Theoretische Grundlagen der Informatik 3
Logik und Kalküle

Stephan Kreutzer

Wintersemester 2012/2013

1. Einleitung

Einleitung

Inhalt.

Die Vorlesung ist eine Einführung in die Logik im Kontext der Informatik.

Fragen.

1. Was ist überhaupt Logik?
2. Warum wollen Sie das lernen und warum wollen wir, dass Sie das lernen?

Was ist Logik?

Logik. "Logik" kommt vom altgriechischen Wort "logos", die Vernunft.

In der Umgangssprache bezeichnet Logik vernünftiges oder richtiges Schließen oder Denken.

Die Logik hat ihren Ursprung in der (griechischen) Philosophie, in der Frage, was richtiges Argumentieren eigentlich bedeutet, d. h. was man aus einer Aussage folgerichtig ableiten kann.

In dieser Vorlesung beschäftigen wir uns vor allem mit formaler, oder mathematischer Logik.

Logische Sprachen oder Systeme

Im Zentrum der Betrachtungen stehen logische Systeme oder Logiken.

Ein logisches System besteht aus:

Syntax. Eine formal definierte Ansammlung von Formeln. Die Syntax bestimmt, was eine korrekte Formel der Logik sein soll.

Beispiel: $\forall x \exists y \big(x < y \wedge \mathit{prim}(y)\big)$

Semantik. Die Semantik gibt die Bedeutung der einzelnen Formeln an. Zum Beispiel, ob obige Formel in den natürlichen Zahlen $(\mathbb{N}, <, +, *, \mathit{prim})$ gilt.

Vergleiche Programmiersprachen wie C oder Java. Die Syntax gibt an, wann ein Programm korrekt compiliert werden kann. Die Semantik definiert dann, was das Programm eigentlich bedeuten soll.

Anwendungen der Logik in der Informatik

Einige ausgewählte Anwendungen der Logik in der Informatik.

1. Logiken als Abfragesprachen für Datenbanken
2. Logiken als Spezifikationsmechanismen in der automatischen Verifikation von Schaltkreisen, Protokollen und Programmen
3. Logik in der künstlichen Intelligenz
4. Semantik von Programmiersprachen
5. ....

Beispiel: Datenbanken

Relationale Datenbanken

Relationale Datenbanken.

Eines der verbreitetsten Datenmodelle in der Praxis.

Datenbanken bestehen in diesem Modell aus Tabellen bzw. Relationen.

Beispiel. Filmdatenbank wie z.B. IMDB

`Schauspieler`(Vorname, Name, Geburtsjahr)

`Film`(Titel, Regisseur, Entstehungsjahr)

`Cast`(Filmtitel, Schauspieler)

...

Datenbanken

Beispiel.

Schauspieler

Kirstin	Dunst	1982
George	Clooney	1961
...	...	...

Film

Melancholia	Lars von Trier	2011
Good Night, and Good Luck	George Clooney	2007
...	...	...

Cast

Melancholia	Kirsten Dunst	
Good Night, and Good Luck	George Clooney	
...	...	...

Datenbankanfragen. Um Informationen aus Datenbanken zu bekommen stellen wir Datenbankanfragen.

Beispiele. Liste alle Filme von “Lars von Trier” auf.

Gibt es einen Film, in dem George Clooney und Kirstin Dunst mitspielen?

Datenbanken

Datenbankanfragesprachen.

Um solche Anfragen stellen zu können, brauchen wir eine Sprache, in der diese Dinge formuliert werden können.

Wichtige Eigenschaften.

- Anfragen müssen präzise und eindeutig formuliert werden können.

  Natürliche Sprache wegen Doppeldeutigkeiten ungeeignet.

  $\rightsquigarrow$ Logik als Basis von Anfragesprachen.

  Beispiel: SQL = Prädikatenlogik plus Aggregation, Zählen etc.

- In der Logik formulierte Anfragen müssen schnell ausgewertet werden können.

  $\rightsquigarrow$ Auswertungsproblem einer Logik.

Beispiel: Automatische Verifikation

Automatische Verifikation

Schaltkreisverifikation. Eine wichtige industrielle Anwendung der Logik ist die Schaltkreisverifikation.

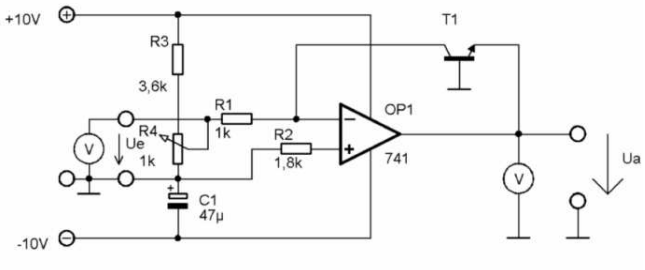
Ziel ist ein Beweis, dass ein gegebener Schaltkreis “korrekt” ist.

Beispiel Flugzeugsicherheitssysteme. “Das Flugzeug darf nie unter 60 Meter Flughöhe sinken, ohne dass das Fahrwerk ausgefahren wird.”

Dazu müssen zunächst die geforderten Eigenschaften des Schaltkreises exakt und formal spezifiziert werden.

Model-Checking

Schaltkreis



Korrektheitseigenschaft

``Das Flugzeug darf nie unter 60 Meter Flughöhe sinken, ohne dass das Fahrwerk ausgefahren wird."

Modell des Schaltkreises *erfüllt?* **Logische Formel**

Wichtige Eigenschaften der verwendeten Logiken.

- Typische Korrektheitseigenschaften müssen formalisierbar sein.
- Formeln müssen effizient auswertbar sein.

Beispiel: Wissensrepräsentation in der künstlichen Intelligenz

Wissensrepräsentation

Ziel. Das Wissen von Experten in bestimmten Bereichen zu speichern und für andere nutzbar und zugänglich zu machen.

Beispiel. Spezialwissen in der Medizin, Biologie, etc.

Wissensrepräsentation. Speicherung von Expertenwissen in Wissensbasen.

Beispiel. Adler $\sqsubseteq$ Vogel $\sqsubseteq$ endothermer Organismus

Dazu werden sogenannte Beschreibungslogiken verwendet.

Wichtige Eigenschaften der verwendeten Logiken.

- Exakte Formalisierung von Wissen möglich.
- Neue Schlüsse aus dem formalisierten Wissen sollen automatisch hergeleitet werden können.

Zusammenfassung

Anwendungen der Logik in der Informatik.

- Logiken werden als Beschreibungs-, Abfrage- oder Spezifikationsformalismen verwendet.
- Wichtige Eigenschaft ist, dass sie exaktes, eindeutiges Formalisieren von Informationen erlauben.
- Verschiedene Anwendungen verlangen oft verschiedenartige Logiken.

  $\rightsquigarrow$ eine Vielzahl untersuchter Logiken in der Informatik.

  Das steht in gewissem Gegensatz zur mathematischen Logik, in der nur wenige Logiken untersucht werden.

Logik in der Informatik

Zentrale Aspekte der Logik in der Informatik.

- Welche Arten von Logiken gibt es?
- Was können wir in einer bestimmten Logik ausdrücken?
- Wie können wir zeigen, dass etwas nicht ausgedrückt werden kann?
- Kann man Aussagen, die in einer bestimmten Logik formalisiert wurden, automatisch beweisen?
- Wie schwer ist es, eine Formel einer bestimmten Logik, in einem Modell auszuwerten?

1.2. Organisatorisches

Organisatorisches

Vorlesung. Donnerstags, 10-12 Uhr MA 001

Materialien.

- Die Vorlesung wird zum Teil auf Folien und zum Teil an der Tafel gehalten.
- Ein vollständiger Foliensatz inklusive des Teils an der Tafel wird online zur Verfügung gestellt.
- Allerdings werden in der Vorlesung auch Beispiele gerechnet und Dinge motiviert, die nicht auf den Folien stehen.
- Prüfungsrelevant ist alles, was in der Vorlesung besprochen wurde (egal ob auf Folie oder nicht).

Literaturhinweise.

- Logik für Informatiker, M. Kreuzer und S. Kühling, Pearson
- Mathematisch-strukturelle Grundlagen der Informatik, H. Ehrig et. al, Springer Verlag
- Einführung in die mathematische Logik, H.-D. Ebbinghaus, J. Flum und W. Thomas, Spektrum

Organisatorisches: Materialien

ISIS-System.

- Vorlesungsmaterialien und Übungsblätter werden über das ISIS-System zur Verfügung gestellt.
- Über dieses System verschicken wir auch Ankündigungen, Sie sollten sich also anmelden.

Foren. Wir haben zwei Nachrichtenforen im ISIS System eingerichtet.

1. Nachrichtenforum für uns, in dem wir Ankündigungen etc. verschicken.
2. Diskussionsforum für Sie, in dem Sie den Stoff der Vorlesung, Organisationsfragen etc. diskutieren können. Wir lesen das unregelmäßig mit,

   d.h. Sie sollten nicht unbedingt darauf vertrauen, dass Sie auf dort gestellte Fragen auch schnell eine Antwort von uns bekommen.

Organisatorisches: Übungen

Übungsblätter.

- Es wird wöchentlich ein Übungsblatt in der Vorlesung ausgeteilt. Beginn: Woche 2.
- Sie haben eine Woche Zeit zur Bearbeitung und geben es dann wieder **nach der Vorlesung** ab.

  Alternativ: bis 10.00 Uhr s.t. im Sekretariat, Raum TEL 711.
- Die Übungen sollen in Gruppen zu maximal 3 Studierenden gelöst werden.
- Es werden 50% der Übungspunkte zur Klausurzulassung benötigt.

Wichtig!!! Auf jedes Übungsblatt müssen Sie Ihren Tutor sowie Tag und Zeit Ihres Tutoriums schreiben!

Wir können Ihre Abgabe sonst nicht zuordnen und nicht korrigieren.

Organisatorisches: Tutorien

Tutorien.

Montag 10-12 Uhr,	EN 187,	Sebastian
Montag 12-14 Uhr,	MA 545,	Friederike
Montag 14-16 Uhr,	H 3008,	Viktor
Montag 16-18 Uhr,	H 3008,	Friederike
Dienstag 10-12 Uhr,	H 3008,	Christoph
Dienstag 14-16 Uhr,	A 060,	Christoph
Dienstag 14-16 Uhr,	MA 144,	Stephan Kreutzer
Dienstag 16-18 Uhr,	EN 185,	Christoph
Mittwoch 10-12 Uhr,	H 3021,	Tobi
Mittwoch 12-14 Uhr,	MA 841,	Tobson
Mittwoch 14-16 Uhr,	H 3008,	Sebastian
Mittwoch 14-16 Uhr,	MA 545,	Viktor

Organisatorisches: Sprechzeiten etc.

Sprechzeiten. Dienstags, 14-15 Uhr, sowie nach Vereinbarung.

Kontakt.

- Raum TEL 712
- EMail: stephan.kreutzer@tu-berlin.de
- Webseite: logic.las.tu-berlin.de

Klausur

Prüfung. Das Modul TheGI 3 wird mit einer Klausur abgeschlossen.

Zur Klausurzulassung müssen Sie 50% der Übungspunkte erreichen.

Ausnahme. Wenn Sie schon bereits eine TheGI 3 Klausur nicht bestanden haben, Sie also eine Widerholungsklausur schreiben, gilt Ihre alte Klausurzulassung weiter.

Klausurtermine.

1. Die Klausur wird am 28.02.2013 stattfinden.
2. Sollten Studierende die Klausur nicht bestehen, werden wir eine Wiederholungsklausur anbieten (oder mündliche Prüfungen).

   Diese würde am 02.04.2013 stattfinden.

2. Aussagenlogik

2.1. Einleitung

Einleitung

Beispiel. Betrachten wir folgende Aussage.

- Wenn der Zug zu spät ist und keine Taxis am Bahnhof stehen, kommt Peter zu spät zu seiner Verabredung.
- Peter kam nicht zu spät zu seiner Verabredung.
- Der Zug hatte Verspätung.

Also standen Taxis am Bahnhof.

Ist das Argument gültig?

Einleitung

Beispiel. Sie kann nicht zu hause sein, da sie entweder an Bord oder zu hause ist und ich gerade gehört habe, dass sie an Bord ist.

Ist das Argument gültig?

Was können wir formal beweisen?

- Klar formulierte Aussagen, die entweder richtig oder falsch sind.

  Die Bedeutung aller verwendeten Ausdrücke muss bekannt sein.

  Ebenso das vorausgesetzte Hintergrundwissen.
- Natürliche Sprache ist dafür nicht gut geeignet.
- Wir werden daher formale Sprachen, oder Logiken, verwenden, in denen alle Ausdrücke formal und vollständig definiert sind.
- Ziel ist es, allgemeine Regeln für korrektes Schließen herleiten zu können, möglichst sogar automatisch.

Beispiel: Syllogismen

Korrektheit folgender Aussagen folgt aus rein formalen Gründen.

Beispiel.

Annahme 1: **Alle** Menschen **sind** sterblich.

Annahme 2: Sokrates **ist ein** Mensch.

Folgerung: **Also ist** Sokrates sterblich.

Diese und verwandte Arten von Schlüssen werden Syllogismen genannt.

Abstrakte Form.

Annahme 1: **Alle** A **sind** B.

Annahme 2: C **ist ein** A.

Folgerung: **Also ist** C B.

Beispiel: Syllogismen

Beispiel.

Annahme 1: **Alle** Ersetzungschiffren **sind** anfällig für brute-force Angriffe.

Annahme 2: Der Caesar Chiffre **ist ein** Ersetzungschiffre.

Folgerung: **Also ist** der Caesar Chiffre anfällig für brute-force Angriffe.

Einleitung

In diesem Teil der Vorlesung werden wir Methoden kennen lernen um

- Aussagen wie auf den vorigen Folien formal auszudrücken
- formalisierte Aussagen zu manipulieren
- logische Behauptungen zu beweisen

2.2. Syntax und Semantik der Aussagenlogik

Propositionen

- Propositionen sind Aussagen die entweder wahr oder falsch sein können

  Beispiel. A : Die Sonne scheint B : Die Vorlesung ist langweilig

- Propositionen können durch Verknüpfungen kombiniert werden, z. B. durch nicht, und, oder or wenn ... dann ....

  Beispiel. A und nicht B

  (Die Sonne scheint und die Vorlesung ist nicht langweilig)

- Die Aussagenlogik studiert grundlegende Prinzipien korrekten Schließens mit Propositionen und deren Kombinationen.

Syntax der Aussagenlogik

Definition 2.1. (Aussagenvariablen)

Eine Aussagenvariable, oder auch einfach Variable, hat die Form V_i für $i \in \mathbb{N}$.

Die Menge aller Variablen bezeichnen wir als AVAR.

Definition 2.2. (Alphabet)

Das Alphabet der Aussagenlogik ist

$$\Sigma_{\mathsf{AL}} := \mathsf{AVAR} \cup \{\top, \bot, \neg, \vee, \wedge, \rightarrow, \leftrightarrow, (,)\}$$

Aussagenlogik

Definition 2.3. (Syntax der Aussagenlogik)

Die Klasse AL der aussagenlogischen Formeln ist induktiv definiert durch

Basis:

- $\top, \bot$ sind aussagenlogische Formeln
- Jede Variable $V_i \in \text{AVAR}$ ist eine aussagenlogische Formel

$\top, \bot$ und die Variablen werden atomare Formeln oder Atome genannt

Induktionsschritt:

- Wenn $\varphi \in \text{AL}$ eine Formel ist, dann auch $\neg\varphi \in \text{AL}$
- Wenn $\varphi, \psi \in \text{AL}$ Formeln sind, dann auch

$$(\varphi \vee \psi), \quad (\varphi \wedge \psi), \quad (\varphi \to \psi), \quad (\varphi \leftrightarrow \psi)$$

$\neg, \vee, \wedge, \to, \leftrightarrow$ werden aussagenlogische Verknüpfungen genannt.

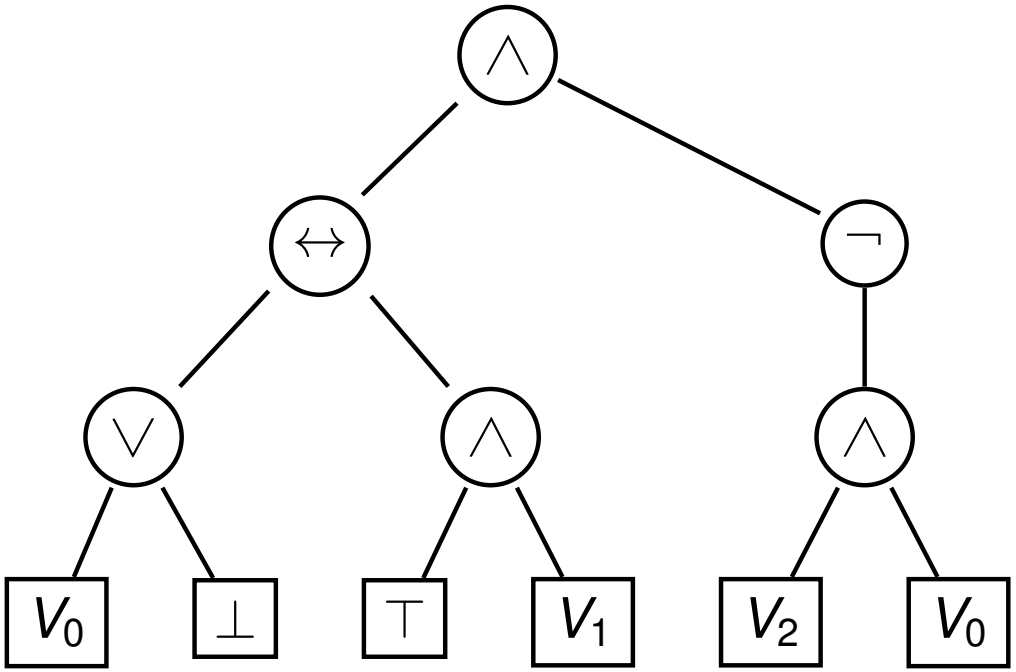
Syntax- oder Ableitungsbäume

Die Struktur einer Formel kann elegant durch ihren Syntax- oder Ableitungsbaum dargestellt werden.

Der Syntaxbaum der Formel

$$\varphi := (((V_0 \vee \bot) \leftrightarrow (\top \wedge V_1)) \wedge \neg(V_2 \wedge V_0))$$

ist definiert wie folgt:



Beispiel

Beispiel 2.4. Erinnern wir uns an das Beispiel vom Anfang der Vorlesung:

- Wenn der Zug zu spät ist und keine Taxis am Bahnhof stehen, kommt Peter zu spät zu seiner Verabredung.
- Peter kam nicht zu spät zu seiner Verabredung.
- Der Zug hatte Verspätung.

Also standen Taxis am Bahnhof.

Aussagenvariablen:

T: der Zug war zu spät
C: es standen Taxis am Bahnhof
L: Peter kam zu spät zu seiner Verabredung

Formel: $((((((T \wedge \neg C) \rightarrow L) \wedge \neg L) \wedge T) \rightarrow C)$

Semantik der Aussagenlogik

Aussagenvariablen:

T: der Zug war zu spät
C: es standen Taxis am Bahnhof
L: Peter kam zu spät zu seiner Verabredung

Formel: $((((((T \wedge \neg C) \rightarrow L) \wedge \neg L) \wedge T) \rightarrow C)$

- Ist die Aussage und somit die Formel wahr?
- Was bedeutet es, wenn eine Formel wahr oder falsch ist?

Um diese Fragen formal zu beantworten, müssen wir für aussagenlogische Formeln eine formale Semantik definieren.

Einschub: Unterformeln und induktive Definitionen

Unterformeln

Definition 2.5.

Die Menge $\mathsf{sub}(\varphi)$ der Unterformeln einer Formel φ ist induktiv wie folgt definiert:

- Ist φ atomar, dann ist $\mathsf{sub}(\varphi) := \{\varphi\}$
- Ist $\varphi := \neg\psi$, dann ist $\mathsf{sub}(\varphi) := \{\varphi\} \cup \mathsf{sub}(\psi)$
- Für alle $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$: Ist $\varphi := (\varphi_1 * \varphi_2)$, dann ist

$$\mathsf{sub}(\varphi) := \{\varphi\} \cup \mathsf{sub}(\varphi_1) \cup \mathsf{sub}(\varphi_2)$$

Bemerkung 2.6.

Formal ist sub eine Funktion

$$\mathsf{sub} : \mathsf{AL} \rightarrow \mathcal{P}(\mathsf{AL})$$

Beispiel

Formel: $\varphi := ((((((T \wedge \neg C) \rightarrow L) \wedge \neg L) \wedge T) \rightarrow C)$

Dann ist $\text{sub}(\varphi) := \{$ $((((((T \wedge \neg C) \rightarrow L) \wedge \neg L) \wedge T) \rightarrow C)$,
$(((T \wedge \neg C) \rightarrow L) \wedge \neg L) \wedge T)$,
C,
$((T \wedge \neg C) \rightarrow L) \wedge \neg L)$,
$((T \wedge \neg C) \rightarrow L)$,
$\neg L$,
L,
$(T \wedge \neg C)$,
T,
$\neg C\}$

Induktive Definition über Formeln

Die Definition der Menge der Unterformeln ist ein Beispiel für eine sehr elegante Methode, Funktionen über Formeln zu definieren oder Aussagen über Formeln zu beweisen.

Induktive Definitionen.

Eine Funktion $f : \mathrm{AL} \to M$, für eine beliebige Menge M, kann induktiv wie folgt definiert werden:

Basisfälle:

- Definiere $f(\top)$ und $f(\bot)$
- Definiere $f(X)$ für alle $X \in \mathrm{AVAR}$

Induktionsschritt:

- Definiere $f(\neg\varphi)$ aus $f(\varphi)$
- Für $* \in \{\wedge, \vee, \to, \leftrightarrow\}$ definiere $f((\varphi * \psi))$ aus $f(\varphi)$ und $f(\psi)$

Beispiel

Wir wollen eine Funktion $f : \mathrm{AL} \to \mathbb{N}$ definieren, so dass $f(\varphi)$ die "Größe" der Formel φ angibt.

Basisfälle:

- *Definiere* $f(\top)$ *und* $f(\bot)$
  $f(\top) = f(\bot) := 1$
- *Definiere* $f(X)$ *für alle* $X \in \mathrm{AVAR}$
  $f(X) := 1$ für alle $X \in \mathrm{AVAR}$

Induktionsschritt:

- *Definiere* $f(\neg\varphi)$ *aus* $f(\varphi)$
  $f(\neg\varphi) := 1 + f(\varphi)$
- *Für* $* \in \{\wedge, \vee, \to, \leftrightarrow\}$ *definiere* $f((\varphi * \psi))$ *aus* $f(\varphi)$ *und* $f(\psi)$
  $f\big((\varphi * \psi)\big) := 1 + f(\varphi) + f(\psi)$

Semantik der Aussagenlogik

Wahrheitsbelegungen

Definition 2.7. Die Menge $\mathrm{var}(\varphi)$ der Variablen einer Formel φ ist die Menge

$$\mathrm{var}(\varphi) := \mathrm{AVAR} \cap \mathrm{sub}(\varphi)$$

Definition 2.8.

1. Eine Wahrheitsbelegung, oder kurz Belegung, ist eine partielle Funktion
$$\beta : \mathrm{AVAR} \to \{0, 1\}.$$
2. Eine Belegung β ist eine Belegung für eine Formel φ, oder ist passend für φ, wenn $\mathrm{var}(\varphi) \subseteq \mathrm{Dom}(\beta)$.

Intuitiv: 1 steht für *wahr* und 0 für *falsch*.

Semantik der Aussagenlogik: Intuitive Bedeutung

Atome:

$\top, \bot$ stehen für *wahr* und *falsch*

Variablen V_i stehen für Aussagen. Wir interessieren uns allerdings nur dafür, ob sie wahr oder falsch sind.

Der Wahrheitswert wird durch die Belegung angegeben.

Negation: Ist $\neg\varphi$ wahr, so ist φ falsch.

Also steht $\neg$ für "nicht"

Konjunktion: Ist $(\varphi \wedge \psi)$ wahr so ist φ wahr und ψ wahr

$\wedge$ bedeutet "und"

Disjunktion: $\vee$ bedeutet "oder"

Implikation: $(\varphi \rightarrow \psi)$ bedeutet "φ impliziert ψ" oder wenn φ dann ψ

Biimplikation: $(\varphi \leftrightarrow \psi)$ bedeutet "φ dann, und nur dann, wenn ψ"

Semantik der Aussagenlogik

Definition 2.9. Per Induktion über die Struktur der Formeln in AL definieren wir eine Funktion $[\![\cdot]\!]^{\cdot}$, die jeder Formel $\varphi \in \mathrm{AL}$ und jeder zu φ passenden Belegung β einen Wahrheitswert $[\![\varphi]\!]^{\beta} \in \{0, 1\}$ zuordnet.

Basisfall.

- $[\![\bot]\!]^{\beta} := 0 \qquad [\![\top]\!]^{\beta} := 1$
- Für alle $X \in \mathrm{AVAR}$ gilt $[\![X]\!]^{\beta} := \beta(X)$

Induktionsschritt. Für zusammen gesetzte Formeln φ definieren wir $[\![\varphi]\!]^{\beta}$ durch die folgenden Wahrheitstafeln:

$[\![\varphi]\!]^{\beta}$	$[\![\neg\varphi]\!]^{\beta}$
0	1
1	0

Semantik der Aussagenlogik

Konjunktion

$[\![\varphi]\!]^\beta$	$[\![\psi]\!]^\beta$	$[\![(\varphi \wedge \psi)]\!]^\beta$
0	0	0
0	1	0
1	0	0
1	1	1

Disjunktion

$[\![\varphi]\!]^\beta$	$[\![\psi]\!]^\beta$	$[\![(\varphi \vee \psi)]\!]^\beta$
0	0	0
0	1	1
1	0	1
1	1	1

Implikation

$[\![\varphi]\!]^\beta$	$[\![\psi]\!]^\beta$	$[\![(\varphi \rightarrow \psi)]\!]^\beta$
0	0	1
0	1	1
1	0	0
1	1	1

Biimplikation

$[\![\varphi]\!]^\beta$	$[\![\psi]\!]^\beta$	$[\![(\varphi \leftrightarrow \psi)]\!]^\beta$
0	0	1
0	1	0
1	0	0
1	1	1

Eine Bemerkung zur Implikation

Bemerkung. Logische Implikation $\rightarrow$ stimmt nicht immer mit der umgangssprachlichen Verwendung der Implikation überein.

Zum Beispiel wird keine Kausalität impliziert.

$\varphi \rightarrow \psi$ heißt einfach, dass wann immer φ wahr ist, so muss auch ψ wahr sein.

Insbesondere, wenn φ falsch ist, dann ist $\varphi \rightarrow \psi$ als Aussage wahr.

Über die Wahl der Verknüpfungen

Unsere Wahl der Verknüpfungen $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ für die Aussagenlogik spiegelt unsere Verwendung in natürlicher Sprache wieder, ist aber zu bestimmtem Grad willkürlich.

Durch die Definition einer Wahrheitstafel können wir auch andere Verknüpfungen und somit auch andere "Aussagenlogiken" definieren.

Beispiel. Exklusives Oder $\varphi \oplus \psi$

Exklusives Oder

$[\![\varphi]\!]^\beta$	$[\![\psi]\!]^\beta$	$[\![\varphi \oplus \psi]\!]^\beta$
0	0	0
0	1	1
1	0	1
1	1	0

Intuitiv: $\varphi \oplus \psi$ bedeutet "entweder φ oder ψ"

Notation

Notation. Wir vereinbaren folgende Notation.

- Belegungen: $\beta, \gamma, \ldots$
- Formeln: $\varphi, \psi, \varphi' \ldots$
- Mengen von Formeln: $\Phi, \Psi, \ldots$
- Wir werden auch $X, Y, \ldots$ für Variablen verwenden

Präferenzregeln. Um unnötige Klammern zu vermeiden,

- lassen wir die äußersten Klammern weg
- vereinbaren, dass $\neg$ stärker bindet als die anderen Verknüpfungen
- $\wedge, \vee$ bindet stärker als $\rightarrow, \leftrightarrow$

Wir schreiben also $\neg X \wedge Y \rightarrow T$ für $((\neg X \wedge Y) \rightarrow T)$

Aber wir können nicht $X \wedge Y \vee Z$ schreiben.

Formalisierung natürlichsprachlicher Aussagen

Beispiel 2.10.

"Wenn es kalt ist und regnet, gehe ich nicht spazieren"

Atomare Aussagen.

- X_k: es ist kalt
- X_r: es regnet
- X_s: ich gehe spazieren

Obige Aussage kann also wie folgt formalisiert werden

$$X_k \wedge X_r \rightarrow \neg X_s$$

Formalisierung natürlichsprachlicher Aussagen

Beispiel 2.11.

"Das Fluchtauto war rot oder grün und hatte vorne und hinten keine Nummernschilder"

Atomare Aussagen.

- X_R: das Auto war rot
- X_G: das Auto war grün
- X_V: das Auto hatte Nummernschilder vorne
- X_H: das Auto hatte Nummernschilder hinten

Zusammengesetze Aussage.

$$(X_R \vee X_G) \wedge \neg X_V \wedge \neg X_H$$

Formalisierung natürlichsprachlicher Aussagen

Das folgende Beispiel ist nicht rein "aussagenlogisch". Eine Formalisierung in der Aussagenlogik kann dennoch nützlich sein.

Beispiel 2.12.

Annahme 1: **Alle** Menschen **sind** sterblich.

Annahme 2: Sokrates **ist ein** Mensch.

Folgerung: **Also ist** Sokrates sterblich.

Atomare Aussagen.

- X_M: x ist ein Mensch
- X_S: x ist sterblich
- X_K: x ist Sokrates

Zusammengesetzte Aussage. $\big((X_M \to X_S) \wedge (X_K \to X_M)\big) \to (X_K \to X_S)$

Dies vernachlässigt aber, dass die Aussage für alle Menschen gelten soll. $\rightsquigarrow$ Prädikatenlogik

Formalisierung natürlichsprachlicher Aussagen

Beispiel 2.13.

Annahme 1: **Alle** Menschen **sind** sterblich.

Annahme 2: Sokrates **ist ein** Mensch.

Folgerung: **Also ist** Sokrates sterblich.

Atomare Aussagen.

- X_M: x ist ein Mensch
- X_S: x ist sterblich
- X_K: x ist Sokrates

Zusammengesetzte Aussage. $\big((X_M \to X_S) \wedge (X_K \to X_M)\big) \to (X_K \to X_S)$

Wie hilft uns das zu entscheiden, ob unsere Folgerung korrekt war?

2.3. Erfüllbarkeit und Allgemeingültigkeit

Erfüllbarkeit und Gültigkeit

Definition 2.14. Sei $\varphi \in \mathrm{AL}$ eine Formel.

1. Eine zu φ passende Belegung β erfüllt φ, oder ist ein Modell von φ, wenn $[\![\varphi]\!]^\beta = 1$.

   Wir schreiben $\beta \models \varphi$.

2. φ ist erfüllbar, wenn es eine Belegung β gibt, die φ erfüllt. Anderenfalls ist φ unerfüllbar.

3. φ ist allgemeingültig, oder eine Tautologie, wenn jede zu φ passende Belegung φ erfüllt.

Beobachtung 2.15. Für alle Formeln $\varphi \in \mathrm{AL}$:

φ ist allgemeingültig genau dann, wenn $\neg\varphi$ unerfüllbar ist.

Erfüllbarkeit und Gültigkeit

Atomare Aussagen.

- X_M: x ist ein Mensch
- X_S: x ist sterblich
- X_K: x ist Sokrates

Zusammengesetzte Aussage. $\varphi := \big((X_M \to X_S) \wedge (X_K \to X_M)\big) \to (X_K \to X_S)$

Korrektheit von Aussagen.

Die Folgerung ist also genau dann korrekt, wenn φ allgemeingültig ist.

Wir brauchen also Methoden, um Allgemeingültigkeit und Erfüllbarkeit von Formeln zu entscheiden.

In dieser Vorlesung werden wir drei Methoden kennen lernen

- Wahrheitstafeln
- Resolution
- Sequenzenkalkül

Wahrheitstafeln

Wir können die Wahrheitswerte aussagenlogischer Formeln $\varphi \in AL$ unter allen möglichen Belegungen in einer Wahrheitstafel darstellen.

Für jede mögliche Belegung $\beta : var(\varphi) \to \{0, 1\}$ hat die Tafel eine Zeile die den Wahrheitswert $\beta(X)$ für alle $X \in var(\varphi)$ sowie $[\![\varphi]\!]^\beta$ enthält.

Beispiel 2.16. $\big((X_H \to X_M) \wedge (X_S \to X_H)\big) \to (X_S \to X_M)$

X_H	X_M	X_S	$\big((X_H \to X_M) \wedge (X_S \to X_H)\big) \to (X_S \to X_M)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Erweiterte Wahrheitstafeln

Es ist oft nützlich, die Wahrheitswerte der Unterformeln ebenfalls in der Wahrheitstafel aufzuführen.

Beispiel (forts.) $\varphi := \big((X_H \to X_M) \wedge (X_S \to X_H)\big) \to (X_S \to X_M)$

X_H	X_M	X_S	$(X_H \to X_M)$	$(X_S \to X_H)$	$(X_H \to X_M) \wedge (X_S \to X_H)$	$(X_S \to X_M)$	φ
0	0	0	1	1	1	1	1
0	0	1	1	0	0	0	1
0	1	0	1	1	1	1	1
0	1	1	1	0	0	1	1
1	0	0	0	1	0	1	1
1	0	1	0	1	0	0	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1

Wir nennen das die erweiterte Wahrheitstafel.

Wahrheitstafel

Bemerkung 2.17. Wenn die Formel φ keine Variable enthält, so besteht die Wahrheitstafel aus einer einzigen Zeile.

Sei $\varphi := (\top \to \bot) \to \bot$

$$\frac{(\top \to \bot) \to \bot}{\top}$$

Die erweiterte Wahrheitstafel ist

$\bot$	$\top$	$\top \to \bot$	$(\top \to \bot) \to \bot$
$\bot$	$\top$	$\bot$	$\top$

Das Koinzidenz Lemma

Wenn wir mit Wahrheitstafeln arbeiten, nehmen wir implizit an, dass $[\![\varphi]\!]^\beta$ nur von den Belegungen der Variablen in $\mathrm{var}(\varphi)$ abhängt, aber nicht von der Belegung anderer Variablen.

Eine Rechtfertigung dieser Annahme liefert das folgende Lemma.

Lemma 2.18. (Koinzidenzlemma)

Sei $\varphi \in \mathrm{AL}$ eine Formel und seien β, β' Belegungen so dass

$$\beta(X) = \beta'(X) \qquad \text{für alle } X \in \mathrm{var}(\varphi).$$

Dann gilt $[\![\varphi]\!]^\beta = [\![\varphi]\!]^{\beta'}$.

Intuitiv ist das Lemma klar, da die anderen Variablen $Y \notin \mathrm{var}(\varphi)$ nicht in der Definition von $[\![\varphi]\!]^\beta$ auftauchen.

Formal kann das Lemma durch strukturelle Induktion bewiesen werden (Übung).

Syllogismen

Syllogismenbeispiel.

- X_M: x ist ein Mensch
- X_S: x ist sterblich
- X_K: x ist Sokrates

Zusammengesetzte Aussage. $\varphi := \big((X_M \to X_S) \wedge (X_K \to X_M)\big) \to (X_K \to X_S)$

Wahrheitstafel.

X_H	X_M	X_S	$\big((X_H \to X_M) \wedge (X_S \to X_H)\big) \to (X_S \to X_M)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Folgerung. Diese Form der Syllogismen ist also korrekt.

Das Wahrheitstafelverfahren

Beobachtung 2.19. Sei $\varphi \in \mathrm{AL}$ eine Formel.

1. φ ist genau dann erfüllbar, wenn die letzte Spalte der Wahrheitstafel mindestens eine 1 enthält.
2. φ ist genau dann unerfüllbar, wenn alle Einträge der letzten Spalte 0 sind.
3. φ ist genau dann allgemeingültig, wenn alle Einträge der letzten Spalte 1 sind.

Das Wahrheitstafelverfahren.

Eingabe: Eine Formel $\varphi \in \mathrm{AL}$.

Ziel: entscheide, ob φ erfüllbar ist.

Methode:
1. Berechne die Wahrheitstafel für φ.
2. Überprüfe, ob die letzte Spalte eine 1 enthält.

Bemerkung. Für Allgemeingültigkeit entscheide, ob die letzte Spalte nur 1 enthält.

Effizienz des Wahrheitstafelverfahrens

Die Wahrheitstafel einer Formel mit n Variablen hat 2^n Zeilen.

Das macht das Wahrheitstafelverfahren extrem ineffizient außer für sehr kleine Formeln.

Variablen	Zeilen
10	$1{,}024 \approx 10^3$
20	$1{,}048{,}576 \approx 10^6$
30	$1{,}073{,}741{,}824 \approx 10^9$
40	$1{,}099{,}511{,}627{,}776 \approx 10^{12}$
50	$1{,}125{,}899{,}906{,}842{,}624 \approx 10^{15}$
60	$1{,}152{,}921{,}504{,}606{,}846{,}976 \approx 10^{18}$

Das Aussagenlogische Erfüllbarkeitsproblem

Bemerkung.

- Das Erfüllbarkeitsproblem der Aussagenlogik ist eines der am besten studierten Probleme der Informatik.
- Es ist “schwer” zu lösen (NP-vollständig, werden wir später beweisen)
- Allerdings existieren Verfahren, die das Problem für viele in der Praxis vorkommende Formeln sehr effizient lösen können. (Das Wahrheitstafelverfahren gehört nicht dazu)
- Diese haben wichtige Anwendungen in der Informatik, z.B. in der Verifikation.

2.4. Logische Äquivalenz und Normalformen

Normalformen

In diesem Abschnitt werden wir Methoden behandeln, Formeln umzuformen, ohne den Wahrheitswert der Formeln zu ändern.

Dies werden wir insbesondere für sogenannte Normalformen benutzen.

Normalformen. Allgemein ist eine Normalform eine Klasse aussagenlogischer Formeln, so dass jede Formel in AL zu einer Formel in dieser Normalform äquivalent ist.

Dazu werden wir zunächst den Äquivalenzbegriff von Formeln formal definieren und einige nützliche Äquivalenzen zwischen Formeln einführen.

Äquivalenz von Formeln

Definition 2.20. Zwei Formeln $\varphi, \psi \in \mathrm{AL}$ sind äquivalent, geschrieben $\varphi \equiv \psi$, wenn für alle Belegungen β gilt:

$$\beta \models \varphi \iff \beta \models \psi$$

Proposition 2.21.

1. Für alle Formeln φ, ψ

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

2. Für alle Formeln φ,

$$\varphi \text{ ist allgemeingültig} \iff \varphi \equiv \top.$$

Beispiel

Beispiel 2.22. Für alle $X, Y \in \mathrm{AVAR}$:

$$(X \to Y) \quad \equiv \quad \neg X \vee Y$$
$$(X \leftrightarrow Y) \quad \equiv \quad (X \to Y) \wedge (Y \to X)$$

Beweis mittels Wahrheitstafeln.

$[\![X]\!]^\beta$	$[\![Y]\!]^\beta$	$[\![X \to Y]\!]^\beta$	$[\![\neg X \vee Y]\!]^\beta$	$[\![(X \leftrightarrow Y)]\!]^\beta$	$[\![(X \to Y) \wedge (Y \to X)]\!]^\beta$
0	0	1	1	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	1	1	1	1	1

Frage. Können wir aus der Äquivalenz $(X \to Y) \equiv \neg X \vee Y$, für alle $X, Y \in \mathrm{AVAR}$, schließen, dass für alle $\varphi, \psi \in \mathrm{AL}$

$$(\varphi \to \psi) \quad \equiv \quad \neg\varphi \vee \psi?$$

Das Substitutionslemma (intuitiv)

Lemma 2.23. Wenn wir in einer Äquivalenz alle Vorkommen einer von Variablen $X_1, \ldots, X_n$ durch Formeln $\varphi_1, \ldots, \varphi_n$ ersetzen, dann bleibt die Äquivalenz erhalten.

Korollar 2.24. Für alle Formeln $\varphi, \psi \in \mathsf{AL}$ gilt:

$$(\varphi \to \psi) \equiv \neg\varphi \vee \psi$$
$$(\varphi \leftrightarrow \psi) \equiv (\varphi \to \psi) \wedge (\psi \to \varphi)$$

Definition 2.25. Eine Substitution ist eine partielle Abbildung

$$\mathcal{S} : \mathsf{AVAR} \to \mathsf{AL}$$

von Aussagenvariablen auf aussagenlogische Formeln mit endlichem Definitionsbereich, d.h. $\mathcal{S}$ ist nur für endlich viele Variablen definiert.

Substitution

Für eine Formel $\varphi \in \mathrm{AL}$ und eine Substitution $\mathcal{S}$ schreiben wir $\varphi\mathcal{S}$ für die Formel, die wir aus φ erhalten, wenn alle Vorkommen einer Variable $X \in \mathrm{Dom}(\mathcal{S})$ in φ durch die Formeln $\mathcal{S}(X)$ ersetzt werden.

Definition 2.26. Für jede Formel $\varphi \in \mathrm{AL}$ und Substitution $\mathcal{S}$ definieren wir die Formel $\varphi\mathcal{S} \in \mathrm{AL}$ induktiv wie folgt:

Induktionsbasis.

- $\bot\mathcal{S} := \bot \qquad \top\mathcal{S} := \top$
- Für $X \in \mathrm{AVAR}$ definieren wir $X\mathcal{S} := \begin{cases} \mathcal{S}(X) & \text{wenn } X \in \mathrm{Dom}(\mathcal{S}) \\ X & \text{sonst} \end{cases}$

Induktionsschritt.

- $(\neg\varphi)\mathcal{S} := \neg(\varphi\mathcal{S})$
- Für $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ definieren wir

$$(\varphi * \psi)\mathcal{S} := \varphi\mathcal{S} \ * \ \psi\mathcal{S}.$$

Beispiel

Sei $\varphi := V_1 \wedge V_2 \rightarrow V_1 \vee V_3$ und sei

$$\mathcal{S} : \{V_1, V_3\} \rightarrow \mathsf{AL}$$

definiert als $\mathcal{S}(V_1) := V_2$ und $\mathcal{S}(V_2) := (V_0 \vee V_1)$.

Dann gilt

$$\begin{aligned} \varphi\mathcal{S} &= (V_1 \wedge V_2)\mathcal{S} \rightarrow (V_1 \vee V_3)\mathcal{S} \\ &= V_1\mathcal{S} \wedge V_2\mathcal{S} \rightarrow V_1\mathcal{S} \vee V_3\mathcal{S} \\ &= V_2 \wedge (V_0 \vee V_1) \rightarrow V_2 \vee V_3 \end{aligned}$$

Beispiel

Sei $\varphi := ((V_1 \vee V_2) \wedge \neg(V_3 \wedge V_1))$ und sei $\mathcal{S} : \{V_1, V_2, V_3\} \to \mathsf{AL}$

definiert als $\mathcal{S} : \begin{cases} V_1 & \mapsto \neg(V_2 \wedge V_3) \\ V_2 & \mapsto (V_4 \vee V_5) \\ V_3 & \mapsto (V_2 \wedge V_3) \end{cases}$

Dann gilt

$$\begin{aligned} \varphi\mathcal{S} \quad = \quad & ((V_1 \vee V_2) \wedge \neg(V_3 \wedge V_1))\mathcal{S} \\ & ((V_1 \vee V_2)\mathcal{S} \wedge \neg(V_3 \wedge V_1)\mathcal{S}) \\ & ((V_1 \vee V_2)\mathcal{S} \wedge (\neg(V_3 \wedge V_1))\mathcal{S}) \\ & ((V_1\mathcal{S} \vee V_2\mathcal{S}) \wedge \neg((V_3 \wedge V_1)\mathcal{S})) \\ & ((V_1\mathcal{S} \vee V_2\mathcal{S}) \wedge \neg(V_3\mathcal{S} \wedge V_1\mathcal{S})) \\ & ((\neg(V_2 \wedge V_3) \vee (V_4 \vee V_5)) \wedge \neg((V_2 \wedge V_3) \wedge \neg(V_2 \wedge V_3))) \end{aligned}$$

Das Substitutionslemma (formal)

Lemma 2.27. Sei $\mathcal{S}$ eine Substitution und seien $\varphi, \varphi' \in \mathsf{AL}$ Formeln. Dann gilt

$$\varphi \equiv \varphi' \quad \Rightarrow \quad \varphi\mathcal{S} \equiv \varphi'\mathcal{S}.$$

Beweis. Der Beweis wird per Induktion über den Formelaufbau geführt.

Wir führen strukturelle Induktion zunächst allgemein ein und beweisen dann das Lemma.

Strukturelle Induktion

- Beweise über strukturelle Induktion basieren auf dem induktiven Aufbau aussagenlogischer Formeln.
- Das Prinzip ist eine elegante und nützliche Methode, Eigenschaften aussagenlogischer Formeln zu beweisen.

Um zu beweisen, dass eine Eigenschaft A für alle Formeln der Aussagenlogik gilt, müssen wir folgende Aussagen zeigen:

Induktionsbasis. Alle atomare Formeln φ haben die Eigenschaft A.

Induktionsschritt.

- Wenn φ die Eigenschaft A hat, so auch $\neg\varphi$.
- Wenn ψ und φ die Eigenschaft A haben, dann gilt A auch für $(\varphi * \psi)$, mit $* \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$.

Dann gilt A für alle Formeln $\varphi \in \mathrm{AL}$.

Beweis des Substitutionslemmas

Um das Substitutionslemma zu beweisen, brauchen wir zunächst noch etwas Notation.

Sei $\mathcal{S}$ eine Substitution.

- Eine Belegung β ist passend für $\mathcal{S}$, wenn sie passend für alle Formeln $\mathcal{S}(X)$ mit $X \in \mathrm{Dom}(\mathcal{S})$ ist.
- Ist β eine zu $\mathcal{S}$ passende Belegung, so definieren wir $\beta\mathcal{S}$ wie folgt

$$\beta\mathcal{S}(X) := \begin{cases} [\![\mathcal{S}(X)]\!]^\beta & \text{wenn } X \in \mathrm{Dom}(\mathcal{S}) \\ \beta(X) & \text{wenn } X \in \mathrm{Dom}(\beta) \setminus \mathrm{Dom}(\mathcal{S}) \end{cases}$$

Lemma 2.27-A. Für alle Formeln φ und alle zu φ und $\mathcal{S}$ passenden Belegungen β gilt:

$$\beta \models \varphi\mathcal{S} \quad \Longleftrightarrow \quad \beta\mathcal{S} \models \varphi$$

Beweis des Substitutionslemmas

Lemma 2.27-A. Für alle Formeln φ und alle zu φ und $\mathcal{S}$ passenden Belegungen β gilt:

$$\beta \models \varphi\mathcal{S} \quad \Longleftrightarrow \quad \beta\mathcal{S} \models \varphi$$

Wir beweisen das Lemma per Induktion über den Formelaufbau.

Induktionsbasis.

- Für $\varphi \in \{\top, \bot\}$ gilt $\varphi\mathcal{S} = \varphi$
- Für $\varphi := X$, wobei $X \in \text{AVAR} \setminus \text{Dom}(\mathcal{S})$, gilt:
  $\beta\mathcal{S}(X) = \beta(X)$ und $\varphi = \varphi\mathcal{S}$ und daher $[\![\varphi\mathcal{S}]\!]^\beta = [\![\varphi]\!]^{\beta\mathcal{S}}$.
- Für $\varphi := X \in \text{Dom}(\mathcal{S})$ gilt: $\varphi\mathcal{S} = \mathcal{S}(X)$ und $\beta\mathcal{S}(X) = [\![\mathcal{S}(X)]\!]^\beta$.
  Somit, $\beta \models \varphi\mathcal{S} \quad \Longleftrightarrow \quad \beta\mathcal{S} \models \varphi$.

Beweis des Substitutionslemmas

Inductionsschritt.

- Negation.

$$\begin{aligned}
\beta \models (\neg\varphi)\mathcal{S} &\iff \beta \models \neg(\varphi\mathcal{S}) && \text{Def. der Substitution}\\
&\iff \beta \not\models \varphi\mathcal{S}\\
&\iff \beta\mathcal{S} \not\models \varphi && \text{Induktionsvoraussetzung}\\
&\iff \beta\mathcal{S} \models \neg\varphi.
\end{aligned}$$

- Konjunktion.

$$\begin{aligned}
\beta \models (\varphi \wedge \psi)\mathcal{S} &\iff \beta \models (\varphi\mathcal{S} \wedge \psi\mathcal{S}) && \text{Def. der Subst.}\\
&\iff \beta \models \varphi\mathcal{S} \text{ und } \beta \models \psi\mathcal{S}\\
&\iff \beta\mathcal{S} \models \varphi \text{ und } \beta\mathcal{S} \models \psi && \text{Ind. Vor.}\\
&\iff \beta\mathcal{S} \models (\varphi \wedge \psi).
\end{aligned}$$

- Das Argument für $* \in \{\vee, \rightarrow, \leftrightarrow\}$ ist analog.

Beweis des Substitutionslemmas

Mit Lemma 2.27-A können wir nun das Substitutionslemma beweisen.

Seien φ, φ' äquivalente Formeln.

Wir zeigen, dass $\varphi\mathcal{S} \equiv \varphi'\mathcal{S}$, d.h. das für alle passenden Belegungen β:

$$\beta \models \varphi\mathcal{S} \iff \beta \models \varphi'\mathcal{S}.$$

Sei β eine zu $\varphi\mathcal{S}$ und $\varphi'\mathcal{S}$ passende Belegung. Dann ist $\beta\mathcal{S}$ passend für φ.

$$\begin{aligned}
\beta \models \varphi\mathcal{S} &\iff \beta\mathcal{S} \models \varphi && \text{nach vorherigem Lemma} \\
&\iff \beta\mathcal{S} \models \varphi' && \text{da } \varphi \equiv \varphi' \\
&\iff \beta \models \varphi'\mathcal{S} && \text{nach vorherigem Lemma}
\end{aligned}$$

Das schließt den Beweis des Substitutionslemmas ab. □

Das Substitutionslemma (formal)

Lemma 2.27. Sei $\mathcal{S}$ eine Substitution und seien $\varphi, \varphi' \in \mathsf{AL}$ Formeln. Dann gilt

$$\varphi \equiv \varphi' \quad \Rightarrow \quad \varphi\mathcal{S} \equiv \varphi'\mathcal{S}.$$

Beweis. Der Beweis wird per Induktion über den Formelaufbau geführt.

Wir führen strukturelle Induktion zunächst allgemein ein und beweisen dann das Lemma.

Notation

Notation 2.28.

1. Sei $\varphi \in \mathsf{AL}$ eine Formel.

   Wenn $X_1, \ldots, X_n$ Variablen in φ asind und $\psi_1, \ldots, \psi_n \in \mathsf{AL}$, schreiben wir

   $$\varphi[X_1/\psi_1, \ldots, X_n/\psi_n]$$

   für die Formel $\varphi\mathcal{S}$, wobei $\mathcal{S}$ die wie folgt definierte Substitution ist

   $$\mathsf{Dom}(\mathcal{S}) := \{X_1, \ldots, X_n\} \text{ und } \mathcal{S}(X_i) := \psi_i.$$

2. Wir schreiben $\varphi(X_1, \ldots, X_n) \in \mathsf{AL}$ um anzudeuten, dass $\varphi \in \mathsf{AL}$ und $\mathsf{var}(\varphi) := \{X_1, \ldots, X_n\}$.

Beispiel

Substitutionslemma. Sei $\mathcal{S}$ eine Substitution und seien $\varphi, \varphi' \in \mathsf{AL}$ Formeln. Dann gilt

$$\varphi \equiv \varphi' \quad \Rightarrow \quad \varphi\mathcal{S} \equiv \varphi'\mathcal{S}.$$

Äquivalenzen. Wir wissen bereits, dass

$$(X \to Y) \equiv \neg X \vee Y \qquad \text{und} \qquad (X \leftrightarrow Y) \equiv (X \to Y) \wedge (Y \to X)$$

Korollar. Für alle Formeln $\varphi, \psi \in \mathsf{AL}$:

$$\begin{aligned} (\varphi \to \psi) &\equiv \neg\varphi \vee \psi \\ (\varphi \leftrightarrow \psi) &\equiv (\varphi \to \psi) \wedge (\psi \to \varphi) \end{aligned}$$

Ersetzungen. Wir würden nun gerne weiter folgern, dass

$$(\varphi \leftrightarrow \psi) \equiv (\varphi \to \psi) \wedge (\psi \to \varphi) \equiv (\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi)$$

Die Substitution erlaubt diese Folgerung aber nicht.

Das Ersetzungslemma

Lemma 2.29. Sei $\varphi \in \mathrm{AL}$ eine Formel und ψ eine Unterformel von φ.

Sei φ' eine Formel, die man aus φ erhält, indem man ein Vorkommen der Unterformel ψ durch eine äquivalente Formel $\psi' \equiv \psi$ ersetzt.

Dann gilt $\varphi \equiv \varphi'$.

Bemerkung. Da Unterformeln mehrfach vorkommen können, ist φ' nicht eindeutig.

Mit Hilfe des Ersetzungslemmas können wir nun folgendes beweisen.

Korolar 2.30. Für alle Formeln φ, ψ:

$$(\varphi \leftrightarrow \psi) \quad \equiv \quad (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) \quad \equiv \quad (\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi)$$

Beweis des Ersetzungslemmas

Der Beweis des Ersetzungslemmas geht per struktureller Induktion.

Induktionsanfang. Ist φ atomar, so gilt offenbar $\psi = \varphi$ und somit $\varphi' = \psi'$. Nach Voraussetzung gilt also $\varphi \equiv \varphi'$.

Induktionsschritt. Angenommen, $\varphi := \neg\vartheta$.

- Ist $\psi = \varphi$, so ist nichts zu zeigen.
- Sonst ist ψ eine Teilformel von ϑ und $\varphi' = \vartheta'$, wobei ϑ' die Formel ist, die aus ϑ entsteht, indem ψ durch ψ' ersetzt wird. Nach IV gilt $\vartheta \equiv \vartheta'$.

  Nun gilt also für alle passenden Belegungen β:

$$[\![\varphi]\!]^\beta = 1 - [\![\vartheta]\!]^\beta = 1 - [\![\vartheta']\!]^\beta = [\![\varphi']\!]^\beta$$

  und somit $\varphi \equiv \varphi'$.

Schließlich bleibt noch der Fall $\varphi := (\varphi_1 \wedge \varphi_2)$. Die anderen Fälle sind analog.

Beweis des Ersetzungslemmas

Angenommen, $\varphi := (\varphi_1 \wedge \varphi_2)$.

Wiederum, falls $\psi = \varphi$, so ist nichts zu zeigen.

Sonst können wir o.B.d.A. annehmen, dass ψ eine Unterformel von φ_1 ist. Sei φ_1' die Formel, die aus φ_1 durch Ersetzen von ψ durch ψ' entsteht.

Nach IV gilt also $\varphi_1 \equiv \varphi_1'$ und somit, mit der gleichen Argumentation wie vorher, $\varphi \equiv \varphi'$. $\square$

Nützliche Äquivalenzen

Theorem 2.31. Für alle $\psi, \varphi, \vartheta \in \mathrm{AL}$:

1. $\neg\neg\varphi \equiv \varphi$ (Elimination doppelter Negation)
2. $\varphi \to \psi \equiv \neg\varphi \vee \psi$ (Elim. der Implikation)
3. $\varphi \leftrightarrow \psi \equiv (\varphi \to \psi) \wedge (\psi \to \varphi)$ (Elim. der Biimplikation)
4. $\neg(\psi \wedge \varphi) \equiv \neg\psi \vee \neg\varphi$
   $\neg(\psi \vee \varphi) \equiv \neg\psi \wedge \neg\varphi$ (de Morgansche Regeln)
5. $\psi \wedge (\varphi \vee \vartheta) \equiv (\psi \wedge \varphi) \vee (\psi \wedge \vartheta)$
   $\psi \vee (\varphi \wedge \vartheta) \equiv (\psi \vee \varphi) \wedge (\psi \vee \vartheta)$ (Distributivität)
6. $\psi \wedge (\psi \vee \varphi) \equiv \psi \vee (\psi \wedge \varphi) \equiv \psi$ (Absorbtionsgesetz)
7. $\psi \wedge \varphi \equiv \varphi \wedge \psi$
   $\psi \vee \varphi \equiv \varphi \vee \psi$ (Kommutativität von $\wedge$ und $\vee$)
8. $\psi \wedge (\varphi \wedge \vartheta) \equiv (\psi \wedge \varphi) \wedge \vartheta$
   $\psi \vee (\varphi \vee \vartheta) \equiv (\psi \vee \varphi) \vee \vartheta$ (Assoziativität von $\wedge$ und $\vee$)

Beweis des Theorems

Wir beweisen hier exemplarisch einige der Äquivalenzen. Die restlichen sind zur Übung empfohlen.

de Morgansche Regel $\neg(\psi \wedge \varphi) \equiv \neg\psi \vee \neg\varphi$. Sei β eine passende Belegung.

Dann gilt

$[\![\neg(\psi \wedge \varphi)]\!]^\beta = 1$	gdw.	wenn $[\![(\psi \wedge \varphi)]\!]^\beta = 0$.
	gdw.	mindestens eins von $[\![\psi]\!]^\beta, [\![\varphi]\!]^\beta$ gleich 0
	gdw.	mindestens eins von $[\![\neg\psi]\!]^\beta, [\![\neg\varphi]\!]^\beta$ gleich 1.
	gdw.	$[\![(\neg\psi \vee \neg\varphi)]\!]^\beta = 1$.

Beweis des Theorems

Distributivität $\psi \vee (\varphi \wedge \vartheta) \equiv (\psi \vee \varphi) \wedge (\psi \vee \vartheta)$.

Sei β eine passende Belegung.

- Ang. $[\![\psi \vee (\varphi \wedge \vartheta)]\!]^\beta = 0$. Also $[\![\psi]\!]^\beta = 0$ und $[\![(\varphi \wedge \vartheta)]\!]^\beta = 0$.

  Es folgt also, dass $[\![\varphi]\!]^\beta = 0$ oder $[\![\vartheta]\!]^\beta = 0$.

  O.B.d.A. sei $[\![\varphi]\!]^\beta = 0$. Dann gilt aber $[\![(\psi \vee \varphi)]\!]^\beta = 0$ und somit $[\![(\psi \vee \varphi) \wedge (\psi \vee \vartheta)]\!]^\beta = 0$.

- Sei also $[\![\psi \vee (\varphi \wedge \vartheta)]\!]^\beta = 1$.

  Es gilt also $[\![\psi]\!]^\beta = 1$ oder $[\![(\varphi \wedge \vartheta)]\!]^\beta = 1$.

  Falls $[\![\psi]\!]^\beta = 1$ so folgt $[\![(\psi \vee \varphi)]\!]^\beta = 1$ und $[\![\psi \vee \vartheta]\!]^\beta = 1$ und somit $[\![(\psi \vee \varphi) \wedge (\psi \vee \vartheta)]\!]^\beta = 1$.

  Anderenfalls gilt $[\![\varphi]\!]^\beta = [\![\vartheta]\!]^\beta = 1$. Dann aber $[\![(\psi \vee \varphi) = [\![\psi \vee \vartheta]\!]^\beta = 1]\!]^\beta$ und somit $[\![(\psi \vee \varphi) \wedge (\psi \vee \vartheta)]\!]^\beta = 1$.

□

Große Disjunktionen und Konjunktionen

Die folgende Notation ist oft nützlich:

$\bigwedge_{i=1}^{n} \varphi_i$ als Abkürzung für $(\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n)$

$\bigvee_{i=1}^{n} \psi_i$ als Abkürzung für $(\varphi_1 \vee \varphi_2 \vee \cdots \vee \varphi_n)$

Beispiel.

$\bigwedge_{i=1}^{999} X_i \to Y$: wenn alle 999 X_i wahr sind, so muss auch Y wahr sein

Bemerkung. Wegen der Assoziativitätsregeln

$$\begin{aligned} \psi \wedge (\varphi \wedge \vartheta) &\equiv (\psi \wedge \varphi) \wedge \vartheta \\ \psi \vee (\varphi \vee \vartheta) &\equiv (\psi \vee \varphi) \vee \vartheta \end{aligned}$$

ändert die Klammerung den Wahrheitswert nicht.

Reduzierte Formeln

Das vorherige Theorem liefert die folgenden Äquivalenzen.

$$\begin{aligned}(\varphi \to \psi) &\equiv \neg\varphi \vee \psi \\ (\varphi \leftrightarrow \psi) &\equiv (\varphi \to \psi) \wedge (\psi \to \varphi) \equiv (\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi) \\ (\varphi \wedge \psi) &\equiv \neg(\neg\varphi \vee \neg\psi)\end{aligned}$$

Mit Hilfe des Ersetzungslemmas erhalten wir daher folgende Aussage.

Korollar 2.32. Jede aussagenlogische Formel ist äquivalent zu einer Formel ohne $\wedge, \to, \leftrightarrow$, d.h. in der nur $\top$, $\bot$, Variablen und $\vee$ und $\neg$ vorkommen.

Wir nennen solche Formeln reduziert.

Reduzierte Formeln sind nützlich, um die Zahl der Fälle in strukturellen Induktionen zu begrenzen.

Normalformen

Negationsnormalform

Definition 2.33. Eine Formel $\varphi \in \mathrm{AL}$ ist in Negationsnormalform (NNF), wenn die Symbole $\rightarrow$ und $\leftrightarrow$ nicht vorkommen und Negation nur vor Variablen auftritt.

Beispiel. $\varphi := (\neg X \vee Y) \wedge \neg Z$

Theorem 2.34. Jede Formel $\varphi \in \mathrm{AL}$ ist äquivalent zu einer Formel $\varphi^* \in \mathrm{AL}$ in Negationsnormalform.

Beweis. Wir wissen bereits, dass φ zu einer Formel φ' ohne $\rightarrow, \leftrightarrow$ äquivalent ist.

O.B.d.A. nehmen wir daher an, dass φ bereits reduziert ist.

Durch Anwendung der Äquivalenzen 1. und 4. können wir die Formel in Negationsnormalform umwandeln.

Ein Algorithmus für die NNF

Eingabe. Eine reduzierte Formel $\varphi \in \mathrm{AL}$

Ausgabe. Eine Formel $\varphi^* \equiv \varphi$ in NNF

Algorithmus. Wiederhole die folgenden Schritte.

Wenn φ in NNF ist, gib $\varphi^* := \varphi$ aus.

Wenn φ eine Unterformel ψ der Form $\neg\neg\psi_1$ enthält,
dann ersetze ψ durch ψ_1 um φ' zu erhalten.

Wenn φ eine Unterformel ψ der Form $\neg(\psi_1 \wedge \psi_2)$ enthält,
dann ersetze ψ durch $(\neg\psi_1 \vee \neg\psi_2)$ um φ' zu erhalten.

Wenn φ eine Unterformel ψ der Form $\neg(\psi_1 \vee \psi_2)$ enthält
dann ersetze ψ durch $(\neg\psi_1 \wedge \neg\psi_2)$ um φ' zu erhalten.

Setze $\varphi := \varphi'$.

Lemma 2.35. Der Algorithmus terminiert auf jeder gültigen Eingabe $\varphi \in \mathrm{AL}$ und konstruiert eine Formel φ^* in NNF, so dass $\varphi \equiv \varphi^*$.

Beispiel

Sei $\varphi := \neg(\neg(X \vee Y) \wedge (\neg(X \wedge Y) \vee Z))$. Dann erzeugt der Algorithmus folgende Zwischenformeln:

$$
\begin{aligned}
& \neg(\neg(X \vee Y) \wedge (\neg(X \wedge Y) \vee Z)) \\
\equiv\ & (\neg\neg(X \vee Y) \vee \neg(\neg(X \wedge Y) \vee Z)) \\
\equiv\ & ((X \vee Y) \vee \neg(\neg(X \wedge Y) \vee Z)) \\
\equiv\ & ((X \vee Y) \vee \neg((\neg X \vee \neg Y) \vee Z)) \\
\equiv\ & ((X \vee Y) \vee (\neg(\neg X \vee \neg Y) \wedge \neg Z)) \\
\equiv\ & ((X \vee Y) \vee ((\neg\neg X \wedge \neg\neg Y) \wedge \neg Z)) \\
\equiv\ & ((X \vee Y) \vee ((X \wedge \neg\neg Y) \wedge \neg Z)) \\
\equiv\ & ((X \vee Y) \vee ((X \wedge Y) \wedge \neg Z))
\end{aligned}
$$

Beweis des Lemmas

Äquivalenz. Der Algorithmus ersetzt in jedem Schritt eine Unterformel ψ von φ durch eine äquivalente Formel. Nach dem Ersetzungslemma sind daher φ und φ' äquivalent.

Terminierung. Wir definieren eine Funktion

$$h : \mathsf{AL} \to \mathbb{N},$$

die die Tiefe einer Formel angibt, wie folgt:

- Ist ψ atomar, so gilt $h(\psi) := 0$.
- Ist $\psi := \neg\psi'$, so gilt $h(\psi) := 1 + f(\psi')$.
- Ist $\psi := (\psi_1 \vee \psi_2)$ oder $\psi := (\psi_1 \wedge \psi_2)$, so gilt $h(\psi) := 1 + \max\{f(\psi_1), f(\psi_2)\}$.

Die Tiefe einer Formel ist also die Höhe des Syntaxbaums der Formel.

Beobachtung. Eine Formel ist in NNF genau dann, wenn jede Unterformel der Form $\neg\psi'$ die Tiefe 1 hat.

Beweis des Lemmas

Sei jetzt $f(\varphi) := \sum\{3^{h(\psi)} : \psi = \neg\psi' \in \mathrm{sub}(\varphi)\}$

Behauptung. Sei φ eine Formel und φ' die Formel, die aus φ in einem Schritt des Algorithmus' entsteht. Dann gilt $f(\varphi') < f(\varphi)$.

Beweis. Angenommen, $\psi := \neg\neg\psi_1$. Wir ersetzen also ψ durch ψ_1. Dadurch erhöht sich die Tiefe der restlichen Negationsformeln nicht. Die Zahl solcher Formeln reduziert sich um 2 und somit gilt $f(\varphi') < f(\varphi)$.

Angenommen, $\psi := \neg(\psi_1 \vee \psi_2)$ oder $\psi := \neg(\psi_1 \wedge \psi_2)$. Dann bleibt die Tiefe aller Negationsformeln außer ψ gleich. In der Summierung wird also $3^{h(\psi)}$ durch $3^{h(\neg\psi_1)} + 3^{h(\neg\psi_2)} = 2 \cdot 3^{h(\psi)-1} < 3^{h(\psi)}$ ersetzt. Also gilt $f(\varphi') < f(\varphi)$. □

Hinweis. Ein Beispiel wird in den Tutorien besprochen.

Normalformen

Definition 2.36. Ein Literal L ist eine Aussagenvariable $X \in \text{AVAR}$ oder deren Negation $\neg X$.
Wir schreiben $\overline{L}$ für das Komplementliteral definiert als

$$\overline{L} := \begin{cases} \neg X & \text{if } L = X \\ X & \text{if } L = \neg X. \end{cases}$$

Definition 2.37. Eine Formel $\varphi \in \text{AL}$ ist in disjunktiver Normalform (DNF), wenn sie folgende Gestalt hat:

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

φ ist in konjunktiver Normalform (KNF), wenn sie folgende Gestalt hat:

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

Normalformen

Theorem 2.38.

1. Jede Formel $\varphi \in AL$ ist äquivalent zu einer Formel in disjunktiver Normalform.
2. Jede Formel $\varphi \in AL$ ist äquivalent zu einer Formel in konjunktiver Normalform.

Beweis. Das Theorem kann ähnlich wie der Satz über die Negationsnormalform bewiesen werden (Übung).

Wir verwenden hier einen anderen Ansatz über Boolesche Funktionen.

Einschub: Boolesche Funktionen

Boolesche Funktionen

Definition 2.39. Eine (n-stellige) Boolesche Funktion, nach Georg Boole benannt, ist eine Funktion

$$f : \{0,1\}^n \to \{0,1\}.$$

Wir definieren $\mathbb{B}^n$ als Menge aller n-stelligen Booleschen Funktionen.

Beispiel. Definiere

$$f(X_1,\ldots,X_n) := \begin{cases} 1 & \text{wenn die Mehrheit der } X_i \text{ gleich 1 sind} \\ 0 & \text{sonst} \end{cases}$$

Proposition 2.40. Jede Formel $\varphi(X_1,\ldots,X_n) \in \mathrm{AL}$ definiert eine Boolesche Funktion $f_\varphi := f(\varphi)$ mit

$$\begin{aligned} f_\varphi &: \{0,1\}^n \to \{0,1\} \\ f_\varphi(v_1,\ldots,v_n) &:= [\![\varphi]\!]^\beta \end{aligned}$$

wobei $\beta(X_i) := v_i$, für alle $1 \leqslant i \leqslant n$.

Boolesche Funktionen

Umgekehrt kann jede Boolesche Funktion durch eine Formel definiert werden.

Theorem 2.41. Zu jeder Booleschen Funktion $f : \{0,1\}^n \to \{0,1\}$ gibt es eine Formel $\varphi(X_1, \ldots, X_n)$, so dass $f(\varphi) = f$.

Beweis. Für jede Sequenz $\overline{v} := (v_1, \ldots, v_n) \in \{0,1\}^n$ definieren wir

$$\varphi_{\overline{v}} := \left(\bigwedge_{v_i=1} X_i\right) \wedge \left(\bigwedge_{v_i=0} \neg X_i\right)$$

Offensichtlich gilt für jede Belegung β, wenn $\beta \models \varphi_{\overline{v}}$ dann gilt für alle $1 \leqslant i \leqslant n$:

$$\beta(X_i) = 1 \iff v_i = 1.$$

Wir definieren nun die Funktion f durch die Formel

$$\varphi_f(X_1, \ldots, X_n) := \bigvee_{\substack{\overline{v} \in \{0,1\}^n \\ f(\overline{v})=1}} \varphi_{\overline{v}}.$$

Boolesche Funktionen

Wir müssen nun zeigen, dass für alle $\overline{v} := (v_1, \ldots, v_n)$:

$$f(\overline{v}) = 1 \iff [\![\varphi_f]\!]^\beta$$

wobei $\beta(X_i) := v_i$, für alle $1 \leqslant i \leqslant n$.

1. Wenn $f(\overline{v}) = 1$, dann $\beta \models \varphi_{\overline{v}}$ und $\varphi_{\overline{v}}$ ist Teil der Disjunktion in φ_f.

   Also $\beta \models \varphi_f$.

2. Umgekehrt, wenn $\beta \models \varphi_f$, dann muss es ein Disjunktionsglied $\varphi_{\overline{w}}$ geben, so dass $\beta \models \varphi_{\overline{w}}$.

   Nach Konstruktion von φ_f gilt $f(\overline{w}) = 1$.

   Aber $\beta \models \varphi_{\overline{w}}$ genau dann, wenn $w_i = \beta(X_i) = v_i$ für alle $1 \leqslant i \leqslant n$.

   Also $f(\overline{v}) = 1$.

Das schließt den Beweis ab. □

Beispiel

Beispiel. Definiere

$$f(X_1, X_2, X_3) := \begin{cases} 1 & \text{falls die Mehrheit der } X_i \text{ gleich 1 ist} \\ 0 & \text{sonst} \end{cases}$$

$f(v_1, v_2, v_3) = 1$ wenn mindestens zwei der v_i gleich 1 sind.

Wir erhalten also die folgende Formel

$$\begin{array}{rcll} \varphi_f(X_1, X_2, X_3) & := & (\neg X_1 \wedge X_2 \wedge X_3) & (=: \varphi_{0,1,1}) \\ & \vee & (X_1 \wedge \neg X_2 \wedge X_3) & (=: \varphi_{1,0,1}) \\ & \vee & (X_1 \wedge X_2 \wedge \neg X_3) & (=: \varphi_{1,1,0}) \\ & \vee & (X_1 \wedge X_2 \wedge X_3) & (=: \varphi_{1,1,1}) \end{array}$$

Formeln vs. Booleschen Funktionen

Wir haben folgenden Zusammenhang zwischen aussagenlogischen Formeln und Booleschen Funktionen gezeigt:

1. Jede Formel $\varphi(X_1, \ldots, X_n) \in \mathrm{AL}$ definiert eine Boolesche Funktion $f_\varphi := f(\varphi)$.
2. Zu jeder Booleschen Funktion $f : \{0,1\}^n \to \{0,1\}$ gibt es eine Formel $\varphi(X_1, \ldots, X_n)$, so dass $f(\varphi) = f$.

D. h. es besteht ein eins-zu-eins Zusammenhang zwischen Formeln und Booleschen Funktionen.

Korollar 2.42. Für alle $n \geqslant 0$ existieren genau 2^{2^n} paarweise nicht-äquivalente aussagenlogische Formeln in den Variablen $X_1, \ldots, X_n$.

Beweis. Es gibt 2^n verschiedene Belegungen der Variablen $X_1, \ldots, X_n$. Also existieren 2^{2^n} verschiedene n-stellige Boolesche Funktionen und somit ebensoviele aussagenlogische Formeln in den Variablen $X_1, \ldots, X_n$.

Zurück zu Normalformen

Normalformen

Theorem 2.43.

1. Jede Formel $\varphi \in \mathrm{AL}$ ist äquivalent zu einer Formel in disjunktiver Normalform.
2. Jede Formel $\varphi \in \mathrm{AL}$ ist äquivalent zu einer Formeln in konjunktiver Normalform.

Beweis. Teil 1 folgt sofort aus dem Beweis des vorherigen Theorems, da die dort konstruierten Formeln in disjunktiver Normalform sind.

Zu Teil 2: Sei $\varphi \in \mathrm{AL}$.

Nach Teil 1 ist $\neg\varphi$ äquivalent zu einer Formel $\psi := \bigvee_{i=1}^{n} \bigwedge_{j=1}^{n_i} L_{i,j}$ in DNF.

Mit Hilfe der de Morganschen Gesetze erhält man

$$\varphi \equiv \neg \bigvee_{i=1}^{n} \bigwedge_{j=1}^{n_i} L_{i,j} \quad \equiv \quad \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n_i} \overline{L_{i,j}}.$$ □

Normalformen

Bemerkung.

- Formeln in disjunktiver Normalform können sehr effizient auf Erfüllbarkeit getestet werden.
- Allerdings gibt es Formeln $\varphi_n \in \mathrm{AL}$, für alle $n \in \mathbb{N}$, so dass die Länge der kürzesten zu φ_n äquivalenten Formeln in DNF exponentiell in der Länge von φ_n sind.
- Es gibt also im Allgemeinen keinen effizienten Weg um aussagenlogische Formeln in disjunktive oder konjunktive Normalform umzuwandeln.
- Jedoch kann zu jeder Formel $\varphi \in \mathrm{AL}$ in Polynomialzeit eine Formel $\psi \in \mathrm{AL}$ in konjunktiver Normalform konstruiert werden, so dass

  φ genau dann erfüllbar ist, wenn ψ erfüllbar ist.

  Dies wird in praktischen Anwendungen benutzt, da die meisten aktuellen SAT-Löser Formeln in KNF als Eingabe erwarten.

Funktional vollständige Mengen von Verknüpfungen

Funktionale Vollständigkeit

Wie wir bereits gesehen haben, können die Verknüpfungen $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ auf natürliche Art als Boolesche Funktionen aufgefasst werden.

So entspricht z.B. $\wedge$ der Funktion $f_\wedge(X_1, X_2) \in \mathbb{B}^2$ mit $f_\wedge(X_1, X_2) = 1$ genau dann, wenn $X_1 = X_2 = 1$.

Umgekehrt kann man jede Boolesche Funktion $f \in \mathbb{B}^n$ als n-stellige Verknüpfung auffassen:

Aus Formeln $\varphi_1, \ldots, \varphi_n$ können wir die Formel $f(\varphi_1, \ldots, \varphi_n)$ bilden.

Beispiel. Sei $\oplus \in \mathbb{B}^2$ (XOR) die Boolesche Funktion definiert durch

$$\oplus(X_1, X_2) = 1 \text{ genau dann, wenn } X_1 \neq X_2.$$

Wir können dann Formeln $(\oplus(X, Y) \wedge \oplus(Y, Z))$ verwenden.

Funktionale Vollständigkeit

Wir können also in der Aussagenlogik auch andere Verknüpfungen zulassen und zum Beispiel die Logik auf den Verknüpfungen $\wedge, \neg, \oplus$ aufbauen.

Je nach verwendeten Verknüpfungen erhalten wir „äquivalente" Logiken oder aber Logiken, in denen nicht mehr alle Booleschen Funktionen definiert werden können.

Definition 2.44.

- Eine Menge $M \subseteq \mathbb{B}$ ist funktional vollständig, wenn sich daraus jede Boolesche Funktion im Sinne des vorherigen Satzes konstruieren läßt.
- Eine Menge V von Verknüpfungen ist funktional vollständig, wenn die entsprechende Menge Boolescher Funktionen funktional vollständig ist.

Hierbei bezeichnet $\mathbb{B} := \bigcup_n \mathbb{B}^n$ die Menge aller Booleschen Funktionen.

Funktionale Vollständigkeit

Beispiel. Wir wissen bereits, dass jede aussagenlogische Formel, und somit jede Boolesche Funktion, zu einer Formel in disjunktiver Normalform äquivalent ist.

Da in DNF nur die Verknüpfungen $\wedge, \vee, \neg$ vorkommen, ist also die Menge $\wedge, \vee, \neg$ funktional vollständig.

Funktionale Vollständigkeit

Beispiel Die Menge $\neg, \vee$ ist funktional vollständig.

Begründung.

Um zu zeigen, dass eine Menge V von Verknüpfungen funktional vollständig ist, reicht es zu zeigen, dass die Verknüpfungen einer funktional vollständigen Menge V' von Verknüpfungen aus V ausgedrückt werden können.

Wir wissen bereits, dass $(\varphi \wedge \psi) \equiv \neg(\neg\varphi \vee \neg\psi)$. Also ist $\neg, \vee$ funktional vollständig, da $\wedge, \vee, \neg$ funktional vollständig ist. □

Bemerkung.

- Ebenso zeigt man, dass $\neg, \wedge$ funktional vollständig ist.
- Die Menge $\wedge, \vee, \rightarrow$ ist nicht funktional vollständig.

  Zum Beispiel kann keine zu $\neg X$ äquivalente Formel gebildet werden (Übung).

Semantische Folgerung

Semantische Folgerung

Erinnern wir uns an das Beispiel vom Anfang der Vorlesung:

Voraussetzungen.

- Wenn der Zug zu spät ist und keine Taxis am Bahnhof stehen, kommt Peter zu spät zu seiner Verabredung.
- Peter kam nicht zu spät zu seiner Verabredung.
- Der Zug hatte Verspätung.

Folgerung. Also standen Taxis am Bahnhof.

Aussagenvariablen:

T: Zug zu spät C: Taxis am Bahnhof L: Peter kam zu spät

Die Argumentation kann wie folgt zusammengefasst werden:

Aus den Voraussetzungen: $\{(T \wedge \neg C) \rightarrow L, \quad \neg L, \quad T\}$

folgern wir Folgerung: C

Semantische Folgerung

Frage. Wie können wir solche logischen Schlüsse formalisieren?

- Gegeben eine Menge von Voraussetzungen:

  { “Zug zu spät und keine Taxis impliziert Peter zu spät",
    “Peter nicht zu spät, “Zug zu spät” }

  können wir daraus “es gab Taxis am Bahnhof” schließen?

- Gibt es allgemeine Methoden, solche Folgerungen aus den Voraussetzungen zu ziehen? Methoden, mit denen
  - nur logisch korrekte Folgerungen abgeleitet werden können, die aber
  - allgemein genug sind, damit alle logisch korrekten Folgerungen abgeleitet werden können?

Damit zusammenhängend stellen sich algorithmische Fragen.

Frage. Wie können wir solche Folgerungen berechnen?

- Die Wahrheitstafelmethode kann für endliche Voraussetzungsmengen benutzt werden.
- Allerdings ist die Methode sehr ineffizient, da die Wahrheitstafeln sehr groß werden können.

Semantische Folgerung

Wir führen zunächst den Folgerungsbegriff zwischen Formelmengen und Formeln ein, einen der wichtigsten Begriffe der Logik überhaupt, nicht nur für die Aussagenlogik.

Definition 2.45. Sei $\Phi \subseteq \mathrm{AL}$ eine Formelmenge und $\psi \in \mathrm{AL}$ eine Formel.

ψ folgt aus Φ, wenn jede zu $\Phi \cup \{\psi\}$ passende Belegung β, die Φ erfüllt, auch ψ erfüllt.

Wir schreiben $\Phi \models \psi$.

Falls $\Phi := \{\varphi\}$ nur eine Formel enthält, schreiben wir nur $\varphi \models \psi$.

Bemerkung. Wir verwenden das Symbol $\models$ sowohl für die Modellbeziehung $\beta \models \psi$ als auch für die semantische Folgerung $\Phi \models \psi$.

Eigenschaften der Folgerungsbeziehung

Das folgende Lemma listet einige einfache Eigenschaften der Folgerungsbeziehung auf, die sofort aus der Definition folgen.

Lemma 2.46. Sei $\Phi \subseteq \mathrm{AL}$ und $\psi, \psi' \in \mathrm{AL}$.

1. $\psi \equiv \psi'$ genau dann, wenn $\psi \models \psi'$ und $\psi' \models \psi$.
2. $\Phi \models \psi$ genau dann, wenn $\Phi \cup \{\neg\psi\}$ unerfüllbar ist.
3. Sei $\Phi_0 \subseteq \Phi$. Wenn $\Phi_0 \models \psi$, dann auch auch $\Phi \models \psi$.

Semantische Folgerung

Aussagenvariablen:

T: Zug zu spät C: Taxis am Bahnhof L: Peter kam zu spät

Die Argumentation kann wie folgt zusammengefasst werden:

Aus den Voraussetzungen: $\{(T \wedge \neg C) \rightarrow L, \quad \neg L, \quad T\}$

folgern wir Folgerung: C

Semantische Folgerung. Das bedeutet: Wir wollen zeigen, dass

$$\{(T \wedge \neg C) \rightarrow L, \quad \neg L, \quad T\} \quad \models \quad C$$

Semantische Folgerung

Wir werden in der Vorlesung zwei Methoden kennen lernen, mit denen semantische Folgerungen automatisch und elegant überprüft werden können.

- Aussagenlogische Resolution (jetzt)
- Der aussagenlogische Sequenzenkalkül (später in der VL)

Zunächst jedoch werden wir einen Satz beweisen, der uns in bestimmten Situationen auch die Behandlung unendlicher Formelmengen erlaubt bzw. vereinfacht.

2.6. Der Kompaktheitssatz der Aussagenlogik

Der Kompaktheitssatz

In bestimmten Anwendungen der Aussagenlogik treten unendliche Formelmengen auf, die auf Erfüllbarkeit untersucht werden sollen.

Wir werden als nächstes einen Satz beweisen, der uns diese Aufgabe wesentlich vereinfachen wird, da er es erlaubt, Erfüllbarkeit unendlicher Formelmengen auf Erfüllbarkeit endlicher Formelmengen zu reduzieren.

Der Kompaktheitssatz der Aussagenlogik

Definition 2.47.

Eine Menge Φ aussagenlogischer Formeln ist erfüllbar, wenn es eine Belegung β gibt, die zu allen $\varphi \in \Phi$ passt und alle $\varphi \in \Phi$ erfüllt.

Wir schreiben wiederum $\beta \models \Phi$.

Theorem (Kompaktheits- oder Endlichkeitssatz)

Sei $\Phi \subseteq AL$ eine Formelmenge und $\psi \in AL$ eine Formel.

1. Φ ist genau dann erfüllbar, wenn jede endliche Teilmenge $\Phi' \subseteq \Phi$ erfüllbar ist.
2. $\Phi \models \psi$ genau dann, wenn eine endliche Teilmenge $\Phi_0 \subseteq \Phi$ existiert, so dass $\Phi_0 \models \psi$.

Bemerkung. Der Satz kann auch für überabzählbare Variablen und somit Formelmengen bewiesen werden.

Beweis des Satzes

Wir werden zunächst den ersten Teil des Satzes beweisen, d.h.

> *Eine Menge $\Phi \subseteq \mathrm{AL}$ ist genau dann erfüllbar, wenn jede endliche Teilmenge $\Phi' \subseteq \Phi$ erfüllbar ist.*

Vorüberlegungen.

Offenbar ist das Lemma trivial, wenn Φ bereits endlich ist.

Sei Φ also eine unendliche Menge aussagenlogischer Formeln.

Ohne Beschränkung der Allgemeinheit nehmen wir an, dass es keine zwei verschiedenen Formeln $\psi, \psi' \in \Phi$ gibt, so dass $\psi \equiv \psi'$.

Denn, angenommen, es gäbe solche Formeln. Dann ist Φ genau dann erfüllbar, wenn $\Phi \setminus \{\psi'\}$ erfüllbar ist.

Es reicht also, den Beweis für Formelmengen zu zeigen, in denen alle Formeln paarweise nicht äquivalent sind.

Beweis des Satzes

Hinrichtung.

Zum Beweis der Hinrichtung sei Φ erfüllbar. Also existiert eine Belegung β, die jede Formel in Φ erfüllt. Also ist auch jede endliche Teilmenge von Φ erfüllbar, z. B. durch die Belegung β.

Rückrichtung.

Angenommen, alle endlichen Teilmengen $\Phi' \subseteq \Phi$ sind erfüllbar.

Seien $X_1, X_2, \ldots$ die in Formeln in Φ vorkommenden Aussagenvariablen.

Für $n \geqslant 0$ sei $\Phi_n \subseteq \Phi$ die Menge aller Formeln aus Φ in denen nur die Variablen $X_1, \ldots, X_n$ vorkommen (es müssen aber nicht alle vorkommen).

Es gilt also $\Phi_0 \subseteq \Phi_1 \subseteq \ldots \subseteq \Phi$.

Beweis des Satzes

Wir haben bereits bewiesen, dass es höchstens 2^{2^n} paarweise nicht-äquivalente Formeln in n Variablen gibt. Da Φ keine paarweise äquivalenten Formeln enthält, umfasst jedes Φ_n höchstens 2^{2^n} Formeln und ist damit endlich.

Nach Voraussetzung gibt es also für jedes $n \geqslant 0$ eine Belegung β_n, so dass $\beta_n \models \Phi_n$ und somit auch $\beta_n \models \Phi_i$ für alle $i \leqslant n$.

Sei $I_0 := \{\beta_n : n \geqslant 0\}$.

Wir konstruieren induktiv eine Belegung $\alpha : \{X_1, \ldots, X_n\} \to \{0, 1\}$ und Mengen $I_n \subseteq I_0$ wie folgt.

Dabei bewahren wir für alle n stets folgende Eigenschaft $(*)$:

- I_n ist unendlich und
- für alle $\beta, \beta' \in I_n$ und $j \leqslant n$ gilt $\beta(X_j) = \beta'(X_j) = \alpha(X_j)$.

Beweis des Satzes

Induktionsbasis $n = 1$. Da Φ unendlich ist, existiert ein $t \in \{0, 1\}$ so dass $\beta_n(X_1) = t$ für unendlich viele $\beta_n \in I_0$ gilt.

Setze $\alpha(X_1) := t$ und $I_1 := \{\beta \in I_0 : \beta(X_1) = t\}$.

Offenbar ist $(*)$ erfüllt.

Induktionsvoraussetzung. Seien $I_{n-1}, \alpha(X_i)$ für alle $i < n$ schon konstruiert so dass $(*)$ gilt.

Induktionsschritt. Da I_{n-1} wegen $(*)$ unendlich ist, gibt es ein $t \in \{0, 1\}$, so das $\beta(X_n) = t$ für unendlich viele $\beta \in I_{n-1}$.

Setze $\alpha(X_n) := t$ und $I_n := \{\beta \in I_{n-1} : \beta(X_n) = t\}$.

Offenbar ist $(*)$ erfüllt.

Beweis des Satzes

Behauptung. $\alpha \models \Phi$.

Sei $\varphi \in \Phi$.

Da φ nur endlich viele Variablen enthält, ist $\varphi \in \Phi_n$ für ein n.

Es gilt also $\beta_s \models \varphi$ für alle $i \geqslant n$, insbesondere also $\beta \models \varphi$ für alle $\beta \in I_n$.

Sei $\beta \in I_n$. So ein β existiert, da wegen $(*)$ $I_n \neq \emptyset$.

Da nach $(*)$ für alle $i \leqslant n$ gilt $\alpha(X_i) = \beta(X_i)$ und $\beta \models \varphi$, folgt also $\alpha \models \varphi$. □

Der Kompaktheitssatz der Aussagenlogik

Theorem (Kompaktheits- oder Endlichkeitssatz)

Sei $\Phi \subseteq \mathrm{AL}$ eine Formelmenge und $\psi \in \mathrm{AL}$ eine Formel.

1. Φ ist genau dann erfüllbar, wenn jede endliche Teilmenge $\Phi' \subseteq \Phi$ erfüllbar ist.
2. $\Phi \models \psi$ genau dann, wenn eine endliche Teilmenge $\Phi_0 \subseteq \Phi$ existiert, so dass $\Phi_0 \models \psi$.

Beweis von Teil 2.

Offenbar gilt $\Phi \models \psi$ genau dann, wenn $\Phi \cup \{\neg\psi\}$ unerfüllbar ist.

Dies ist aber nach Teil 1. genau dann der Fall, wenn bereits eine endliche Teilmenge Φ_0 unerfüllbar ist.

- Ist $\neg\psi \in \Phi_0$, so gilt also $\Phi_0 \setminus \{\neg\psi\} \models \psi$.
- Anderenfalls ist $\Phi_0 \subseteq \Phi$, und da Φ_0 unerfüllbar ist, folgt $\Phi_0 \models \psi$.

Die Umkehrung ist trivial. □

Eine Anwendung

Definition. Ein Graph $G := (V, E)$ besteht aus einer Knotenmenge V und einer Kantenmenge $E \subseteq \{\{u, v\} : u \neq v, u, v \in V\}$.

G ist 3-färbbar, wenn es eine Funktion $c : V \to \{C_1, C_2, C_3\}$ gibt, so dass $c(u) \neq c(v)$ für alle Kanten $\{u, v\} \in E$.

Lemma. Ein Graph ist genau dann 3-färbbar, wenn bereits jeder endliche Untergraph 3-färbbar ist.

Beweis. Übung.

2.7. Aussagenlogische Resolution

Aussagenlogische Resolution

Wir werden in diesem Abschnitt eine Methode kennen lernen, um die Unerfüllbarkeit aussagenlogischer Formeln nachzuweisen.

Da für eine Formelmenge $\Phi \subseteq AL$ und eine Formel ψ gilt:

$$\Phi \models \psi \text{ genau dann, wenn } \Phi \cup \neg\psi \text{ unerfüllbar ist}$$

können mit Hilfe der Resolution auch semantische Folgerungen überprüft werden.

Aussagenlogische Resolution: Beispiel

Wir wollen zeigen, dass die folgende Formel φ unerfüllbar ist.

$$\neg V \wedge (Y \vee Z \vee V) \wedge (\neg X \vee \neg Z) \wedge (X \vee \neg Z) \wedge (\neg Y \vee W) \wedge (\neg W \vee Z)$$

Angenommen, φ wäre erfüllbar. Sei β eine Belegung, so dass $\beta \models \varphi$.

- Offensichtlich gilt, $\beta \models \neg V$
- Aus $\beta \models \neg V$ und $\beta \models Y \vee Z \vee V$ folgt $\beta \models Y \vee Z$
- Aus $\beta \models Y \vee Z$ und $\beta \models \neg Y \vee W$ folgt $\beta \models Z \vee W$
- Aus $\beta \models Z \vee W$ und $\beta \models \neg W \vee Z$ folgt $\beta \models Z$
- Aus $\beta \models \neg X \vee \neg Z$ und $\beta \models X \vee \neg Z$ folgt aber auch $\beta \models \neg Z$
- Offensichtlich ist das ein Widerspruch zu $\beta \models Z$.

Aussagenlogische Resolution

Die aussagenlogische Resolution ist eine Methode um zu zeigen, dass eine Formel in konjunktiver Normalform nicht erfüllbar ist.

Wir haben bereits gezeigt, dass jede Formel $\varphi \in \mathrm{AL}$ zu einer Formel ψ in KNF äquivalent ist.

Also kann die Resolutionsmethode für alle Formeln verwendet werden, indem sie zunächst in KNF umgewandelt werden.

Für praktische Anwendungen der Resolutionsmethode ist folgender Satz nützlich.

Theorem 2.48. Zu jeder Formel φ gibt es eine Formel ψ in KNF, so dass

1. φ ist genau dann erfüllbar, wenn ψ erfüllbar ist.
2. $|\psi| \leqslant c \cdot |\varphi|$ für eine Konstante $c \in \mathbb{N}$ unabhängig von φ.
3. ψ kann aus φ effizient (in Linearzeit) berechnet werden.

Notation

Um das Schreiben von Resolutionsableitungen zu vereinfachen, verwenden wir folgende Notation.

Eine Formel

$$(\neg X \vee \neg Z) \wedge (X \vee \neg Z) \wedge (\neg Y \vee W) \wedge (Y \vee Z \vee V) \wedge \neg V \wedge (\neg W \vee Z)$$

in KNF schreiben wir als Klauselmenge wie folgt:

$$\{\neg X, \neg Z\}, \quad \{X, \neg Z\}, \quad \{\neg Y, W\}, \quad \{Y, Z, V\}, \quad \{\neg V\}, \quad \{\neg W, Z\}$$

D.h., aus jeder Disjunktion $Y \vee Z \vee V$ wird eine als Klausel bezeichnete Menge.

Klauseln

Definition 2.49.

- Eine Klausel ist eine endliche Menge von Literalen.
- Die leere Klausel wird mit $\square$ bezeichnet.
- Mit jeder Formel $\varphi := \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} L_{i,j}$ in KNF assoziieren wir eine endliche Menge $\mathcal{C}(\varphi)$ von Klauseln wie folgt:
  - mit jeder Disjunktion $\bigvee_{j=1}^{m_i} L_{i,j}$ assoziieren wir die Klausel $C_i := \{L_{i,j} : 1 \leqslant j \leqslant m_i\}$
  - wir definieren $\mathcal{C}(\varphi) := \{C_1, \ldots, C_n\}$.
- Umgekehrt, entspricht jeder Menge $\mathcal{C} := \{C_1, \ldots, C_n\}$ von Klauseln

  $$C_i := \{L_{i,j} : 1 \leqslant j \leqslant m_i\}$$

  die Formel $\varphi(\mathcal{C}) := \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} L_{i,j}$.

  Falls $\mathcal{C} := \emptyset$, definieren wir $\varphi(\mathcal{C}) := \top$.

- Der leeren Klausel $\square$ wird die Formel $\bot$ zugeordnet.

Formeln vs. Klauselmengen

Formeln in KNF und Klauselmengen entsprechen sich eins zu eins.

Wir erweitern daher Notation für Formeln auf naheliegende Weise auf Klauselmengen.

- Für eine Belegung β und Klauselmenge $\mathcal{C}$ schreiben wir $\beta \models \mathcal{C}$ für $\beta \models \varphi(\mathcal{C})$
- Wir schreiben $\mathcal{C} \models C$ um zu sagen, dass jede $\mathcal{C}$ erfüllende Belegung auch C erfüllt.
- Eine Klauselmenge $\mathcal{C}$ ist erfüllbar, wenn $\varphi(\mathcal{C})$ erfüllbar ist.
- Wenn $\mathcal{C}$ nur eine Klausel C enthält, schreiben wir einfach nur $\beta \models C$ etc.

Offenbar erfüllt eine Belegung β eine Klauselmenge $\mathcal{C}$, wenn jede Klausel $C \in \mathcal{C}$ ein Literal L enthält, so dass $[\![L]\!]^\beta = 1$.

Insbesondere ist also jede Klauselmenge, die die leere Klausel enthält, unerfüllbar.

Resolution

Notation. Für Literal L ist $\bar{L}$ das duale Literal, d.h. $\bar{X} = \neg X$ und $\overline{\neg X} = X$.

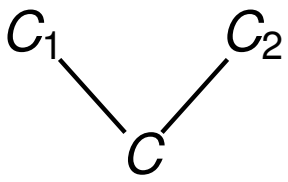
Definition 2.50. Seien C, C_1, C_2 Klauseln.

C ist eine Resolvente von C_1, C_2, wenn es ein Literal L gibt mit

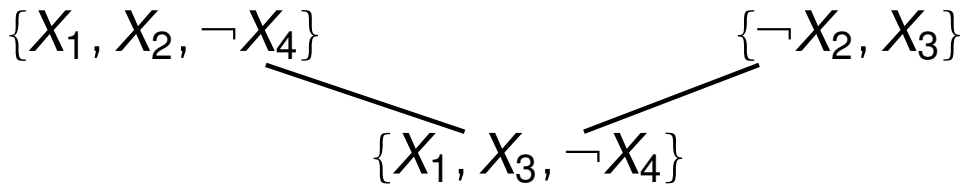
$$L \in C_1 \text{ und } \bar{L} \in C_2 \text{ und } C = \left(C_1 \setminus \{L\}\right) \cup \left(C_2 \setminus \{\bar{L}\}\right).$$

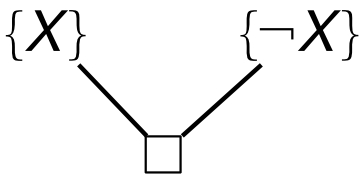
Wir sagen, dass C_1 und C_2 resolviert werden und schreiben $Res(C_1, C_2)$ für die Menge der Resolventen von C_1 und C_2.

Wir werden das oft wie folgt graphisch darstellen



Beispiel.





Resolution

Anmerkung. Zwei Klauseln können mehr als eine Resolvente haben, d.h. $Res(C_1, C_2)$ kann mehr als ein Element enthalten.

Die Klauseln $\{X, Y, Z\}$ und $\{\neg X, \neg Y, W\}$ haben als Resolventen $\{Y, \neg Y, Z, W\}$ und $\{X, \neg X, Z, W\}$.

Dies ist aber ein degenerierter Fall, der im weiteren keine Rolle spielen wird. Insbesondere ist die Resolvente immer allgemeingültig, da sie ein Literal und sein duales Literal enthält.

Aussagenlogische Resolution

Lemma 2.51.

Sei $\mathcal{C}$ eine Klauselmenge. Seien $C_1, C_2 \in \mathcal{C}$ und $C \in Res(C_1, C_2)$.

Dann gilt $\{C_1, C_2\} \models C$ und $\mathcal{C}$ und $\mathcal{C} \cup \{C\}$ sind äquivalent.

Beweis (Teil 1). Wir zeigen zunächst, dass $\{C_1, C_2\} \models C$.

Wir müssen also zeigen, dass jede Belegung β mit $\beta \models \{C_1, C_2\}$ auch C erfüllt.

Sei also β so, dass $\beta \models \{C_1, C_2\}$.

Sei L das resolvierte Literal, d.h. $C := \left(C_1 \setminus \{L\}\right) \cup \left(C_2 \setminus \{\bar{L}\}\right)$.

1. Angenommen, $[\![L]\!]^\beta = 1$. Da $\beta \models \{C_2\}$ folgt, dass es ein $L' \in C_2 \setminus \{\bar{L}\}$ gibt mit $[\![L']\!]^\beta = 1$. Also $\beta \models \{C\}$, da nach Definition der Resolvente $L' \in C$.
2. Sei nun $[\![L]\!]^\beta = 0$. Da $\beta \models \{C_1\}$, gibt es ein Literal $L' \in C_1 \setminus \{L\}$ mit $[\![L']\!]^\beta = 1$. Also $\beta \models \{C\}$, da nach Definition der Resolvente $L' \in C$.

Insgesamt gilt also $\beta \models C$.

Aussagenlogische Resolution

Lemma 2.52.

Sei $\mathcal{C}$ eine Klauselmenge. Seien $C_1, C_2 \in \mathcal{C}$ und $C \in Res(C_1, C_2)$.

Dann gilt $\{C_1, C_2\} \models C$ und $\mathcal{C}$ und $\mathcal{C} \cup \{C\}$ sind äquivalent.

Beweis (Teil 2).

Wir zeigen als nächstes, dass $\mathcal{C}$ und $\mathcal{C} \cup \{C\}$ äquivalent sind.

Dazu müssen wir zeigen, dass für alle Belegungen β gilt:

$$\beta \models \mathcal{C} \text{ gdw. } \beta \models \mathcal{C} \cup \{C\}.$$

- Wir haben bereits gesehen, dass $\{C_1, C_2\} \models C$. Da $C_1, C_2 \in \mathcal{C}$, folgt $\mathcal{C} \models \mathcal{C} \cup \{C\}$.
- Umgekehrt, wenn $\beta \models \mathcal{C} \cup \{C\}$ dann $\beta \models \mathcal{C}$.

Also sind $\mathcal{C}$ und $\mathcal{C} \cup \{C\}$ äquivalent. □

Resolutionsableitungen

Definition 2.53.

1. Eine Resolutionsableitung einer Klausel C aus eine Klauselmenge $\mathcal{C}$ ist eine Sequenz $(C_1, \ldots, C_n)$, so dass $C_n = C$ und für alle $1 \leqslant i \leqslant n$
   - $C_i \in \mathcal{C}$ oder
   - es gibt $j, k < i$ mit $C_i \in \mathit{Res}(C_j, C_k)$.

   Wir sagen, dass C einen Resolutionsbeweis aus $\mathcal{C}$ hat und schreiben dies als $\mathcal{C} \vdash_R C$.
2. Eine Resolutionswiderlegung einer Klauselmenge $\mathcal{C}$ ist eine Resolutionsableitung der leeren Klausel $\square$.

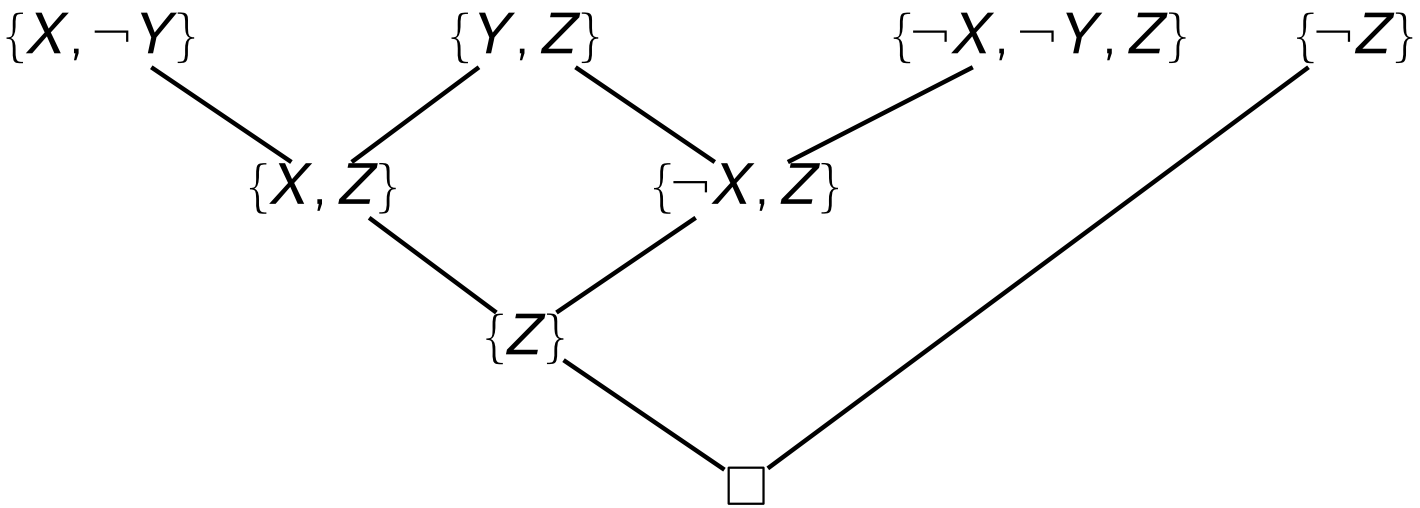
Beispiel. Sei $\mathcal{C} := \big\{\{X, \neg Y\}, \quad \{Y, Z\}, \quad \{\neg X, \neg Y, Z\}, \quad \{\neg Z\}\big\}$.
Dann ist

$$\Big(\begin{array}{ccccc} \{X, \neg Y\}, & \{Y, Z\}, & \{X, Z\}, & \{\neg X, \neg Y, Z\}, & \{\neg X, Z\}, \\ & & \{Z\}, & \{\neg Z\}, & \square \end{array} \Big)$$

eine Resolutionswiderlegung von $\mathcal{C}$.

Beispiel

Es ist oft hilfreich, Resolutionsableitungen graphisch darzustellen.



Der aussagenlogische Resolutionskalkül

Wir werden als nächstes zeigen, dass die Resolutionsmethode eine korrekte und auch vollständige Methode ist, um Unerfüllbarkeit von Klauselmengen, und damit auch aussagenlogischen Formeln, nachzuweisen.

Das heißt, wir werden folgenden Satz beweisen.

Theorem 2.54. Eine Menge $\mathcal{C}$ von Klauseln hat genau dann eine Resolutionswiderlegung, wenn $\mathcal{C}$ unerfüllbar ist.

Korrektheit der Resolution

Dazu zeigen wir zunächst das folgende Lemma.

Lemma 2.55. Sei $\mathcal{C}$ eine Klauselmenge und C eine Klausel.

Wenn $\mathcal{C} \vdash_R C$ dann $\mathcal{C} \models C$.

Beweis Sei $(C_1, \ldots, C_n)$ eine Resolutionsableitung von C aus $\mathcal{C}$. Per Induktion über i zeigen wir, dass $\mathcal{C} \models C_i$ für alle $1 \leqslant i \leqslant n$.

Für $i = n$ folgt somit $\mathcal{C} \models C_n = C$.

Induktionsbasis: $i = 1$. Es gilt $C_1 \in \mathcal{C}$ und somit $\mathcal{C} \models C_1$.

Induktionsschritt. Angenommen, die Behauptung gilt für $1, \ldots, i$.

Falls $C_{i+1} \in \mathcal{C}$, so gilt $\mathcal{C} \models C_{i+1}$.

Anderenfalls gibt es $j, k < i + 1$ mit $C_{i+1} \in \mathit{Res}(C_j, C_k)$.

Aus der Induktionsannahme folgt $\mathcal{C} \models C_j$ und $\mathcal{C} \models C_k$.

Nach dem vorherigen Lemma gilt $C_j, C_k \models C_{i+1}$ und somit $\mathcal{C} \models C$. $\square$

Vollständigkeit und Korrektheit der Resolution

Korollar 2.56. (Korrektheit des Resolutionskalküls)

Wenn eine Menge $\mathcal{C}$ von Klauseln eine Resolutionswiderlegung hat, ist $\mathcal{C}$ unerfüllbar.

Wir zeigen als nächstes die Umkehrung des Korollars.

Lemma 2.57. (Vollständigkeit des Resolutionskalküls)

Jede unerfüllbare Klauselmenge $\mathcal{C}$ hat eine Resolutionswiderlegung.

Dazu beweisen wir zunächst die folgende Behauptung..

Behauptung. *Sei $n \in \mathbb{N}$ und sei $\mathcal{C}$ eine unerfüllbare Klauselmenge in den Variablen $\{V_1, \ldots, V_{n-1}\}$. Dann hat $\mathcal{C}$ eine Resolutionswiderlegung.*

Beweis der Behauptung

Der Beweis der Behauptung wird per Induktion über n geführt.

Induktionsbasis $n = 1$**.** In diesem Fall ist $\mathcal{C}$ unerfüllbar und enthält keine Variablen. Also $\mathcal{C} := \{\square\}$ und somit existiert eine Resolutionswiderlegung.

Induktionsschritt $n \to n+1$**.**

Sei $\mathcal{C}$ eine unerfüllbare Klauselmenge in den Variablen $\{V_1, \ldots, V_n\}$.

Wir definieren
$$\mathcal{C}^+ := \{C \setminus \{\neg V_n\} : C \in \mathcal{C} \text{ und } V_n \notin C\} \text{ und}$$
$$\mathcal{C}^- := \{C \setminus \{V_n\} : C \in \mathcal{C} \text{ und } \neg V_n \notin C\}.$$

D.h., man erhält $\mathcal{C}^+$ indem alle Klauseln, die V_n enthalten entfernt werden und $\neg V_n$ aus den anderen entfernt wird. $\mathcal{C}^-$ ist analog definiert.

$\mathcal{C}^+$ und $\mathcal{C}^-$ sind beide unerfüllbar. Denn wäre z.B. $\mathcal{C}^+$ erfüllbar, z.B. durch $\beta \models \mathcal{C}^+$, dann würde $\beta' := \beta \cup \{V_n \mapsto 1\}$ die Menge $\mathcal{C}$ erfüllen.

Beweis der Behauptung (Forts.)

Nach Induktionsvoraussetzung gibt es Resolutionsableitungen $(C_1, \ldots, C_s)$ und $(D_1, \ldots, D_t)$ der leeren Klausel $C_s = D_t = \square$ aus $\mathcal{C}^+$ bzw. $\mathcal{C}^-$.

Falls $(C_1, \ldots, C_s)$ schon eine Ableitung von $\square$ aus $\mathcal{C}$ ist, sind wir fertig. Anderenfalls werden Klauseln C_i benutzt, die aus $\mathcal{C}$ durch Entfernen von $\neg V_n$ entstanden sind, d.h. $C_i \cup \{\neg V_n\} \in \mathcal{C}$.

Fügen wir zu diesen Klauseln und allen Resolventen wieder $\neg V_n$ hinzu, so erhalten wir eine Ableitung $(C'_1, \ldots, C'_s)$ von $\neg V_n$ aus $\mathcal{C}$.

Analog ist entweder $(D_1, \ldots, D_t)$ bereits eine Ableitung von $\square$ aus $\mathcal{C}$ oder wir erhalten eine Ableitung $(D'_1, \ldots, D'_t)$ von $\{V_n\}$ aus $\mathcal{C}$.

Beweis der Behauptung (Forts.)

Ein weiterer Resolutionsschritt auf $\{V_n\}$ und $\{V_n\}$ ergibt dann $\square$.

In diesem Fall ist also $(C'_1, \ldots, C'_s, D'_1, \ldots, D'_t, \square)$ eine Resolutionswiderlegung von $\mathcal{C}$.

Dies schließt den Beweis der Behauptung ab. $\dashv$

Vollständigkeit des Resolutionskalküls

Behauptung. *Sei $n \in \mathbb{N}$ und sei $\mathcal{C}$ eine unerfüllbare Klauselmenge in den Variablen $\{V_1, \ldots, V_{n-1}\}$. Dann hat $\mathcal{C}$ eine Resolutionswiderlegung.*

Lemma 2.58. (Vollständigkeit des Resolutionskalküls)
Jede unerfüllbare Klauselmenge $\mathcal{C}$ hat eine Resolutionswiderlegung.

Beweis. Sei $\mathcal{C}$ eine unerfüllbare Klauselmenge.

1. Ist $\mathcal{C}$ endlich, dann enthält sie nur endlich viele Variablen und der Beweis folgt sofort aus der Behauptung.

2. Ist $\mathcal{C}$ unendlich, dann folgt aus dem Kompaktheitssatz, dass bereits eine endliche Teilmenge $\mathcal{C}' \subseteq \mathcal{C}$ unerfüllbar ist.
   Also hat $\mathcal{C}'$ eine Resolutionswiderlegung.
   Diese ist aber auch eine Resolutionswiderlegung von $\mathcal{C}$. □

Vollständigkeit und Korrektheit des Resolutionskalküls

Zusammengenommen ergeben die beiden vorherigen Aussagen also folgenden Satz.

Theorem 2.59. Eine Menge $\mathcal{C}$ von Klauseln hat genau dann eine Resolutionswiderlegung, wenn $\mathcal{C}$ unerfüllbar ist.

Der Resolutionskalkül ist also eine vollständige Methode, um Unerfüllbarkeit (und damit auch Erfüllbarkeit) nachzuweisen.

Trotz seiner abstrakten Formulierung hat das Erfüllbarkeitsproblem der Aussagenlogik wichtige algorithmische Anwendungen in der Informatik.

Ein effizientes Verfahren zur Lösung des Problems hätte daher weitreichende Auswirkungen.

Wir werden aber als nächstes zeigen, dass ein solches Verfahren vermutlich nicht existieren kann, da das Problem NP-vollständig ist.

2.8. Das aussagenlogische Erfüllbarkeitsproblem

NP-Vollständigkeit von SAT

Definition 2.60. Das aussagenlogische Erfüllbarkeitsproblem (SAT) ist das Problem

SAT

Eingabe. Eine aussagenlogische Formel $\varphi \in \mathrm{AL}$

Problem. Entscheide, ob φ erfüllbar ist.

Theorem 2.61. (Cook 1970, Levin 1973)

SAT ist NP-vollständig.

Erinnerung: NP-Vollständigkeit

Erinnerung: NP-Vollständigkeit

Wir erinnern hier kurz an den Begriff der NP-Vollständigkeit eines Berechnungsproblems P bzw. einer Sprache $L \subseteq \Sigma^*$ über einem Alphabet Σ.

Wir führen hier die Begriffe nur soweit ein, wie sie zum Beweis des Satzes von Cook und Levin nötig sind.

Für eine ausführliche Behandlung des Stoffes verweisen wir auf die Vorlesung TheGI 2.

Nicht-deterministische Turing-Maschinen

Definition 2.62.

Eine nicht-deterministische (1-Band) Turing-Maschine (NTM) ist ein Tuple $M := (Q, \Sigma, \Gamma, \Delta, q_0, F)$, wobei:

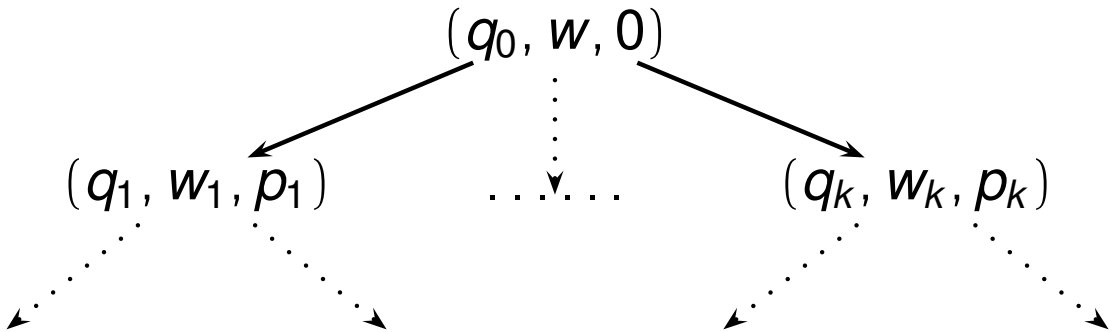
- Q ist eine endliche Menge von Zuständen
- Σ ist das endliche Eingabealphabet
- $\Gamma \supseteq \Sigma \dot\cup \{\square\}$ ist das endliche Arbeitsalphabet
- $\Delta \subseteq (Q \setminus F) \times \Gamma \times Q \times \Gamma \times \{-1, 0, 1\}$ ist die Übergangsrelation
- $q_0 \in Q$ ist der Startzustand
- $F \subseteq Q$ ist die Menge der Endzustände

Wir nehmen hier immer an, dass $\Sigma := \{0, 1\}$ und $\Gamma := \Sigma \dot\cup \{\square\}$.

Eine Konfiguration von M ist ein Tripel (q, w, p), wobei q der aktuelle Zustand, w die aktuelle Bandinschrift und p die aktuelle Position des Lese-/Schreibkopfes ist.

Der Lauf einer NTM

Die Berechnung einer nicht-deterministischen Turing-Maschine $(Q, \Sigma, \Gamma, \Delta, q_0, F)$ auf Eingabe $w \in \Sigma^*$ ist ein Berechnungsbaum:



Startkonfiguration. $(q_0, w, 0)$.

Stopkonfiguration. (q, w, p) hat keinen Nachfolger, wenn $q \in F$.

Nachfolger. (q, w, p) mit $w := w_1 \dots w_k$ und $p \leqslant k$ hat Nachfolger $(q', w_1 \dots w_{p-1} w' w_{p+1} \dots w_{k'}, p')$ für alle $(q, w_p, q', w', m) \in \Delta$ wobei

$$k' := \begin{cases} k+m & \text{wenn } p = k \text{ und } m = 1 \\ k & \text{sonst} \end{cases} \qquad p' := \begin{cases} 0 & \text{wenn } p = 0 \text{ und } m = -1 \\ p+m & \textit{sonst} \end{cases}$$

Nicht-deterministische Turing-Akzeptoren

Wir interessieren uns besonders für NTMs, die Sprachen entscheiden, d.h. die Eingabe entweder akzeptieren oder verwerfen.

Nicht-deterministischer Turing-Akzeptor: $\mathcal{M} := (Q, \Sigma, \Gamma, \Delta, q_0, F_a, F_r)$

Berechnungspfad oder Lauf von $\mathcal{M}$ auf Eingabe w:

Ein Pfad von der Anfangskonfiguration $(q_0, w, 0)$ zu einer Stopkonfiguration (q_f, w', p) mit $q_f \in F_a \cup F_r$ im Berechnungsbaum.

Akzeptierender Pfad: (q_f, w, p') ist akzeptierend, d.h. $q \in F_a$.
(wir sagen auch akzeptierender Lauf)

Verwerfender Pfad. (q_f, w, p') ist verwerfend, d.h. $q \notin F_a$.

Die durch eine NTM $\mathcal{M}$ akzeptierte Sprache:

$$\mathcal{L}(\mathcal{M}) := \{w \in \Sigma^* : \text{ es existiert ein akzeptierender Lauf von } \mathcal{M} \text{ auf } w\}$$

Nicht-deterministische Zeit- und Platzkomplexität

Definition 2.63.

Sei $\mathcal{M}$ ein nicht-deterministischer Turing-Akzeptor und seien $S, T : \mathbb{N} \to \mathbb{N}$ Funktionen.

1. $\mathcal{M}$ ist T-Zeitbeschränkt, wenn sie auf jeder Eingabe $w \in \Sigma^*$ nach $\leqslant T(|w|)$ Schritten hält.

   Genauer:

   - Auf Eingabe w ist die Länge jedes Berechnungspfades von $\mathcal{M}$ höchstens $T(|w|)$

2. $\mathcal{M}$ ist S-Platzbeschränkt wenn, auf Eingabe $w \in \Sigma^*$, jeder Berechnungspfad $\leqslant S(|w|)$ Zellen benutzt.

   (Hier nehmen wir an, dass die NTM ein separates Eingabeband hat, dessen Größe wir nicht mitzählen.)

Nicht-deterministische Komplexitätsklassen

Definition 2.64. Seien $T, S : \mathbb{N} \to \mathbb{N}$ monoton wachsende Funktionen.

1. $\text{NTIME}(T)$ ist die Klasse aller Sprachen $\mathcal{L}$ für die es eine T-zeitbeschränkte NTM gibt, die $\mathcal{L}$ entscheidet.
2. $\text{NSPACE}(S)$ ist die Klasse aller Sprachen $\mathcal{L}$, für die es eine S-platzbeschränkte NTM gibt, die $\mathcal{L}$ entscheidet.

Nicht-deterministische Komplexitätsklassen

Einige wichtige nicht-deterministische Komplexitätsklasen:

- Zeitkomplexitätsklassen:
  - NP $:= \bigcup_{d \in \mathbb{N}} \text{NTIME}(n^d)$
  - NEXPTIME $:= \bigcup_{d \in \mathbb{N}} \text{NTIME}(2^{n^d})$
- Platzkomplexitätsklassen:
  - NLOGSPACE $:= \bigcup_{d \in \mathbb{N}} \text{NSPACE}(d \log n)$
  - NPSPACE $:= \bigcup_{d \in \mathbb{N}} \text{NSPACE}(n^d)$
  - NEXPSPACE $:= \bigcup_{d \in \mathbb{N}} \text{NSPACE}(2^{n^d})$

Polynomial-Time Reductions

Zur Definition der NP-Vollständigkeit brauchen wir noch den Begriff der polynomialzeit many-one Reduktionen.

Definition 2.65. Eine Sprache $\mathcal{L}_1 \subseteq \Sigma^*$ ist polynomiell reduzierbar auf $\mathcal{L}_2 \subseteq \Sigma^*$, geschrieben $\mathcal{L}_1 \leqslant_p \mathcal{L}_2$, wenn es eine polynomialzeit berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ gibt, so dass für alle $w \in \Sigma^*$

$$w \in \mathcal{L}_1 \qquad \Longleftrightarrow \qquad f(w) \in \mathcal{L}_2.$$

Lemma 2.66.

Wenn $\mathcal{L}_1 \leqslant_p \mathcal{L}_2$ und $\mathcal{L}_2 \in \text{PTIME}$, dann auch $\mathcal{L}_1 \in \text{PTIME}$.

Wir brauchen meistens die Kontraposition: Wenn $\mathcal{L}_1 \leqslant_p \mathcal{L}_2$ und $\mathcal{L}_1 \notin \text{PTIME}$, dann auch $\mathcal{L}_2 \notin \text{PTIME}$.

NP-Vollständigkeit

Definition. Sei Σ ein Alphabet. Ein Problem $L \subseteq \Sigma^*$ ist NP-vollständig, wenn es in NP ist und jedes andere Problem in NP auf L in polynomieller Zeit reduziert werden kann.

- In TheGI 2 haben Sie schon gesehen, wie NP-Vollständigkeit eines Problems $P \in$ NP dadurch gezeigt werden kann, dass man ein anderes NP-vollständiges Problem auf P reduziert.
- Die Herausforderung besteht darin, ein erstes Problem zu finden, von dem man NP-Vollständigkeit ohne Reduktionen nachweisen kann.
- Dies gelang Ende der 60er Jahre Stephen Cook für das SAT Problem, der für seine bahnbrechenden Arbeiten zur NP-Vollständigkeit den Turing-Award erhielt.

Der Satz von Cook und Levin

Der Satz von Cook und Levin

Theorem 2.67. (Cook 1970, Levin 1973)

SAT ist NP-vollständig.

Beweis.

1. $\text{SAT} \in \text{NP}$

   Eine NTM kann einfach in Polynomialzeit eine Belegung der Variablen raten und verifizieren, dass sie die Eingabeformel erfüllt.

2. Zur NP-Vollständigkeit müssen wir noch zeigen: für jede Sprache $\mathcal{L} \in \text{NP}$,

   $$\mathcal{L} \leqslant_p \text{SAT}$$

   Sei also $\mathcal{L} \in \text{NP}$ mit $\mathcal{L} \subseteq \Sigma^*$.

   Für jede Eingabe $w \in \Sigma^*$ konstruieren wir eine aussagenlogische Formel $\varphi_{\mathcal{L},w}$, so dass

   $$w \in \mathcal{L} \iff \varphi_{\mathcal{L},w} \text{ erfüllbar.}$$

Beweis des Satzes von Cook und Levin

Da $\mathcal{L} \in \mathrm{NP}$ existiert eine 1-Band NTM $\mathcal{M} := (Q, \Sigma, \Gamma, q_0, \Delta, F_a, F_r)$ die $\mathcal{L}$ in Zeit $p(n)$, für ein Polynom p, akzeptiert.

Beachte. Aus Eingaben der Länge n ist die Länge jedes Berechnungspfads sowie die Länge jeder Konfiguration eines Pfades von $\mathcal{M}$ durch $p(n)$ beschränkt.

Idee. Wir beschreiben den Lauf von $\mathcal{M}$ auf EIngabe w durch eine Formel.

Beweis des Satzes von Cook und Levin

Beschreiben einer Konfiguration: Wir repräsentieren eine Konfiguration $(q, p, a_0 \dots a_{p(n)})$ durch eine Menge

$$\overline{C} := \{Q_q, P_i, S_{a,i} : \quad q \in Q, \quad a \in \Gamma \quad 0 \leqslant i < p(n)\}$$

von Variablen und die Wahrheitsbelegung β mit

$$\beta(Q_s) := \begin{cases} 1 & s = q \\ 0 & s \neq q \end{cases} \qquad \beta(P_s) := \begin{cases} 1 & s = p \\ 0 & s \neq p \end{cases} \qquad \beta(S_{a,i}) := \begin{cases} 1 & a = a_i \\ 0 & a \neq a_i \end{cases}$$

Variablen.

Q_q: Für alle $q \in Q$

Bedeutung: $\mathcal{M}$ ist im Zustand $q \in Q$

P_i: Für alle $0 \leqslant i \leqslant p(n)$

Bedeutung: der Kopf steht an Position i

$S_{a,i}$: Für alle $a \in \Gamma$ und $0 \leqslant i \leqslant p(n)$

Bedeutung: Bandposition i enthält Symbol a

Beweis des Satzes von Cook und Levin

Man beachte die folgende Formel $\textsc{Conf}(\overline{C})$ mit Variablen

$$\overline{C} := \left\{ Q_q, P_i, S_{a,i} : \quad q \in Q, \quad a \in \Gamma, \quad 0 \leqslant i < p(n) \right\}$$

$$\textsc{Conf} := \bigvee_{q \in Q} \left(Q_q \wedge \bigwedge_{q' \neq q} \neg Q_{q'} \right) \quad \wedge \quad \bigvee_{p \leqslant p(n)} \left(P_p \wedge \bigwedge_{p' \neq p} \neg P_{p'} \right) \wedge$$

$$\bigwedge_{1 \leqslant i \leqslant p(n)} \bigvee_{a \in \Gamma} \left(S_{a,i} \wedge \bigwedge_{b \neq a \in \Gamma} \neg S_{b,i} \right)$$

Definition 2.68. Für jede Belegung β von $\overline{C}$ definieren wir $\mathrm{conf}(\overline{C}, \beta)$ als

$$\left\{ (q, p, w_1 \dots w_{p(n)}) : \beta(Q_q) = 1, \beta(P_p) = 1, \beta(S_{w_i,i}) = 1, 0 \leqslant i \leqslant p(n) \right\}$$

Lemma 2.69. Wenn $\beta \models \textsc{Conf}(\overline{C})$, dann gilt $|\mathrm{conf}(\overline{C}, \beta)| = 1$.

Beweis des Satzes von Cook und Levin

Definition. Für jede Belegung β von $\overline{C}$ definieren wir $\mathrm{conf}(\overline{C}, \beta)$ als

$$\left\{ (q, p, w_0 \dots w_{p(n)}) : \begin{array}{l} \beta(Q_q) = 1, \\ \beta(P_p) = 1, \\ \beta(S_{w_i,i}) = 1 \text{ für alle } 0 \leqslant i \leqslant p(n) \end{array} \right\}$$

Lemma. Wenn $\beta \models \textsc{Conf}(\overline{C})$ erfüllt, dann $|\mathrm{conf}(\overline{C}, \beta)| = 1$.

Bemerkung. β kann auch für weitere Variablen neben $\overline{C}$ definiert sein.

Notation. Wir schreiben $\mathrm{conf}(\overline{C}, \beta) := (q, p, w)$.

$\mathrm{conf}(\overline{C}, \beta)$ ist eine potentielle Konfiguration von $\mathcal{M}$, aber eventuell ist sie nicht von der Startkonfiguration von $\mathcal{M}$ auf Eingabe w erreichbar.

Umgekehrt: Jede Konfiguration $(q, p, w_1 \dots w_{p(n)})$ induziert eine erfüllende Belegung β von conf.

Beweis des Satzes von Cook und Levin

Betrachte die folgende Formel $\textsc{Next}(\overline{C}, \overline{C}')$ definiert als

$$\textsc{Conf}(\overline{C}) \wedge \textsc{Conf}(\overline{C}') \wedge \textsc{Nochange}(\overline{C}, \overline{C}') \wedge \textsc{Change}(\overline{C}, \overline{C}').$$

$$\textsc{Nochange} := \bigvee_{0 \leqslant p \leqslant p(n)} \left(P_p \wedge \bigwedge_{\substack{i \neq p \\ a \in \Gamma}} (S_{a,i} \to S'_{a,i}) \right)$$

$$\textsc{Change} := \bigvee_{0 \leqslant p \leqslant p(n)} \left(P_p \wedge \bigvee_{\substack{q \in Q \\ a \in \Gamma}} (Q_q \wedge S_{a,p} \wedge \bigvee_{(q,a,q',b,m) \in \Delta} (Q'_{q'} \wedge S'_{b,p} \wedge P'_{p+m})) \right)$$

Lemma 2.70. Für jede Belegung β, die auf $\overline{C}, \overline{C}'$ definiert ist:

$$\beta \text{ erfüllt } \textsc{Next}(\overline{C}, \overline{C}') \iff \text{conf}(\overline{C}, \beta) \vdash_{\mathcal{M}} \text{conf}(\overline{C}', \beta)$$

Beweis des Satzes von Cook und Levin

Bisher definiert:

- $\textsc{Conf}(\overline{C})$: $\overline{C}$ beschreibt eine potentielle Konfiguration
- $\textsc{Next}(\overline{C}, \overline{C}')$: $\mathrm{conf}(\overline{C}, \beta) \vdash_{\mathcal{M}} \mathrm{conf}(\overline{C}', \beta)$

Startkonfiguration. Sei $w := w_0 \dots w_{n-1} \in \Sigma^*$ eine Eingabe von $\mathcal{M}$

$$\textsc{Start}_{\mathcal{M},w}(\overline{C}) := \textsc{Conf}(\overline{C}) \wedge Q_{q_0} \wedge P_0 \wedge \bigwedge_{i=0}^{n-1} S_{w_i,i} \wedge \bigwedge_{i=n}^{p(n)} S_{\square,i}$$

Eine Belegung β erfüllt $\textsc{Start}_{\mathcal{M},w}(\overline{C})$ gdw. $\overline{C}$ die Startkonfiguration von $\mathcal{M}$ auf Eingabe w repräsentiert.

Akzeptierende Stopkonfiguration.

$$\textsc{Acc-Conf}(\overline{C}) := \textsc{Conf}(\overline{C}) \wedge \bigvee_{q \in F_a} Q_q$$

Belegung β erfüllt $\textsc{Acc-Conf}(\overline{C})$ gdw. $\overline{C}$ repräsentiert eine akzeptierende Stopkonfiguration von $\mathcal{M}$.

Die komplette Reduktion: Variablen

Da $\mathcal{L} \in \mathrm{NP}$, existiert eine 1-Band NTM $\mathcal{M} := (Q, \Sigma, \Gamma, q_0, \Delta, F_a, F_r)$, die $\mathcal{L}$ in Zeit $p(n)$ entscheidet, für ein Polynom p.

Variablen:

$Q_{q,t}$: Für alle $q \in Q$, $0 \leqslant t \leqslant p(n)$

Bedeutung: $\mathcal{M}$ ist zur Zeit t im Zustand $q \in Q$

$P_{i,t}$: Für alle $0 \leqslant i, t \leqslant p(n)$

Bedeutung: Zur Zeit t ist der Kopf an Position i

$S_{a,i,t}$: Für alle $a \in \Sigma \dot{\cup} \{\square\}$ und $0 \leqslant i, t \leqslant p(n)$

Bedeutung: Zur Zeit t enthält Bandposition i das Symbol a

Notation. $\overline{C}_t := \{Q_{q,t},\ P_{i,t},\ S_{a,i,t} : \quad q \in Q, 0 \leqslant i \leqslant p(n), \quad a \in \Gamma\}$

Die komplette Reduktion: Die Formeln

Gegeben:

1-Band NTM $\mathcal{M} := (Q, \Sigma, \Gamma, q_0, \Delta, F_a, F_r)$, die $\mathcal{L}$ in Zeit $p(n)$ akzeptiert

Eingabe $w := w_0 \dots w_{n-1}$.

Konstruiere

$$\varphi_{\mathcal{M},w} := \quad \textsc{Start}_{\mathcal{M},w}(\overline{C}_0) \wedge \qquad C_0 \text{ kodiert Startkonf.}$$

$$\bigvee_{0 \leqslant t \leqslant p(n)} \begin{cases} \textsc{Acc-Conf}(\overline{C}_t) \wedge & \mathcal{M} \text{ akzeptiert nach } t \text{ Schritten} \\ \bigwedge_{0 \leqslant i < t} \textsc{Next}(\overline{C}_i, \overline{C}_{i+1}) & \overline{C}_0, \dots, \overline{C}_t \text{ kodieren Berechn.pfad} \end{cases}$$

Bemerkung. Eine akzeptierende oder verwerfende Stopkonfiguration hat keinen Nachfolger.

Lemma 2.71. $w \in \mathcal{L} \iff \varphi_{\mathcal{M},w}$ erfüllbar.

Der Satz von Cook und Levin

Theorem 2.67. (Cook 1970, Levin 1973)

SAT ist NP-vollständig.

Beweis.

1. $\text{SAT} \in \text{NP}$

   Eine NTM kann einfach in Polynomialzeit eine Belegung der Variablen raten und verifizieren, dass sie die Eingabeformel erfüllt.

2. Zur NP-Vollständigkeit müssen wir noch zeigen: für jede Sprache $\mathcal{L} \in \text{NP}$,

   $$\mathcal{L} \leqslant_p \text{SAT}$$

   Sei also $\mathcal{L} \in \text{NP}$ mit $\mathcal{L} \subseteq \Sigma^*$.

   Für jede Eingabe $w \in \Sigma^*$ konstruieren wir eine aussagenlogische Formel $\varphi_{\mathcal{L},w}$, so dass

   $$w \in \mathcal{L} \iff \varphi_{\mathcal{L},w} \text{ erfüllbar.}$$

Der DPLL Algorithmus

Der DPLL Algorithmus

Wir werden als nächstes einen auf der Resolution basierenden Algorithmus kennen lernen, um Formeln auf Unerfüllbarkeit zu testen.

Der DPLL-Algorithmus ist benannt nach seinen Erfindern, Davis, Putnam, Logemann, Loveland.

Der Algorithmus kombiniert backtracking mit Einheitsresolution, bei der nur Resolventen gebildet werden können, wenn mindestens eine der Klauseln nur ein Literal enthält.

Varianten des DPLL-Algorithmus bilden die Basis der meisten aktuellen SAT-Löser, wie z. B. BerkMin, zChaff, etc.

Der DPLL Algorithmus

Der Algorithmus arbeitet auf Formeln $\varphi := \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n_i} L_{i,j}$ in KNF.

Repr. φ als Klauselmenge: $\Phi := \big\{\{L_{1,1}, \ldots, L_{1,n_1}\}, \ldots, \{L_{n,1}, \ldots, L_{n,n_n}\}\big\}$.

ALGORITHMUS DPLL(Φ)

Eingabe. Klauselmenge $\Phi := \big\{\{L_{1,1}, \ldots, L_{1,n_1}\}, \ldots, \{L_{n,1}, \ldots, L_{n,n_n}\}\big\}$

Ausgabe. entscheide, ob Φ erfüllbar ist.

Algorithmus. Wenn Φ leer ist, gib erfüllbar zurück.
Wenn Φ die leere Klausel $\square$ enthält, gib unerfüllbar zurück.

Unit Clause/Boolean Constraint Propagation (bcp):

Solange Φ eine Einheitsklausel $\{L\}$ enthält "setze $L := 1$"
d.h. entferne alle Klauseln, die L enthalten und
entferne $\overline{L}$ aus allen anderen Klauseln

Anderenfalls, wähle Literal L, das in Φ vorkommt.

Ergibt DPLL($\Phi \cup \{\{L\}\}$) oder DPLL($\Phi \cup \{\{\overline{L}\}\}$) erfüllbar,
gib erfüllbar zurück, sonst unerfüllbar.

Beispiel

$\{V_1\, V_2\, V_7, \neg V_2\, V_1, \neg V_1\, V_3, \neg V_3 \neg V_1, \neg V_7\, V_4\, V_5, \neg V_4\, V_5, \neg V_5\, V_6, \neg V_6 \neg V_5\}$

Wähle $V_7 := 0$

$\{V_1\, V_2, \neg V_2\, V_1, \neg V_1\, V_3, \neg V_3 \neg V_1, \neg V_4\, V_5, \neg V_5\, V_6, \neg V_6 \neg V_5\}$

Wähle $V_1 := 0$

$\{V_2, \neg V_2, \neg V_4\, V_5, \neg V_5\, V_6, \neg V_6 \neg V_5\}$ bcp $V_2 := 1 \rightsquigarrow \square$

Wähle $V_1 := 1$

$\{V_3, \neg V_3, \neg V_4\, V_5, \neg V_5\, V_6, \neg V_6 \neg V_5\}$ bcp $V_3 := 1 \rightsquigarrow \square$

Wähle $V_7 := 1$

$\{\neg V_2\, V_1, \neg V_1\, V_3, \neg V_3 \neg V_1,\ V_4\, V_5, \neg V_4\, V_5, \neg V_5\, V_6, \neg V_6 \neg V_5\}$

Wähle $V_5 := 0$

$\{\neg V_2\, V_1, \neg V_1\, V_3, \neg V_3 \neg V_1,\ V_4, \neg V_4\}$ bcp $V_4 := 1 \rightsquigarrow \square$

Wähle $V_5 := 1$

$\{\neg V_2\, V_1, \neg V_1\, V_3, \neg V_3 \neg V_1,\ V_6, \neg V_6\}$ bcp $V_6 := 1 \rightsquigarrow \square$

DPLL vs. Resolution

Es besteht ein enger Zusammenhang zwischen Resolutionswiderlegungen und Widerlegungen einer Formel durch den DPLL-Algorithmus.

Dazu stellt man einen Lauf des DPLL-Algorithmus als Entscheidungsbaum dar.

Hat der Baum die Höhe h, so gibt es auch eine (baumartige) Resolutionswiderlegung gleicher Höhe.

Der DPLL-Algorithmus ist also nicht “schneller” als die Resolution.

Der DPLL-Algorithmus

In praktischen Implementierungen des Algorithmus' werden verschiedene Optimierungen verwendet.

Auswahlregel: Welches Literal wird beim Verzweigen gewählt

Conflict Analysis: Bei einem Backtracking Schritt wird der Grund der Unerfüllbarkeit (Conflict) analysiert und intelligenter zurück gesprungen.

Clause Learning: Aus der Konfliktanalyse werden neue Klauseln generiert, die zur Formel hinzugenommen werden.

Dies soll verhindern, dass in den gleichen Konflikt hineingelaufen wird.

Random restarts: Bisweilen wird ein DPLL-Lauf abgebrochen und neu angefangen. Die gelernten Klauseln bleiben erhalten.

Eine Anwendung aus der künstlichen Intelligenz

Aussagenlogik in der KI

Eine mögliche Anwendung der Aussagenlogik sind sogenannte constraint satisfaction Probleme (CSPs).

Definition. Ein Constraint Satisfaction Problem besteht aus

- einer Menge V von Variablen
- einem Wertebereich D
- und einer Menge C von constraints, d.h. Tupeln $((v_1, \ldots, v_n), R)$ wobei $R \subseteq D^n$.

Eine Lösung eines CSPs ist eine Abbildung $f : V \to D$, so dass für alle constraints $((v_1, \ldots, v_n), R) \in C$ gilt:

$$(f(v_1), \ldots, f(v_n)) \in R$$

CSPs sind eine in der künstlichen Intelligenz viel untersuchte Problemklasse, da viele natürliche Probleme als CSPs modelliert werden können.

Beispiel für ein CSP

Ein einfaches Beispiel ist die Stundenplangenerierung.

Beispiel. Eine (kleine) Schule mit nur einem Klassenzimmer hat drei Lehrer die jeweils ein Fach D, M oder P unterrichten und zwei Klassen, die in jedem der drei Fächer unterrichtet werden sollen.

Variablenmenge V. $D_1, D_2, M_1, M_2, P_1, P_2$

Bedeutung: M_1: Klasse 1 wird durch den Mathematiklehrer unterrichtet.

Wertebereich D. Mögliche Unterrichtszeiten $8, 9, 10, \ldots$.

Constraint-Menge C. Sei $R_{\neq} := \{(t, t') \in D \times D : t \neq t'\}$.

$((D_1, D_2), R_{\neq}), \quad ((M_1, M_2), R_{\neq}), \quad ((P_1, P_2), R_{\neq})$

"Kein Lehrer hält zwei Klassen gleichzeitig"

Für alle $i = 1, 2, 3$: $((M_i, D_i), R_{\neq}), \quad ((M_i, P_i), R_{\neq}), \quad ((P_i, D_i), R_{\neq})$

"Keine Klasse hat zwei Stunden gleichzeitig"

Sudoku als CSP

Ein weiteres Beispiel ist das Spiel Sudoku.

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

Ziel ist es, die fehlenden Positionen so mit den Zahlen $1, \ldots, 9$ zu füllen, dass in jeder Zeile und jeder Spalte und in jedem Block jede der Zahlen $1, \ldots, 9$ genau einmal vorkommt.

Eine Lösung durch die Aussagenlogik

Wir formalisieren ein gegebenes Sudoku in der Aussagenlogik.

Variablen. $X_{i,j}^c$ für alle $1 \leqslant i, j, c \leqslant 9$

Idee: $X_{i,j}^c$ wird mit 1 belegt, wenn in Position (i, j) die Zahl c steht.

Generische Formeln.

- $\bigwedge_{1 \leqslant i,j \leqslant 9} \bigvee_{c=1}^{9} X_{i,j}^c$ "jede Position erhält eine Zahl"
- $\bigwedge_{1 \leqslant i,j \leqslant 9} \bigwedge_{1 \leqslant c < c' \leqslant 9} \neg(X_{i,j}^c \wedge X_{i,j}^{c'})$ "jede Position erhält $\leqslant$ 1 Zahl"
- Für alle $1 \leqslant i, c \leqslant 9$ und $1 \leqslant j < j' \leqslant 9$: $\neg(X_{i,j}^c \wedge X_{i,j'}^c)$
  "Keine Zeile enthält zwei gleiche Einträge"
- Für alle $1 \leqslant j, c \leqslant 9$ und $1 \leqslant i < i' \leqslant 9$: $\neg(X_{i,j}^c \wedge X_{i',j}^c)$
  "Keine Spalte enthält zwei gleiche Einträge"
- Die entsprechende Aussage für jeden Block:
  $\neg(X_{i,j}^c \wedge X_{i',j'}^c)$ für alle $1 \leqslant c \leqslant 9$ und i, j, i', j' mit
  - $(i, j) \neq (i', j')$ und
  - $i \div 3 = j \div 3 = i' \div 3 = j' \div 3$