlogische Sprache richtet sich nach dem umgangssprachlichen Gebrauch und den Erfordernissen des formalen Schließens, ist aber in gewisser Weise willkürlich.

- Durch Festlegung ihrer Wahrheitstafeln können wir auch andere Junktoren definieren und erhalten daraus andere aussagenlogische Sprachen.

- Für jede Menge τ von so definierten Junktoren und den boolschen Konstanten (die wir als „nullstellige“ Junktoren auffassen können) sei $\mathrm{AL}(\tau)$ die daraus gebildete aussagenlogische Sprache.

- Satz 2.31 besagt dann, dass jede Formel in $\mathrm{AL}(\tau)$ zu einer Formel in AL äquivalent ist. Gilt die Umkehrung ebenfalls, so bezeichnen wir τ als adäquat.

Beispiele 1: Exklusives Oder

Der 2-stellige Junktor $\oplus$ sei definiert durch

φ	ψ	$\varphi \oplus \psi$
0	0	0
0	1	1
1	0	1
1	1	0

Intuitiv bedeutet $\varphi \oplus \psi$ „entweder φ oder ψ“.
Man nennt $\oplus$ auch exklusives Oder.

Der dreistellige Mehrheitsjunktor

Der 3-stellige Junktor M sei definiert durch

φ	ψ	χ	$M(\varphi, \psi, \chi)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Intuitiv ist $M(\varphi, \psi, \chi)$ also genau dann wahr, wenn mindestens zwei (also die Mehrheit) der Formeln φ, ψ, χ wahr sind.

NAND

Der folgende zweistellige Junktor ist bekannt als NAND-Gatter (not-and) oder Sheffer-Strich:

φ	ψ	$(\varphi \mid \psi)$
0	0	1
0	1	1
1	0	1
1	1	0

Satz 2.36

$\{\mid\}$ *ist adäquat.*

Beweis.

Induktion über den Aufbau von $\mathrm{AL}(\{\neg, \wedge\})$ unter Verwendung der Äquivalenzen

$$\neg\varphi \;\equiv\; (\varphi \mid \varphi) \quad \text{und} \quad (\varphi \wedge \psi) \;\equiv\; \neg(\varphi \mid \psi) \;\equiv\; \big((\varphi \mid \psi) \mid (\varphi \mid \psi)\big)\,.$$

Details: Übung. □

Abschnitt 2.4:

Normalformen

Vereinfachende Annahme

In diesem Abschnitt betrachten wir nur Formeln in AL($\{\neg, \vee, \wedge\}$).

Rechtfertigung

Die Annahme bedeutet keine wesentliche Einschränkung, weil die Menge $\{\neg, \vee, \wedge\}$ adäquat ist.

NNF

Definition 2.37

Eine Formel ist in Negationsnormalform (NNF), wenn sie zu $\mathrm{AL}(\{\neg, \wedge, \vee\})$ gehört und Negationszeichen nur unmittelbar vor Aussagensymbolen auftreten.

Satz 2.38

Jede aussagenlogische Formel $\varphi \in \mathrm{AL}(\{\neg, \wedge, \vee\})$ ist äquivalent zu einer Formel in NNF, deren Länge linear in der Länge von φ ist.

- Da $\{\neg, \vee, \wedge\}$ adäquat ist und jede Formel aus AL in Linearzeit in eine äquivalente Formel aus $\mathrm{AL}(\{\neg, \vee, \wedge\})$ überführt werden kann, gilt der Satz auch für alle $\varphi \in \mathrm{AL}$.

Ein NNF-Algorithmus

Eingabe: Formel $\varphi \in \mathsf{AL}(\{\neg, \wedge, \vee\})$.

Ausgabe: Formel φ' in NNF

Verfahren:

1. Wiederhole folgende Schritte:
2. Wenn φ in NNF ist, dann halte mit Ausgabe φ.
3. Ersetze eine Subformel von φ der Gestalt $\neg(\psi_1 \wedge \psi_2)$ durch $(\neg\psi_1 \vee \neg\psi_2)$
   oder eine Subformel der Gestalt $\neg(\psi_1 \vee \psi_2)$ durch $(\neg\psi_1 \wedge \neg\psi_2)$
   oder eine Subformel der Gestalt $\neg\neg\psi$ durch ψ.
   Sei φ' die resultierende Formel.
4. $\varphi := \varphi'$.

Korrektheit des NNF-Algorithmus

Satz 2.39

Für jede Eingabeformel $\varphi \in \mathsf{AL}(\{\neg, \wedge, \vee\})$ gibt der NNF-Algorithmus nach endlich vielen Schritten eine zu φ äquivalente Formel φ' in NNF aus.

(hier ohne Beweis)

Bemerkung

Unter Verwendung geeigneter Datenstrukturen lässt sich der NNF-Algorithmus mit linearer Laufzeit implementieren, d.h., Laufzeit $O(n)$ bei Eingabe einer Formel der Länge n.

Beispiel 2.40

Das Ziel ist, die Formel $\Big(\big(\neg A_0 \wedge \neg\big((A_0 \vee A_1) \to A_0\big)\big) \to \mathbf{0}\Big)$

in NNF zu bringen, d.h. eine zu ihr äquivalente Formel in NNF zu finden.

Lösung: Wir ersetzen zunächst die Konstanten $\mathbf{0}$ und $\mathbf{1}$ sowie alle Implikationspfeile durch geeignete Formeln aus $\mathrm{AL}(\{\neg, \wedge, \vee\})$ und wenden dann den NNF-Algorithmus an. Der Teil einer Formel, der als Nächstes ersetzt wird, ist im Folgenden jeweils unterstrichen.

$$
\begin{aligned}
& \Big(\big(\neg A_0 \wedge \neg\big((A_0 \vee A_1) \to A_0\big)\big) \to \underline{\mathbf{0}}\Big) \\
\equiv \quad & \Big(\big(\neg A_0 \wedge \neg\big((A_0 \vee A_1) \to A_0\big)\big) \underline{\to} (A_0 \wedge \neg A_0)\Big) \\
\equiv \quad & \Big(\neg\big(\neg A_0 \wedge \neg\big((A_0 \vee A_1) \underline{\to} A_0\big)\big) \vee (A_0 \wedge \neg A_0)\Big) \\
\equiv \quad & \Big(\underline{\neg}\big(\neg A_0 \wedge \neg\big(\neg(A_0 \vee A_1) \vee A_0\big)\big) \vee (A_0 \wedge \neg A_0)\Big) \\
\equiv \quad & \Big(\big(\underline{\neg\neg} A_0 \vee \underline{\neg\neg}\big(\neg(A_0 \vee A_1) \vee A_0\big)\big) \vee (A_0 \wedge \neg A_0)\Big) \\
\equiv \quad & \Big(\big(A_0 \vee \big(\underline{\neg}(A_0 \vee A_1) \vee A_0\big)\big) \vee (A_0 \wedge \neg A_0)\Big) \\
\equiv \quad & \Big(\big(A_0 \vee \big((\neg A_0 \wedge \neg A_1) \vee A_0\big)\big) \vee (A_0 \wedge \neg A_0)\Big).
\end{aligned}
$$

Diese Formel ist offensichtlicherweise in NNF.

Klammern bei Konjunktionen und Disjunktionen

Weil $\wedge$ assoziativ ist, können wir Formeln der Gestalt $\bigwedge_{i=1}^{n} \varphi_i$ etwas großzügiger interpretieren. Von nun an stehe $\bigwedge_{i=1}^{n} \varphi_i$ für $\varphi_1 \wedge \cdots \wedge \varphi_n$ mit *irgendeiner* Klammerung.

Entsprechend verfahren wir mit Disjunktionen.

Beispiel

Die Formel $\bigwedge_{i=1}^{4} \varphi_i$ kann für jede der folgenden Formeln stehen:

$$(((\varphi_1 \wedge \varphi_2) \wedge \varphi_3) \wedge \varphi_4) \ ,$$

$$((\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)) \wedge \varphi_4) \ ,$$

$$((\varphi_1 \wedge \varphi_2) \wedge (\varphi_3 \wedge \varphi_4)) \ ,$$

$$(\varphi_1 \wedge ((\varphi_2 \wedge \varphi_3) \wedge \varphi_4)) \ ,$$

$$(\varphi_1 \wedge (\varphi_2 \wedge (\varphi_3 \wedge \varphi_4))) \ .$$

DNF und KNF

Definition 2.41

(a) Ein Literal ist eine Formel der Gestalt X oder $\neg X$, wobei $X \in \mathrm{AS}$.
Im ersten Fall sprechen wir von einem positiven, im zweiten Fall von einem negativen Literal.

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

$$\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} \lambda_{i,j} \right)$$

hat, wobei $n, m_1, \ldots, m_n \geqslant 1$ sind und die $\lambda_{i,j}$ für alle $i \in [n]$ und $j \in [m_i]$ Literale sind. Die Subformeln $\kappa_i := \bigwedge_{j=1}^{m_i} \lambda_{i,j}$, für $i \in [n]$, nennen wir die (konjunktiven) Klauseln der Formel.

Beispiele:

- $(A_1 \wedge \neg A_2 \wedge A_3) \vee (\neg A_2 \wedge \neg A_3) \vee (A_2 \wedge A_1)$ ist in DNF
- $A_1 \vee \neg A_2 \vee A_3$ ist in DNF (mit $n = 3$ und $m_1 = m_2 = m_3 = 1$)
- $A_1 \wedge \neg A_2 \wedge A_3$ ist in DNF (mit $n = 1$ und $m_1 = 3$) und gleichzeitig ist diese Formel eine konjunktive Klausel

(c) Eine Formel ist in konjunktiver Normalform (KNF), wenn sie eine Konjunktion von Disjunktion von Literalen ist, d.h., wenn sie die Form

$$\bigwedge_{i=1}^{n} \Big(\bigvee_{j=1}^{m_i} \lambda_{i,j} \Big)$$

hat, wobei $n, m_1, \ldots, m_n \geqslant 1$ sind und die $\lambda_{i,j}$ für alle $i \in [n]$ und $j \in [m_i]$ Literale sind. Die Subformeln $\kappa_i := \bigvee_{j=1}^{m_i} \lambda_{i,j}$, für $i \in [n]$, nennen wir die (disjunktiven) Klauseln der Formel.

Beispiele:

- $(A_1 \vee \neg A_2 \vee A_3) \wedge (\neg A_2 \vee \neg A_3) \wedge (A_2 \vee A_1)$ ist in KNF
- $A_1 \vee \neg A_2 \vee A_3$ ist in KNF (mit $n = 1$ und $m_1 = 3$) und gleichzeitig ist diese Formel eine disjunktive Klausel
- $A_1 \wedge \neg A_2 \wedge A_3$ ist in KNF (mit $n = 3$ und $m_1 = m_2 = m_3 = 1$)

Normalformen spielen in vielen Anwendungsgebieten eine wichtige Rolle. Beispielsweise geht man in der Schaltungstechnik (Hardware-Entwurf) oft von DNF-Formeln aus, während bei der aussagenlogischen Modellbildung oftmals KNF-Formeln auftreten, da sich eine Sammlung von einfach strukturierten Aussagen sehr gut durch eine Konjunktion von Klauseln ausdrücken lässt.

Satz 2.42

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

Bemerkung 2.43

Der Beweis von Satz 2.42 zeigt Folgendes:
Um für eine gegebene Formel ψ eine äquivalente Formel φ in

- DNF zu erzeugen, können wir die Wahrheitstafel für ψ aufstellen und dann wie in Beispiel 2.33 vorgehen (bzw. $\varphi := A_1 \wedge \neg A_1$ setzen, falls ψ unerfüllbar ist).
- KNF zu erzeugen, können wir wie folgt vorgehen:
  (1) Stelle die Wahrheitstafel für ψ auf.
  (2) Falls in der letzten Spalte nur „1“en stehen, setze $\varphi := A_1 \vee \neg A_1$.
  (3) Ansonsten gehe wie folgt vor:
    - Betrachte alle Zeilen der Wahrheitstafel, bei denen in der letzten Spalte eine „0“ steht.
    - Für jede solche Zeile konstruiere die disjunktive Klausel, die von allen Interpretationen außer der zur Zeile gehörenden erfüllt wird.

      Beispiel: Wenn die Zeile der Wahrheitstafel die Form

      $$0\ 1\ 1 \mid 0$$

      hat, so gehört dazu die disjunktive Klausel

      $$A_1 \vee \neg A_2 \vee \neg A_3.$$

    - Bilde die Konjunktion all dieser disjunktiven Klauseln.
      Dies liefert die gesuchte KNF-Formel φ.

Wenn eine Formel sehr viele verschiedene Aussagensymbole enthält, die zur Formel gehörige Wahrheitstafel also sehr groß ist, ist das gerade beschriebene Verfahren zur Umformung in DNF oder KNF sehr zeitaufwändig.

In solchen Fällen ist es ratsam, stattdessen zu versuchen, die gewünschte Normalform durch Äquivalenzumformungen zu erzeugen.

Beispiel 2.44

Sei $\varphi := \Big(\big(\neg A_0 \wedge (A_0 \to A_1) \big) \vee (A_2 \to A_3) \Big)$.

Transformation von φ in NNF: siehe Tafel

Transformation in DNF: siehe Tafel

Transformation in KNF: siehe Tafel

Je nach Formel muss man ggf. die Distributivitätsregel mehrmals anwenden, bis man eine Formel der gewünschten Normalform erhält.

Ein DNF-Algorithmus

Eingabe: Formel $\varphi \in \mathrm{AL}(\{\neg, \wedge, \vee\})$ in NNF.
Ausgabe: Formel φ'' in DNF
Verfahren:

1. Wiederhole folgende Schritte:
2. Wenn φ in DNF ist, dann halte mit Ausgabe φ.
3. Ersetze eine Subformel von φ der Gestalt $(\psi_1 \wedge (\psi_2 \vee \psi_3))$ durch $((\psi_1 \wedge \psi_2) \vee (\psi_1 \wedge \psi_3))$ oder eine Subformel der Gestalt $((\psi_1 \vee \psi_2) \wedge \psi_3)$ durch $((\psi_1 \wedge \psi_3) \vee (\psi_2 \wedge \psi_3))$. Sei φ' die resultierende Formel.
4. $\varphi := \varphi'$.

Satz 2.45

Für jede Eingabeformel φ in NNF gibt der DNF-Algorithmus nach endlich vielen Schritten eine zu φ äquivalente Formel φ'' in DNF aus.

(hier ohne Beweis)

Analog kann man auch einen „KNF-Algorithmus“ angeben, der bei Eingabe einer NNF-Formel eine äquivalente Formel in KNF erzeugt (Details: Übung).

Eine kleine Formel mit großer DNF

Satz 2.46

Sei $n \in \mathbb{N}$ mit $n \geqslant 1$, seien $X_1, \ldots, X_n$ und $Y_1, \ldots, Y_n$ genau $2n$ verschiedene Aussagensymbole und sei

$$\varphi_n \;:=\; \bigwedge_{i=1}^{n} \left(X_i \vee \neg Y_i \right).$$

Jede zu φ_n äquivalente Formel in DNF hat mindestens 2^n konjunktive Klauseln.

Beweis: Übung

Korollar 2.47

Jeder Algorithmus, der bei Eingabe von beliebigen aussagenlogischen Formeln dazu äquivalente Formeln in DNF erzeugt, hat eine Laufzeit, die im worst-case exponentiell ist, d.h., $2^{\Omega(n)}$ bei Eingabe von Formeln der Länge n.

Abschnitt 2.5:

Der Endlichkeitssatz

Der Endlichkeitssatz (auch bekannt als Kompaktheitssatz)

Um nachzuweisen, dass eine gegebene *unendliche* Formelmenge erfüllbar ist, ist der folgende Satz sehr nützlich.

Satz 2.48 (Der Endlichkeitssatz der Aussagenlogik)

Für jede Formelmenge $\Phi \subseteq \mathsf{AL}$ *gilt:*

$$\Phi \textit{ ist erfüllbar} \iff \textit{Jede } \underline{\textit{endliche}} \textit{ Teilmenge von } \Phi \textit{ ist erfüllbar.}$$

Korollar 2.49 (Variante des Endlichkeitssatzes)

Sei $\Phi \subseteq \mathsf{AL}$ *und sei* $\psi \in \mathsf{AL}$*. Dann gilt:*

$$\Phi \models \psi \iff \textit{Es gibt eine endliche Teilmenge } \Gamma \textit{ von } \Phi\textit{, so dass } \Gamma \models \psi.$$

Anwendung: Färbbarkeit

Zur Erinnerung:

- Ein Graph $G = (V, E)$ besteht aus einer nicht-leeren Menge V von Knoten und einer Menge $E \subseteq \big\{\{x, y\} : x, y \in V,\ x \neq y\big\}$ von (ungerichteten) Kanten.
- Ein Subgraph eines Graphen $G = (V, E)$ ist ein Graph $H = (V', E')$ mit $V' \subseteq V$ und $E' \subseteq E$.
- Ein Graph ist endlich (bzw. unendlich), wenn seine Knotenmenge endlich (bzw. unendlich) ist.

Definition 2.50

Sei $k \in \mathbb{N}$ mit $k \geqslant 1$.
Eine k-Färbung eines Graphen $G = (V, E)$ ist eine Abbildung $f : V \to [k]$, so dass für alle Kanten $\{v, w\} \in E$ gilt: $f(v) \neq f(w)$.
G heißt k-färbbar, falls es eine k-Färbung von G gibt.

Satz 2.51

Sei $k \in \mathbb{N}$ mit $k \geqslant 1$.
Ein unendlicher Graph G mit Knotenmenge $\mathbb{N}$ ist genau dann k-färbbar, wenn jeder endliche Subgraph von G k-färbbar ist.

Abschnitt 2.6:

Resolution

Um nachzuweisen, dass eine gegebene KNF-Formel *unerfüllbar* ist, ist das im Folgenden vorgestellte Resolutionsverfahren nützlich.

Beispiel 2.52

Wir wollen nachweisen, dass die KNF-Formel

$$\varphi := (\neg P \vee \neg R) \wedge (P \vee \neg R) \wedge (\neg Q \vee S) \wedge (Q \vee R \vee T) \wedge \neg T \wedge (\neg S \vee R)$$

unerfüllbar ist. Dazu können wir wie folgt argumentieren:

Angenommen, eine Interpretation $\mathcal{I}$ erfüllt φ.

- Dann gilt $\mathcal{I} \models \neg T$.
- Aus $\mathcal{I} \models Q \vee R \vee T$ und $\mathcal{I} \models \neg T$ folgt dann $\mathcal{I} \models Q \vee R$.
- Aus $\mathcal{I} \models Q \vee R$ und $\mathcal{I} \models \neg Q \vee S$ folgt $\mathcal{I} \models R \vee S$.
- Aus $\mathcal{I} \models R \vee S$ und $\mathcal{I} \models \neg S \vee R$ folgt $\mathcal{I} \models R$.
- Aus $\mathcal{I} \models \neg P \vee \neg R$ und $\mathcal{I} \models P \vee \neg R$ folgt $\mathcal{I} \models \neg R$.

  Das ist ein *Widerspruch*. Somit ist φ *nicht* erfüllbar.

Umwandlung in kleine KNF-Formeln

Das Resolutionsverfahren, das wir im Folgenden vorstellen, funktioniert nur für KNF-Formeln.

Wir wissen bereits:

- Zu jeder Formel φ gibt es eine äquivalente Formel in KNF.
- Aber möglicherweise ist die kleinste zu φ äquivalente KNF-Formel exponentiell groß in der Größe von φ.

Wenn es uns nur um die Frage geht, ob eine Formel φ (un)erfüllbar ist, ist es aber auch gar nicht nötig, eine zu φ äquivalente KNF-Formel zu finden. Es reicht, eine zu φ erfüllbarkeitsäquivalente KNF-Formel zu konstruieren.

Definition 2.53

Zwei Formeln φ und ψ heißen erfüllbarkeitsäquivalent, falls gilt:

$$\varphi \text{ ist erfüllbar} \iff \psi \text{ ist erfüllbar.}$$

Eine beliebige Formel in eine *erfüllbarkeitsäquivalente* KNF-Formel umzuwandeln, ist in Linearzeit möglich.

Beispiel 2.54

Um die Formel

$$\varphi \;:=\; \big(\, P \to \neg Q \,\big) \;\vee\; \big(\, \neg \big(\, P \wedge Q \,\big) \;\wedge\; R \,\big)$$

in eine erfüllbarkeitsäquivalente KNF-Formel umzuformen, können wir wie folgt vorgehen.

Das Tseitin-Verfahren

Auf die gleiche Weise wie in Beispiel 2.54 können wir jede beliebige aussagenlogische Formel in eine erfüllbarkeitsäquivalente KNF-Formel umwandeln. Dieses Verfahren wird Tseitin-Verfahren genannt. Eine Laufzeitanalyse zeigt, dass das Tseitin-Verfahren in Linearzeit durchgeführt werden kann. Insgesamt erhalten wir so den folgenden Satz.

Satz 2.55

Zu jeder aussagenlogischen Formel φ gibt es eine aussagenlogische Formel φ_K mit folgenden Eigenschaften:

(a) *φ_K ist erfüllbarkeitsäquivalent zu φ.*

(b) *φ_K ist in 3-KNF, d.h., in KNF, wobei jede disjunktive Klausel aus höchstens 3 Literalen besteht (wir sagen: die Klauseln haben Länge $\leqslant 3$).*

(c) $|\varphi_K| = O(|\varphi|)$.

Außerdem gibt es einen Algorithmus, der φ_K bei Eingabe von φ in Linearzeit berechnet.

Notation

$|\varphi|$ bezeichnet die Länge (bzw. Größe) einer aussagenlogischen Formel φ, d.h. die Länge von φ aufgefasst als Wort über dem Alphabet A_{AL}.

Repräsentation von KNF-Formeln

Für den Rest diese Abschnitts werden wir nur noch KNF-Formeln betrachten, und wenn wir von Klauseln sprechen, meinen wir stets disjunktive Klauseln, also Disjunktionen von Literalen.

Für das Resolutionsverfahren ist die folgende Repräsentation von Klauseln und KNF-Formeln sehr hilfreich:

- Eine Klausel $(\lambda_1 \vee \cdots \vee \lambda_\ell)$, die aus Literalen $\lambda_1, \ldots, \lambda_\ell$ besteht, identifizieren wir mit der Menge $\{\lambda_1, \ldots, \lambda_\ell\}$ ihrer Literale.

  Beispiel: Wir schreiben z.B. $\{A_1, \neg A_2, A_3\}$ um die Klausel $(A_1 \vee \neg A_2 \vee A_3)$ zu bezeichnen.

  D.h.: Ab jetzt sind disjunktive Klauseln für uns dasselbe wie endliche Mengen von Literalen. Wenn wir von einer Klausel sprechen, meinen wir eine endliche Menge von Literalen und identifizieren diese mit der Formel, die aus der Disjunktion all dieser Literale besteht.

  Spezialfall: Die leere Menge $\emptyset$ entspricht der unerfüllbaren Formel $\mathbf{0}$ (die wiederum der „Formel“ entspricht, die aus der Disjunktion aller Literale aus $\emptyset$ besteht).

- Eine KNF-Formel $\varphi = \bigwedge_{i=1}^{m} \gamma_i$, die aus (disjunktiven) Klauseln $\gamma_1, \ldots, \gamma_m$ besteht, identifizieren wir mit der Menge $\Gamma := \{\gamma_1, \ldots, \gamma_m\}$ ihrer Klauseln.

  Offensichtlicherweise gilt für alle Interpretationen $\mathcal{I}$:

$$\mathcal{I} \models \varphi \iff \mathcal{I} \models \Gamma.$$

Beispiel: Die KNF-Formel $\varphi = A_1 \wedge (\neg A_2 \vee A_1) \wedge (A_3 \vee \neg A_2 \vee \neg A_1)$ repräsentieren wir durch die endliche Klauselmenge

$$\big\{\ A_1,\ \ (\neg A_2 \vee A_1),\ \ (A_3 \vee \neg A_2 \vee \neg A_1)\ \big\}$$

bzw. durch

$$\big\{\ \{A_1\},\ \ \{\neg A_2, A_1\},\ \ \{A_3, \neg A_2, \neg A_1\}\ \big\}$$

„Erfüllbarkeit von KNF-Formeln“ ist damit im Wesentlichen dasselbe Problem wie „Erfüllbarkeit von endlichen Mengen von Klauseln“.

Resolution

Notation

Für ein Literal λ sei

$$\overline{\lambda} := \begin{cases} \neg X\,, & \text{falls } \lambda \text{ von der Form } X \text{ für ein } X \in \mathsf{AS} \text{ ist} \\ X\,, & \text{falls } \lambda \text{ von der Form } \neg X \text{ für ein } X \in \mathsf{AS} \text{ ist.} \end{cases}$$

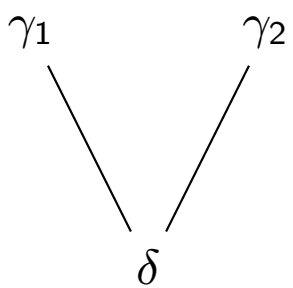
Wir nennen $\overline{\lambda}$ auch das Negat von λ.

Definition 2.56 (Resolutionsregel)

Seien γ_1, γ_2 und δ endliche Mengen von Literalen (d.h. disjunktive Klauseln). Dann ist δ eine Resolvente von γ_1 und γ_2, wenn es ein Literal λ gibt, so dass gilt:

$$\lambda \in \gamma_1, \qquad \overline{\lambda} \in \gamma_2 \qquad \text{und} \qquad \delta = \big(\gamma_1 \setminus \{\lambda\}\big) \cup \big(\gamma_2 \setminus \{\overline{\lambda}\}\big).$$

Graphische Darstellung:



„δ ist eine Resolvente von γ_1 und γ_2.“

Das Resolutionslemma

Notation

Ein Klausel ist eine endliche Menge von Literalen (eine solche Klausel repräsentiert die Disjunktion der in ihr enthaltenen Literale).
Eine Klauselmenge ist eine (endliche oder unendliche) Menge von Klauseln.

Lemma 2.57 (Resolutionslemma)

Sei Γ *eine Klauselmenge, seien* $\gamma_1, \gamma_2 \in \Gamma$ *und sei* δ *eine Resolvente von* γ_1 *und* γ_2*. Dann sind die Klauselmengen* Γ *und* $\Gamma \cup \{\delta\}$ *äquivalent.*

Resolutionsableitungen und -widerlegungen

Definition

Sei Γ eine Klauselmenge.

(a) Eine Resolutionsableitung einer Klausel δ aus Γ ist ein Tupel $(\delta_1, \ldots, \delta_\ell)$ von Klauseln, so dass gilt: $\ell \geqslant 1$, $\delta_\ell = \delta$, und für alle $i \in [\ell]$ ist

- $\delta_i \in \Gamma$, oder
- es gibt $j, k \in [i-1]$, so dass δ_i eine Resolvente von δ_j und δ_k ist.

(b) Eine Resolutionswiderlegung von Γ ist eine Resolutionsableitung der leeren Klausel aus Γ.

Zur Erinnerung:

Eine Klausel δ ist genau dann eine Resolvente zweier Klauseln γ_1 und γ_2, wenn es ein Literal λ gibt, so dass gilt:

$$\lambda \in \gamma_1, \qquad \overline{\lambda} \in \gamma_2 \qquad \text{und} \qquad \delta = \big(\gamma_1 \setminus \{\lambda\}\big) \cup \big(\gamma_2 \setminus \{\overline{\lambda}\}\big).$$

Notation 2.58

(a) Wir schreiben kurz $\Gamma \vdash_R \delta$ um auszudrücken, dass es eine Resolutionsableitung von δ aus Γ gibt.

Insbesondere bedeutet $\Gamma \vdash_R \emptyset$, dass es eine Resolutionswiderlegung von Γ gibt.

(b) An Stelle von $(\delta_1, \ldots, \delta_\ell)$ schreiben wir Resolutionsableitungen der besseren Lesbarkeit halber oft zeilenweise, also

$$\begin{array}{ll} (1) & \delta_1 \\ (2) & \delta_2 \\ \vdots & \\ (\ell) & \delta_\ell \end{array}$$

und geben am Ende jeder Zeile eine kurze Begründung an.

Beispiel 2.59

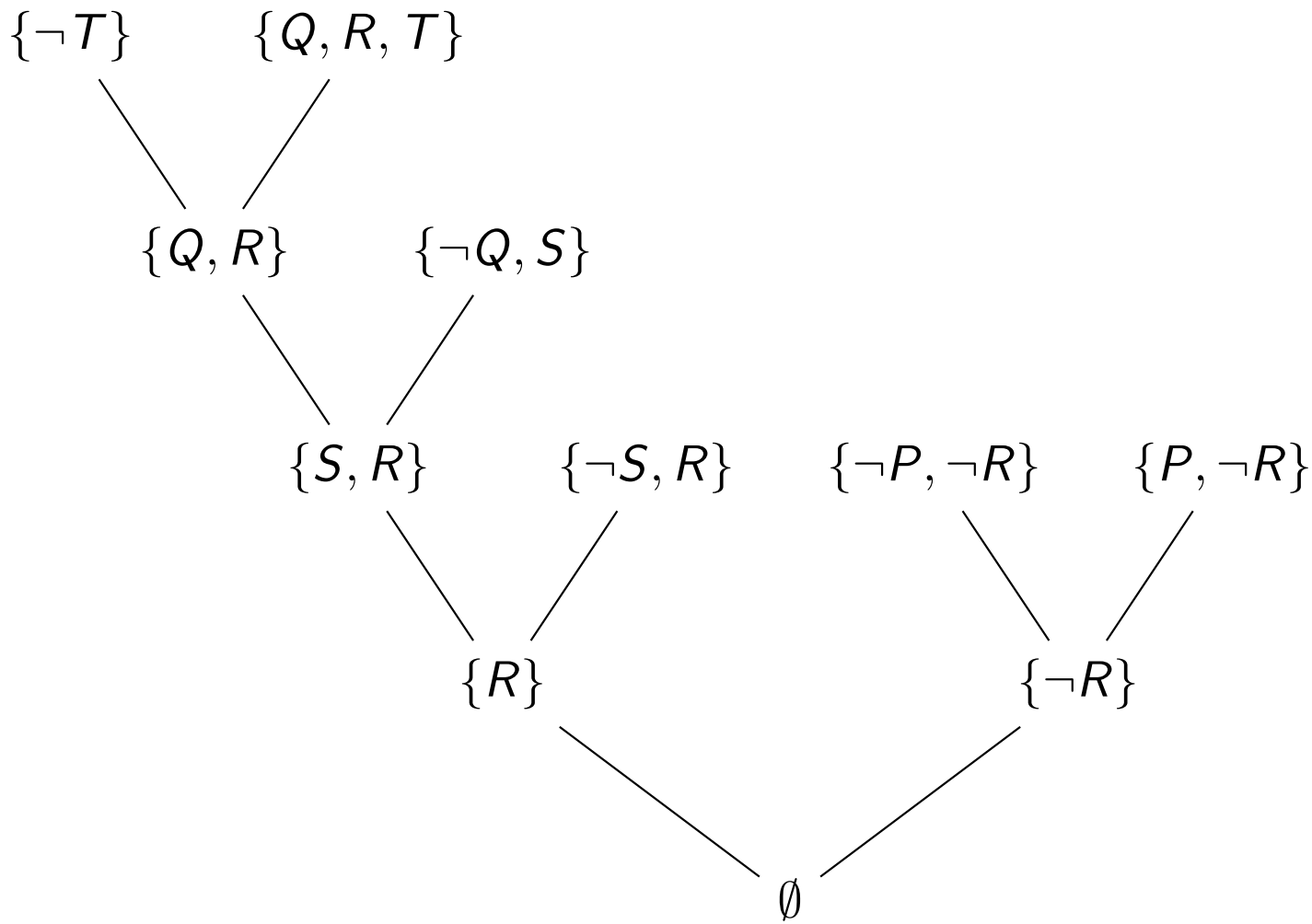
Sei

$$\Gamma := \{ \{\neg P, \neg R\}, \{P, \neg R\}, \{\neg Q, S\}, \{Q, R, T\}, \{\neg T\}, \{\neg S, R\} \}$$

Eine Resolutionswiderlegung von Γ ist:

(1)	$\{\neg T\}$	(in Γ)
(2)	$\{Q, R, T\}$	(in Γ)
(3)	$\{Q, R\}$	(Resolvente von (1), (2))
(4)	$\{\neg Q, S\}$	(in Γ)
(5)	$\{S, R\}$	(Resolvente von (3), (4))
(6)	$\{\neg S, R\}$	(in Γ)
(7)	$\{R\}$	(Resolvente von (5), (6))
(8)	$\{\neg P, \neg R\}$	(in Γ)
(9)	$\{P, \neg R\}$	(in Γ)
(10)	$\{\neg R\}$	(Resolvente von (8), (9))
(11)	$\emptyset$	(Resolvente von (7), (10))

Graphische Darstellung der Resolutionswiderlegung



Korrektheit und Vollständigkeit der Resolution

Satz 2.60

Für jede Klauselmenge Γ *gilt:*

$$\Gamma \vdash_R \emptyset \iff \Gamma \textit{ ist unerfüllbar.}$$

D.h.: Eine Klauselmenge hat genau dann eine Resolutionswiderlegung, wenn sie unerfüllbar ist.

Vorsicht

Beim Anwenden der Resolutionsregel (Definition 2.56) darf immer nur ein Literal λ betrachtet werden.

Beispiel:

Betrachte die Klauselmenge $\Gamma := \{\gamma_1, \gamma_2\}$ mit $\gamma_1 := \{X, Y\}$ und $\gamma_2 := \{\neg X, \neg Y\}$ (wobei X und Y zwei verschiedene Ausagensymbole sind).

Der Satz von Haken

Für eine endliche Klauselmenge Γ sei die *Größe* von Γ die Zahl

$$\|\Gamma\| \;:=\; \sum_{\gamma\in\Gamma} |\gamma|,$$

wobei $|\gamma|$ die Anzahl der Literale in γ bezeichnet.

Der folgende (schwer zu beweisende) Satz zeigt, dass es im Worst-Case exponentiell lange dauern kann, eine Resolutionswiderlegung zu finden.

Satz 2.61 (Satz von Haken, 1985)

Es gibt Konstanten $c, d > 0$ und endliche Klauselmengen Γ_n für $n \geqslant 1$, so dass für alle $n \in \mathbb{N}$ mit $n \geqslant 1$ gilt:

1. $\|\Gamma_n\| \leqslant n^c$,
2. Γ_n *ist unerfüllbar, und*
3. *jede Resolutionswiderlegung von* Γ_n *hat Länge* $\geqslant 2^{dn}$.

(Hier ohne Beweis)

Abschnitt 2.7:

Erfüllbarkeitsalgorithmen

Das aussagenlogische Erfüllbarkeitsproblem

Wir betrachten im Folgenden Algorithmen für das

Aussagenlogische Erfüllbarkeitsproblem:

Eingabe:	eine Formel $\varphi \in \mathrm{AL}$
Ausgabe:	„erfüllbar“, falls φ erfüllbar ist;
	„unerfüllbar“, sonst.

Notation

Im Folgenden bezeichnet n immer die Anzahl der in φ vorkommenden verschiedenen Aussagensymbole, und $m := |\varphi|$ bezeichnet die Länge von φ (aufgefasst als Wort über dem Alphabet der Aussagenlogik).

Varianten des Erfüllbarkeitsproblems

Berechnen einer erfüllenden Interpretation:

Zusätzlich soll bei erfüllbaren Formeln $\varphi(A_1, \ldots, A_n)$ noch ein Modell berechnet werden, d.h., ein Tupel $(b_1, \ldots, b_n) \in \{0,1\}^n$, so dass $\varphi[b_1, \ldots, b_n] = 1$.

Einschränkung auf KNF-Formeln:

Oft beschränkt man sich auf Eingabeformeln in KNF oder sogar 3-KNF. Das ist keine wesentliche Einschränkung, weil sich mit Hilfe des Tseitin-Verfahrens jede Formel in Linearzeit in eine erfüllbarkeitsäquivalente Formel in 3-KNF transformieren lässt (Satz 2.55).
Das Erfüllbarkeitsproblem für Formeln in KNF bzw. 3-KNF bezeichnet man mit SAT bzw. 3-SAT.

Komplexität des Erfüllbarkeitsproblems

Satz 2.62 (Satz von Cook und Levin, ≈1971)

Das aussagenlogische Erfüllbarkeitsproblem (und sogar die Einschränkung 3-SAT) ist NP-vollständig.

Die Komplexitätsklassen P und NP, der Begriff der NP-Vollständigkeit, sowie ein Beweis des Satzes von Cook und Levin werden in der Vorlesung *Einführung in die Theoretische Informatik* behandelt.

Bemerkung

- Wenn also P $\neq$ NP ist (was allgemein vermutet wird), gibt es für das aussagenlogische Erfüllbarkeitsproblem keinen Polynomialzeitalgorithmus.
- Man vermutet sogar, dass es eine Konstante $c > 1$ gibt, so dass jeder Algorithmus für 3-SAT eine worst-case Laufzeit von $\Omega(c^n)$ hat.
  Diese Vermutung ist unter dem Namen „Exponential Time Hypothesis“ (ETH) bekannt.
- Der im Worst-Case beste derzeit bekannte Algorithmus für 3-SAT hat eine Laufzeit von etwa $O(1.4^n)$.

Der Wahrheitstafelalgorithmus

Lemma 2.63

Es gibt einen Linearzeitalgorithmus, der bei Eingabe einer Formel $\varphi(A_1, \ldots, A_n) \in \mathrm{AL}$ *und eines Tupels* $(b_1, \ldots, b_n) \in \{0,1\}^n$ *den Wert* $\varphi[b_1, \ldots, b_n]$ *berechnet.*

Beweis: Übung.

Der folgende Algorithmus löst das aussagenlogische Erfüllbarkeitsproblem.

Wahrheitstafelalgorithmus

Eingabe: eine Formel $\varphi \in \mathrm{AL}$

1. Berechne die Wahrheitstafel für φ.
2. Falls in der letzten Spalte mindestens eine 1 auftaucht, gib „erfüllbar“ aus, sonst gib „unerfüllbar“ aus.

Laufzeit: $O(m \cdot 2^n)$ (sogar im „Best-Case“)
Speicherplatz: $O(m + 2^n)$ . . . bei zeilenweiser Auswertung: $O(m + n)$

Der Resolutionsalgorithmus

Der Resolutionsalgorithmus probiert einfach alle möglichen Resolutionsableitungen durch und testet so, ob es eine Resolutionswiderlegung gibt (d.h. die Klauselmenge unerfüllbar ist).

Resolutionsalgorithmus

Eingabe: eine endliche Klauselmenge Γ (entspricht einer KNF-Formel)

1. Wiederhole, bis keine neuen Klauseln mehr generiert werden:
   Füge alle Resolventen aller Klauseln aus Γ zu Γ hinzu.
2. Falls $\emptyset \in \Gamma$, gib „unerfüllbar“ aus, sonst gib „erfüllbar“ aus.

Laufzeit: $2^{O(n)}$ (weil es bei n Aussagensymbolen 4^n verschiedene Klauseln gibt).

Speicherplatz: $2^{O(n)}$

Mit geschicktem „vergessen“ nicht mehr benötigter Klausen ist auch Speicherplatz $O(m+n)$ möglich.

Der Davis-Putnam-Logemann-Loveland Algorithmus

Der DPLL-Algorithmus ist ein in der Praxis sehr erfolgreicher Algorithmus, der die Wahrheitstafelmethode mit Resolution kombiniert.

Ähnlich wie bei dem Wahrheitstafelalgorithmus durchsucht der DPLL-Algorithmus systematisch den Raum aller möglichen Interpretationen und testet, ob diese die gegebene Klauselmenge erfüllen. Resolution wird dabei dazu verwendet, die Suche geschickt zu steuern und Dinge, die während der Suche bereits über die Klauselmenge „gelernt“ wurden, weiterzuverwenden.

Der DPLL-Algorithmus ist die Basis moderner SAT-Solver, die Klauselmengen, die aus Millionen von Klauseln und Hunderttausenden von Aussagensymbolen bestehen, auf Erfüllbarkeit testen können.

DPLL-Algorithmus

Eingabe: eine endliche Klauselmenge Γ (entspricht einer KNF-Formel)

1. Vereinfache Γ. *% Details dazu: siehe nächste Folie*
2. Falls $\Gamma = \emptyset$, gib „erfüllbar“ aus.
3. Falls $\emptyset \in \Gamma$, gib „unerfüllbar“ aus.
4. Wähle ein Literal λ.
5. *% probiere aus, ob Γ ein Modell hat, bei dem das Literal λ erfüllt wird:*
   Löse rekursiv $\Gamma \cup \big\{\{\lambda\}\big\}$. Falls dies erfüllbar ist, gib „erfüllbar“ aus.
6. *% probiere aus, ob Γ ein Modell hat, bei dem das Literal $\overline{\lambda}$ erfüllt wird:*
   Löse rekursiv $\Gamma \cup \big\{\{\overline{\lambda}\}\big\}$. Falls dies erfüllbar ist, gib „erfüllbar“ aus. Sonst gib „unerfüllbar“ aus.

Vereinfachungsheuristiken, die in Schritt 1. angewendet werden:

- Unit Propagation: Für alle „Einerklauseln“ $\{\lambda\} \in \Gamma$ (wobei λ ein Literal ist), bilde alle Resolventen von $\{\lambda\}$ mit anderen Klauseln und streiche anschließend alle Klauseln, die λ enthalten. Wiederhole dies, so lange es Einerklauseln gibt.

  Präzise:
  Für jede „Einerklausel“ $\{\lambda\} \in \Gamma$ tue Folgendes:
  1. Ersetze jede Klausel $\gamma \in \Gamma$ durch die Klausel $\gamma \setminus \{\overline{\lambda}\}$.
  2. Entferne aus Γ jede Klausel, die das Literal λ enthält.

  Wiederhole dies, so lange es in Γ Einerklauseln gibt.

- Pure Literal Rule: Literale λ, deren Negat $\overline{\lambda}$ nirgendwo in der Klauselmenge auftaucht, können auf 1 gesetzt werden. Alle Klauseln, die ein solches Literal enthalten, sind dann wahr und können gestrichen werden.

- Streiche Klauseln, die Obermengen von anderen Klauseln sind (dies ist allerdings ineffizient und wird in der Praxis zumeist weggelassen).

Man sieht leicht, dass der DPLL-Algorithmus stets die korrekte Antwort gibt (d.h., er terminiert immer, und er gibt genau dann „erfüllbar“ aus, wenn die eingegebene Klauselmenge Γ erfüllbar ist).

Laufzeit des DPLL-Algorithmus:

$O(m \cdot 2^n)$ im Worst-Case, in der Praxis aber häufig sehr effizient.

Beispiel 2.64

Sei $\Gamma :=$

$$\Big\{ \{X_1, \neg X_5, \neg X_6, X_7\},\ \{\neg X_1, X_2, \neg X_5\},\ \{\neg X_1, \neg X_2, \neg X_3, \neg X_5, \neg X_6\},$$
$$\{X_1, X_2, \neg X_4, X_7\},\ \{\neg X_4, \neg X_6, \neg X_7\},\ \{X_3, \neg X_5, X_7\},$$
$$\{X_3, \neg X_4, \neg X_5\},\ \{X_5, \neg X_6\},\ \{X_5, X_4, \neg X_8\},$$
$$\{X_1, X_3, X_5, X_6, X_7\},\ \{\neg X_7, X_8\},\ \{\neg X_6, \neg X_7, \neg X_8\} \Big\}$$

DPLL auf unerfüllbaren Klauselmengen

DPLL und Resolution

Wenn der DPLL-Algorithmus auf einer unerfüllbaren Formelmenge Γ nach N Vereinfachungs- und Rekursionsschritten *unerfüllbar* ausgibt,
dann gibt es eine Resolutionswiderlegung der der Länge $O(|\Gamma| + N)$.

(Hier ohne Beweis)

- Der Suchbaum des DPLL-Algorithmus kann in den Ableitungsbaum der Resolutionswiderlegung überführt werden.
- Damit sind Klauselmengen, die eine lange Resolutionswiderlegung benötigen (wie in Satz 2.61) auch schwer für den DPLL-Algorithmus.

SAT-Solver

Moderne SAT-Solver erweitern DPLL auf vielfältige Weise, insbesondere durch das geschickte Hinzufügen zusätzlicher Resolventen der aktuellen Klauselmenge, geeignete Suchheuristiken und das gelegentliche Neustarten der Suche.

- Aus dem Lauf gängiger SAT-Solver auf unerfüllbaren Instanzen lassen sich ebenfalls Resolutionswiderlegungen generieren. Diese sind aber nicht notwendigerweise *baumartig* (wie es bei DPLL der Fall ist).

Abschnitt 2.8:

Hornformeln

Hornklauseln und Hornformeln

Hornformeln sind spezielle aussagenlogische Formeln, die die Basis der logischen Programmierung bilden, und für die das Erfüllbarkeitsproblem effizient gelöst werden kann.

Definition 2.65

Eine Hornklausel ist eine disjunktive Klausel, in der höchstens ein positives Literal vorkommt.
Eine Hornformel ist eine Konjunktion endlich vieler Hornklauseln.

Beispiele

- $\{\neg X, \neg Y, \neg Z\}$ (bzw. $\neg X \vee \neg Y \vee \neg Z$) ist eine Hornklausel.
- $\{\neg X, \neg Y, Z\}$ (bzw. $\neg X \vee \neg Y \vee Z$) ist eine Hornklausel.
- $\{\neg X, Y, Z\}$ (bzw. $\neg X \vee Y \vee Z$) ist keine Hornklausel.
- $\{X\}$ (bzw. X) ist eine Hornklausel.
- $\emptyset$ ist eine Hornklausel.
- $(X \vee \neg Y) \wedge (\neg Z \vee \neg X \vee \neg Y) \wedge Y$ ist eine Hornformel.

Hornklauseln als Implikationen

- Eine Hornklausel der Form $\{\neg X_1, \ldots, \neg X_{n-1}, X_n\}$ (bzw. $\neg X_1 \vee \cdots \vee \neg X_{n-1} \vee X_n$) ist äquivalent zur Formel

$$(X_1 \wedge \cdots \wedge X_{n-1}) \to X_n.$$

  Solche Klauseln werden auch „Regeln“ (oder „Prozedurklauseln“) genannt.

- Eine Hornklausel der Form $\{\neg X_1, \ldots, \neg X_{n-1}\}$ ist äquivalent zur Formel

$$(X_1 \wedge \ldots \wedge X_{n-1}) \to \mathbf{0}.$$

  Solche Klauseln werden auch „Zielklauseln“ (oder „Frageklauseln“) genannt.

- Eine Hornklausel der Form $\{X_1\}$ ist äquivalent zur Formel

$$\mathbf{1} \to X_1.$$

  Solche Klauseln werden auch „Tatsachenklausel“ genannt.

- Die leere (Horn-)Klausel $\emptyset$ ist unerfüllbar und daher äquivalent zur Formel

$$\mathbf{1} \to \mathbf{0}.$$

Der Streichungsalgorithmus

Der folgende Algorithmus löst das Erfüllbarkeitsproblem für Hornformeln in Polynomialzeit.

Wir geben zunächst den Algorithmus an, betrachten dann Beispielläufe davon, analysieren die Laufzeit und zeigen danach, dass der Algorithmus korrekt ist, d.h. stets die richtige Antwort gibt.

Streichungsalgorithmus

Eingabe: eine endliche Menge Γ von Hornklauseln

1. Wiederhole:
2. Falls $\emptyset \in \Gamma$, so halte mit Ausgabe „unerfüllbar“.
3. Falls Γ keine Tatsachenklausel (d.h. Klausel $\{X\}$ mit $X \in \mathrm{AS}$) enthält, so halte mit Ausgabe „erfüllbar“.
   % Γ wird erfüllt, indem jedes Aussagensymbol mit 0 belegt wird
4. Wähle eine Tatsachenklausel $\{X\} \in \Gamma$.
   % Idee: Um Γ zu erfüllen, muss X mit dem Wert 1 belegt werden
5. Streiche $\neg X$ aus allen Klauseln $\delta \in \Gamma$, die das Literal $\neg X$ enthalten.
   % Wenn X den Wert 1 hat, trägt $\neg X$ nichts zum Erfüllen einer Klausel bei
6. Streiche aus Γ alle Klauseln $\delta \in \Gamma$, die das Literal X enthalten (d.h. entferne aus Γ alle $\delta \in \Gamma$, für die gilt: $X \in \delta$).
   % Wenn X den Wert 1 hat, sind solche Klauseln erfüllt

Beispiele 2.66

Wir wenden den Streichungsalgorithmus auf die beiden folgenden Mengen von Hornklauseln an.

(a) $\Gamma_a := \{\ S \to \mathbf{0},\ \ (P \wedge Q) \to R,\ \ (S \wedge R) \to \mathbf{0},\ \ (U \wedge T \wedge Q) \to P,$
$(U \wedge T) \to Q,\ \ \mathbf{1} \to U,\ \ \mathbf{1} \to T\ \}$

(b) $\Gamma_b := \{\ (Q \wedge P) \to T,\ \ (U \wedge T \wedge Q) \to R,\ \ (U \wedge T) \to Q,$
$\mathbf{1} \to U,\ \ R \to \mathbf{0},\ \ \mathbf{1} \to T\ \}$

Laufzeit des Streichungsalgorithmus

Man sieht leicht, dass in jedem Schleifendurchlauf die Anzahl der Klauseln in Γ kleiner wird. Daher terminiert der Algorithmus nach maximal m Schleifendurchläufen, wobei m die Anzahl der Klauseln in der Eingabemenge Γ ist.

In jedem einzelnen Schleifendurchlauf betrachtet der Algorithmus alle Klauseln der aktuellen Klauselmenge und führt dabei $O(n)$ Schritte durch, wobei $n = \|\Gamma\|$ die Größe der Klauselmenge ist.

Insgesamt terminiert der Streichungsalgorithmus also nach $O(m{\cdot}n)$ Schritten, d.h. in Zeit polynomiell in der Größe von Γ.

Satz 2.67

Die Laufzeit des Streichungsalgorithmus ist $O(m{\cdot}n)$, wobei $m = |\Gamma|$ die Anzahl der Hornklauseln in der eingegebenen Menge Γ und $n = \|\Gamma\|$ die Größe von Γ ist.

Bemerkung

Eine Variante des Streichungsalgorithmus läuft sogar in Linearzeit, d.h. in Zeit $O(n)$.

Der Streichungsalgorithmus und Resolution

Lemma 2.68

Sei Γ_0 eine endliche Menge von Hornklauseln und δ eine Klausel, die zu irgendeinem Zeitpunkt während des Laufs des Streichungsalgorithmus bei Eingabe Γ_0 in der vom Algorithmus gespeicherten Menge Γ liegt. Dann gilt: $\Gamma_0 \vdash_R \delta$.

Korrektheit des Streichungsalgorithmus

Satz 2.69

Der Streichungsalgorithmus ist korrekt.

Das heißt, bei Eingabe einer endlichen Menge Γ_0 von Hornklauseln hält der Algorithmus mit Ausgabe „erfüllbar“, falls Γ_0 erfüllbar ist, und mit Ausgabe „nicht erfüllbar“, falls Γ_0 unerfüllbar ist.

Semesterüberblick

1. Einleitung — heute
2. Aussagenlogik — Woche 1–6
   Syntax und Semantik, Normalformen, Modellierung, Resolution, Erfüllbarkeitsalgorithmen
3. Logik erster Stufe — Woche 7–10
   Syntax und Semantik, Normalformen, Modellierung, Nichtausdrückbarkeit
4. Grundlagen des automatischen Schließens — Woche 11–14
   Sequenzenkalkül, Vollständigkeits- und Endlichkeitssatz, Grenzen der Berechenbarkeit, automatische Theorembeweiser
5. Logik-Programmierung — Woche 15–16
   theoretische Grundlagen der Logik-Programmierung

Learn Prolog Now! Einführung in Prolog findet semesterbegleitend statt.